Частное учреждение образования

«Колледж бизнеса и права»

ОТЧЕТ

ПО ПРОИЗВОДСТВЕННОЙПРЕДДИПЛОМНОЙ ПРАКТИКЕ

ОП Т.992002

Руководитель практики

от предприятия (М.В.Кривель )

М.П.

Руководитель практики

от колледжа (Е.Н. Коропа )

Учащийся (М.М. Быковский)

2023

Изм.

Лист

№ докум.

Подпись

Дата

Лист

2

ОП Т.992002

Разраб.

Быковский М.М.

Провер.

Коропа Е.Н.

Т.контр.

Н.контр.

Утверд.

*Отчет по преддипломной практике*

Лит

Листов

КБП

У

СОДЕРЖАНИЕ

[Введение 3](#_Toc133952223)

[1 Описание задачи 5](#_Toc133952224)

[1.1 Анализ предметной области 5](#_Toc133952225)

[1.2 Постановка задачи 6](#_Toc133952226)

[2 Проектирование системы 8](#_Toc133952227)

[2.1 Требования к приложению 8](#_Toc133952228)

[2.2 Проектирование модели 8](#_Toc133952229)

[2.3 Проектирование структуры базы данных 10](#_Toc133952230)

[2.4 Концептуальный прототип 13](#_Toc133952231)

[3 Описание реализации программного средства 16](#_Toc133952232)

[3.1 Инструменты разработки и применяемые технологии 16](#_Toc133952233)

[3.2 Порядок авторизации пользователей 19](#_Toc133952234)

[3.3 Организация данных 20](#_Toc133952235)

[3.4 Функции: логическая и физическая организация 22](#_Toc133952236)

[3.5 Входные и выходные данные 23](#_Toc133952237)

[3.6 Функциональное тестирование 24](#_Toc133952238)

[3.7 Описание справочной системы 27](#_Toc133952239)

[4 Применение 29](#_Toc133952240)

[4.1 Назначение программного средства 29](#_Toc133952241)

[4.2 Условия применения 29](#_Toc133952242)

[5 Охрана труда и окружающей среды 31](#_Toc133952243)

[Заключение 35](#_Toc133952244)

[Список информационных источников 36](#_Toc133952245)

[Приложение А Текст программных модулей 38](#_Toc133952246)

[Приложение Б Формы выходных документов 54](#_Toc133952247)

# **Введение**

Подготовка к олимпиадам может быть сложной задачей, требующей сбора и обработки большого количества данных. Часто этот процесс может быть утомительным и трудоемким, что приводит к ошибкам и неэффективности. Для решения этой проблемы необходимо разработать программное средство, которое автоматизирует сбор и обработку данных для подготовки и проведения олимпиад.

Цель программного средства ­ упростить процесс сбора и обработки данных, связанных с подготовкой и проведением олимпиады: регистрация участников, составление расписания мероприятий, управление оценками и результатами, а также подготовка отчетов. Программное средство обеспечит интуитивно понятный и удобный интерфейс, который позволит организаторам эффективно управлять всем процессом.

Актуальность данной задачи значительна, учитывая растущий интерес к участию в олимпиадных соревнованиях во всем мире. Эти соревнования предоставляют учащимся возможность продемонстрировать свои навыки и знания в различных областях, включая математику, физику и информатику. Однако организация таких соревнований может оказаться сложной задачей, особенно для небольших организаций и учреждений с ограниченными ресурсами. Автоматизируя задачи сбора и обработки данных, предлагаемое программное средство может значительно снизить нагрузку на организаторов и сделать этот процесс более доступным для широкой аудитории.

В целом, разработка программного инструмента для автоматизации сбора и обработки данных для подготовки и проведения олимпиад может упростить этот процесс и сделать его более доступным для организаторов. Это, в свою очередь, может стимулировать более активное участие и способствовать росту олимпиадных соревнований во всем мире.

В первом разделе «Описание задачи» проводится исследование предметной области, описываются бизнес-процессы, происходящие в предметной области. Кроме того, описываются задачи из иследованной предметной области, которые могут быть решены при использовании программы, приводятся способы рещения задач. Описываются существующие аналоги.

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

Третий раздел «Описание реализации программного обеспечения» приведится описание разарботанного программного средства: применяемые интсрументы разработки и технологии, порядок авторизации пользователей, органихация данных в рамках выбранной сиситемы управления базамми данных, логическая и физическая орнащинацияфукнций, входные и выходные данные. Кроме того, приводится тестирование на основании подготовленных тест-кейсов, описывается справочная система.

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

В заключении проанализировано созданное программное средство, определена степень соответствия поставленной задачи и выполненной работы.

В приложении А представлен текст программных модулей.

В приложении Б представлены формы выходных документов.

Графическая часть представлена диаграммами вариантов использования, классов, деятельности.

1. Описание задачи
   1. Анализ предметной области

Олимпиада - это соревновательное мероприятие, в котором участвуют студенты из различных школ, колледжей и университетов. Это возможность для студентов продемонстрировать свои академические знания и навыки, а также посоревноваться со своими сверстниками. Однако подготовка и проведение олимпиады предполагает сбор и обработку большого количества данных, что может стать сложной задачей для организаторов. Для преодоления этой проблемы необходим автоматизированный инструмент, который может упростить этот процесс и сделать его более эффективным.

Предметная область инструмента для автоматизации сбора и обработки данных для подготовки и проведения олимпиады включает несколько бизнес-процессов, в том числе регистрацию учащихся и школ, создание заданий, выставление баллов за задания и формирование отчетов. Инструмент должен быть разработан с учетом потребностей как учащихся, так и организаторов, облегчая им управление всем процессом.

Бизнес-процессы в данной предметной области можно описать следующим образом:

* регистрация: учащиеся и школы должны зарегистрироваться для участия в олимпиаде. Процесс регистрации должен быть удобным для пользователя и позволять легко зарегистрироваться;
* создание заданий: инструмент должен позволять организаторам создавать задания для учащихся. Задания должны быть составлены таким образом, чтобы проверить знания и навыки учащихся по различным учебным предметам;
* оценка заданий: после выполнения заданий учащимися они должны быть оценены организаторами. Инструмент должен иметь систему подсчета баллов, обеспечивающую справедливое и точное оценивание;
* формирование отчетов: после завершения процесса подсчета баллов инструмент должен генерировать отчеты, содержащие подробную информацию об успеваемости учащихся и школ.

Автоматизированный инструмент сбора и обработки данных для подготовки и проведения олимпиады имеет решающее значение для успеха мероприятия. Без этого инструмента организаторам пришлось бы полагаться на ручные процессы, которые отнимают много времени и чреваты ошибками. Инструмент может упростить процесс, сделать его более эффективным и позволить организаторам сосредоточиться на других аспектах мероприятия.

Одной из ключевых особенностей инструмента является возможность создания заданий для студентов. Задания должны быть составлены таким образом, чтобы проверить знания и навыки студентов по различным учебным предметам. Инструмент также должен иметь систему, позволяющую создавать различные уровни сложности, чтобы студенты могли проходить тестирование в соответствии со своими способностями.

Одной из основных проблем в данной предметной области является управление данными. Процесс сбора и обработки данных может занимать много времени и быть сопряжен с ошибками. Другой проблемой является создание справедливых и точных заданий, которые проверяют знания и навыки студентов по различным учебным предметам. Инструмент должен быть разработан таким образом, чтобы решить эти проблемы и сделать процесс более эффективным.

В заключение следует отметить, что предметная область инструмента для автоматизации сбора и обработки данных для подготовки и проведения олимпиады включает несколько бизнес-процессов, в том числе регистрацию, создание заданий, подсчет баллов за задания и формирование отчетов. Этот инструмент важен для успешного проведения мероприятия, поскольку он может упростить этот процесс, сделав его более эффективным, и позволить организаторам сосредоточиться на других аспектах мероприятия. Возможность создания заданий для учащихся является ключевой особенностью инструмента, и он должен быть разработан таким образом, чтобы решать проблемы, связанные с управлением данными и созданием заданий.

* 1. Постановка задачи

Исходя из анализа предметной области, существует несколько бизнес-проблем, которые можно решить с помощью автоматизированного инструмента для сбора и обработки данных для подготовки и проведения олимпиады. Ниже приводится подробная постановка проблемы:

* управление данными: ручной процесс сбора и обработки данных может отнимать много времени и быть чреват ошибками. Автоматизированный инструмент может упростить этот процесс, сделав его более эффективным, и снизить риск ошибок;
* создание заданий: ручной процесс создания заданий может быть сложным, особенно когда речь идет об обеспечении справедливости и точности заданий. Автоматизированный инструмент позволяет создавать справедливые и точные задания, которые проверяют знания и навыки студентов по различным учебным предметам;
* оценка заданий: ручной процесс оценивания заданий может быть субъективным и подверженным ошибкам. Автоматизированный инструмент может иметь систему подсчета баллов, которая обеспечивает справедливое и точное оценивание.

В настоящее время подготовка и проведение олимпиады управляются организаторами вручную. Это предполагает значительный объем сбора и обработки данных, что может отнимать много времени и чревато ошибками. Составление заданий также осуществляется вручную, что может быть сложной задачей, особенно когда речь идет об обеспечении справедливости и точности заданий. Оценка заданий также производится вручную, что может быть субъективным и чревато ошибками.

На рынке существует несколько аналогичных решений, но они либо слишком дороги, либо слишком сложны в использовании. Некоторые из этих решений разработаны специально для академических учреждений, другие являются более общими. Однако ни одно из них не предоставляет полного решения для подготовки и проведения олимпиады. Некоторые из этих решений также имеют ограниченные возможности, что делает их менее эффективными для управления всем процессом.

Ручной процесс управления подготовкой и проведением олимпиады может отнимать много времени, приводить к ошибкам и быть сложным. Автоматизированный инструмент для сбора и обработки данных для подготовки и проведения олимпиады может упростить этот процесс, сделав его более эффективным, и снизить риск ошибок. Этот инструмент также может создавать справедливые и точные задания, которые проверяют знания и навыки студентов по различным учебным предметам, и иметь систему подсчета баллов, позволяющую выставлять справедливые и точные оценки. Хотя на рынке существуют аналогичные решения, они либо слишком дороги, либо слишком сложны в использовании, что делает их менее эффективными для управления всем процессом. Поэтому автоматизированный инструмент для сбора и обработки данных для подготовки и проведения олимпиады может стать эффективным решением существующих проблем, связанных с ручным процессом управления мероприятием.

1. Проектирование системы
   1. Требования к приложению

Разрабатываемое приложение для автоматизации работы управления отелямидолжно иметь понятный и удобный в использовании интерфейс, чтобы взаимодействие между программой и пользователем было максимально упрощено. Для того чтобы интерфейс был понятен пользователю, необходимо оформить диалог приложения с пользователем путем использования кнопочного меню, позволяющего пользоваться ключевым функционалом. Программное приложение должно обеспечивать оперативноереагирование на все действия пользователя.

Все входные данные должны проверяться на ошибки. При совершении пользователем неверных действий или введении неверных данных будут выводиться сообщения с пояснением ошибок.

Функциональными требованиями разрабатываемого приложения для автоматизации работы управления отелем являются:

* регистрация участников олимпиды;
* просмотр информации об участниках олимпиады;
* возможность публикации заданий для участников олимпиады;
* возможность публикации задания до начала олимпиады;
* возможность активации задания;
* возможность получения решений от участников в заданные сроки.

Минимальные системные требования для оптимальной работы программного средства является персональный компьютер (ПК) со следующими характеристиками:

* 32 – разрядный (x86) или 64 – разрядный (x64) процессор с тактовой частотой 1 ГГц или выше;
* 1 ГБ (для 32 – разрядного процессора) или 2 ГБ (для 64 – разрядного процессора) ОЗУ;
* 16 ГБ (для 32 – разрядной системы) или 20 ГБ (для 64 – разрядной системы) свободного места на жестком диске;
* графическое устройство DirectX 9 с драйвером WDDM 1.0 или более поздней версии.
  1. Проектирование модели

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

Основными функциями программы являются:

* регистрация участников олимпиды;
* просмотр информации об участниках олимпиады;
* возможность публикации заданий для участников олимпиады;
* возможность публикации задания до начала олимпиады;
* возможность активации задания;
* возможность получения решений от участников в заданные сроки.

Диаграмма вариантов использования представлена в графической части на листе 1.

Диаграмма классов – диаграмма, определяющая типы классов системы и различного рода статические связи, которые есть между ними. На диаграммах классов изображаются также атрибуты классов, операции классов и ограничения, которые накладываются на связи между классами.

ОсновнымиклассамибудутявлятьсяApplicationUser, Category, Question, Quizz, Answers,Event, Reuslt, Group. Класс ApplicationUserпредставляет пользователя программы, т.е преподавателя или ученика. Класс Category служит для классификации различных типов викторин. Класс Question представляет собой вопрос, который будет находится в викторине. Класс Quizz является комплексной сущность, содержащей ряд вопросов. Класс Answer служит для хранения ответов к соответствующим вопросам. Класс Event служит для управлениемучастниками и викториной. Класс Group предназначен для хранения учеников, которые будут участвовать в той или иной викторине. Диаграмма классов представлена в графической части на листе 2.

Диаграмма деятельности UML (Unified Modeling Language) - это графическое представление потока деятельности, действий или задач в системе. Она используется для моделирования потока управления в программном приложении и дает четкое представление о последовательности действий, выполняемых в конкретном сценарии или процессе. На диаграмме деятельности UML действия изображаются в виде закругленных прямоугольников, а переходы между действиями - стрелками. Решения, циклы и параллельные потоки также могут быть представлены на диаграммах деятельности UML. Основная цель диаграммы деятельности UML - обеспечить четкое и ясное понимание поведения системы, выявить потенциальные узкие места или области для улучшения, а также облегчить общение между разработчиками, заинтересованными сторонами и клиентами. Диаграмма деятельности представлена в графической части на листе 3.

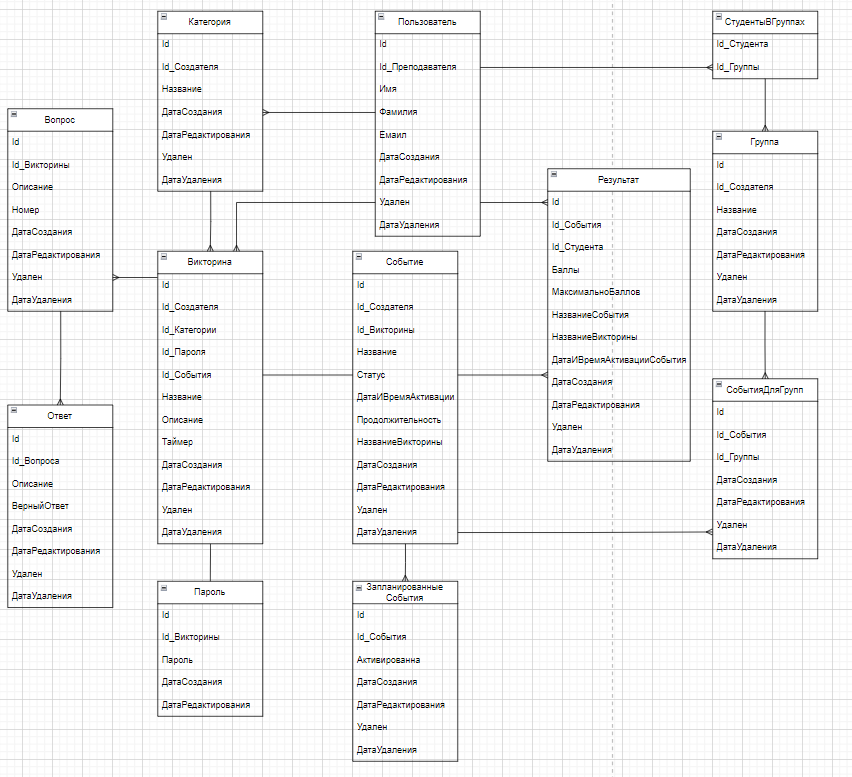
* 1. Проектирование структуры базы данных

ERD (Entity-RelationshipDiagram) - это графическое представление сущностей и их связей друг с другом. Она используется при проектировании баз данных для иллюстрации отношений между таблицами, сущностями, атрибутами и ключами.

ERD состоят из трех основных компонентов: сущностей, атрибутов и отношений. Сущности - это объекты или понятия в системе, такие как клиенты или заказы. Атрибуты описывают характеристики сущности, например, имя или адрес клиента. Отношения описывают связи между сущностями, например, клиент размещает заказ.

ERD можно использовать для визуализации структуры базы данных и выявления потенциальных проблем или областей для улучшения. Они также обеспечивают четкий и лаконичный способ передачи информации о структуре базы данных заинтересованным сторонам и разработчикам.

ER диаграмма представлена на рисунке 1.1.

Рисунок 1.1 – ER диаграмма

ER-диаграмма содержит следующие сущности:

* «Пользователь»;
* «Категория»;
* «Викторина»;
* «Вопрос»;
* «Ответ»;
* «Пароль»;
* «Событие»;
* «ЗапланированныеСобытия»;
* «Результат»;
* «СобытияДляГрупп»;
* «Группа»;
* «СтудентыВГруппах».

Сущность «Пользователь» представляет собой конечного пользователя, т.е учащегося или преподавателя. Атрибуты этой сущности включают:

* «Id»: уникальный идентификатор пользователя;
* «Id\_Преподавателя»: уникальный идентификатор преподавателя, который добавил
* текущего ученика;
* «Имя»: имя пользователя;
* «Фамилия»: фамилия пользователя;
* «Емаил»: почтовый адрес пользователя;
* «ДатаСоздания»: хранит дату и время создания учетной записи пользователя;
* «ДатаРедактирования»: хранит дату и время последнего изменения учетной записи
* пользователя;
* «Удален»: указывает, была ли удалена учетная запись пользователя;
* «ДатаУдаления»: хранит дату и время удаления учетной записи пользователя.

Сущность «Викторина» представляет собой викторину, которую будут проходить ученики. Атрибуты этой сущности включают:

* «Id»: уникальный идентификатор викторины;
* «Id\_Создателя»: уникальный идентификатор создателя, который создал викторину;
* «Id\_Категории»: уникальный идентификатор соответствующей категории;
* «Id\_Пароля»: уникальный идентификатор соответствующего пароля;
* «Id\_События»: уникальный идентификатор соответствующего события;
* «Название»: имя викторины;
* «Описание»: информация о викторине;
* «Таймер»: время проведения викторины;
* «ДатаСоздания»: хранит дату и время создания викторины;
* «ДатаРедактирования»: хранит дату и время последнего изменения викторины;
* «Удален»: указывает, была ли удалена викторина;
* «ДатаУдаления»: хранит дату и время удаления викторины.

Сущность «Вопрос» представляет собой единицу задания для викторины. Атрибуты этой сущности включают:

* «Id»: уникальный идентификатор вопроса;
* «Id\_Викторины»: уникальный идентификатор викторины к которой относится вопрос;
* «Описание»: содержание вопроса;
* «Номер»: порядковый номер вопроса;
* «ДатаСоздания»: хранит дату и время создания вопроса;
* «ДатаРедактирования»: хранит дату и время последнего изменения вопроса;
* «Удален»: указывает, был ли удаленвопроса;
* «ДатаУдаления»: хранит дату и время удаления вопроса.

Сущность «Ответ» представляет ответ для соответствующего вопроса. Атрибуты этой сущности включают:

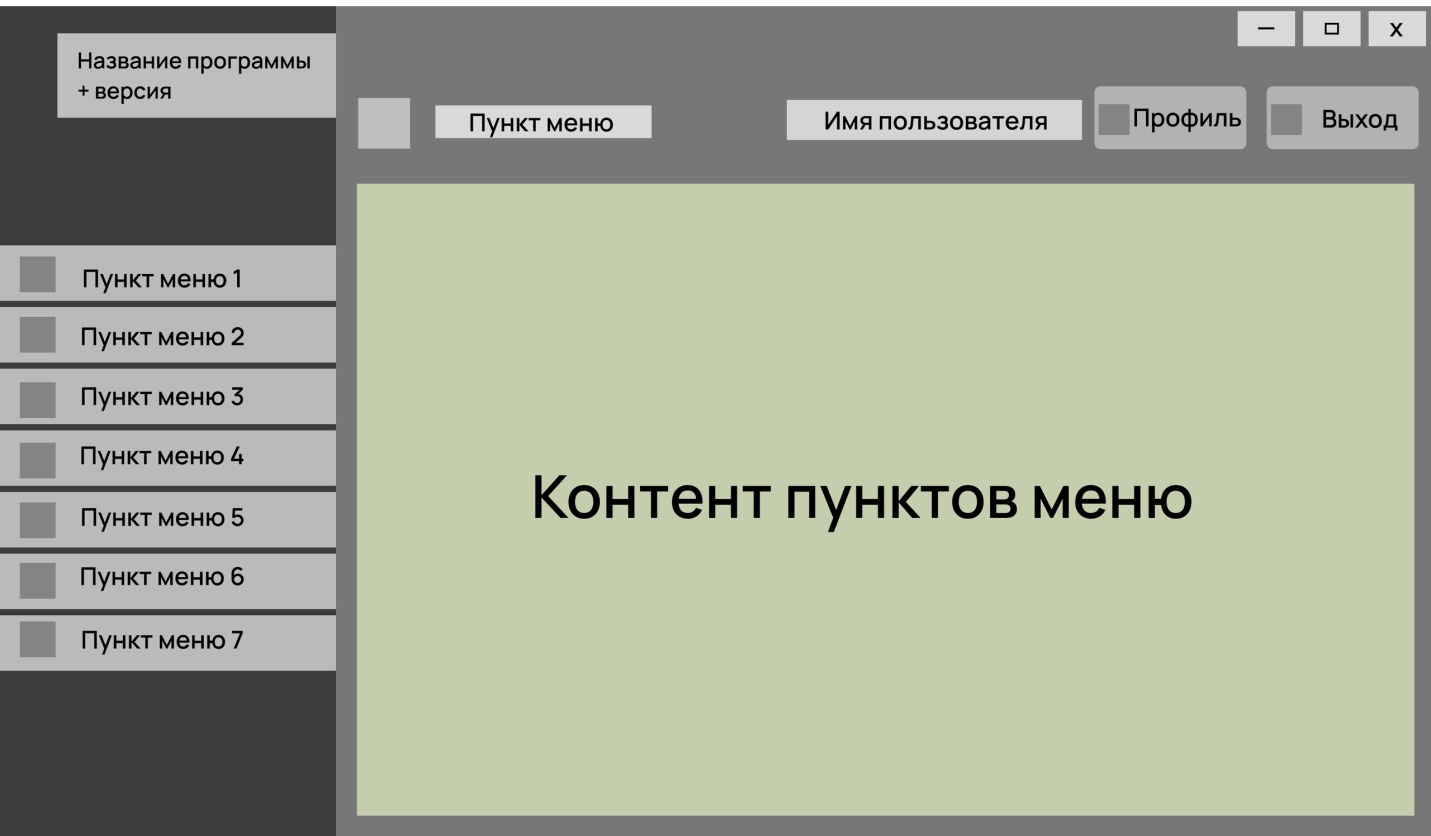
* «Id»: уникальный идентификатор ответа;
* «Id\_Вопроса»: уникальный идентификатор вопроса к которому относится ответ;
* «Описание»: содержание ответа;
* «ВерныйОтвет»: указывает, был ли дан верный ответ на вопрос;
* «ДатаСоздания»: хранит дату и время создания ответа;
* «ДатаРедактирования»: хранит дату и время последнего изменения ответа;
* «Удален»: указывает, был ли удален ответ;
* «ДатаУдаления»: хранит дату и время удаления ответа.
  1. Концептуальный прототип

Концептуальный прототип представляет собой описание внешнего пользовательского интерфейса – систему меню и форм.

При запуске программы будет появляться окно авторизации с возможностью перейти на формырегистрации преподавателя и восстановления пароля.При успешной авторизации пользователю (преподавателю), будет отображено главноеокно на котором можно будет воспользоваться всем функционалом приложения.

Главная форма будет иметь кнопки для выбора действия, поля для ввода данных пользователем, позволяющие использовать функционал программы, а также получать справочную информацию о работе с приложением. При нажатии на любой из пунктов меню, будет отображаться соответствующая информация в центральном окне.

Перечисленные слева пункты меню буду служить для выполенния таких операций как: создание/редактирование/удаление групп, категорий, викторин, вопросов, ответов. Макет главногоокна представлен на рисунке 2.1.

Рисунок 2.1– Макет главной формы

В поле «Электронная почта» пользователь должен ввести свой адрес электронной почты, который будет использоваться для входа в систему. В полях «Имя» и «Фамилия» пользователь должен указать свое имя и фамилию. В поле Password пользователь должен ввести пароль, который будет использоваться для защиты его учетной записи. В поле «Повторите пароль» пользователь должен повторить пароль для подтверждения его правильности и избежания ошибок при вводе. Макет окна регистрации преподавателя представлен на рисунке 2.2



Рисунок 2.2– Макет окна регистрации преподавателя

В поле «Электронная почта» пользователь должен ввести адрес электронной почты, на который придет код подтверждения, который необходимо будет вставить в поле «Код». После успешного подтверждения кода, пользователь должен будет ввести новый пароль и нажать кнопку подтверждения.

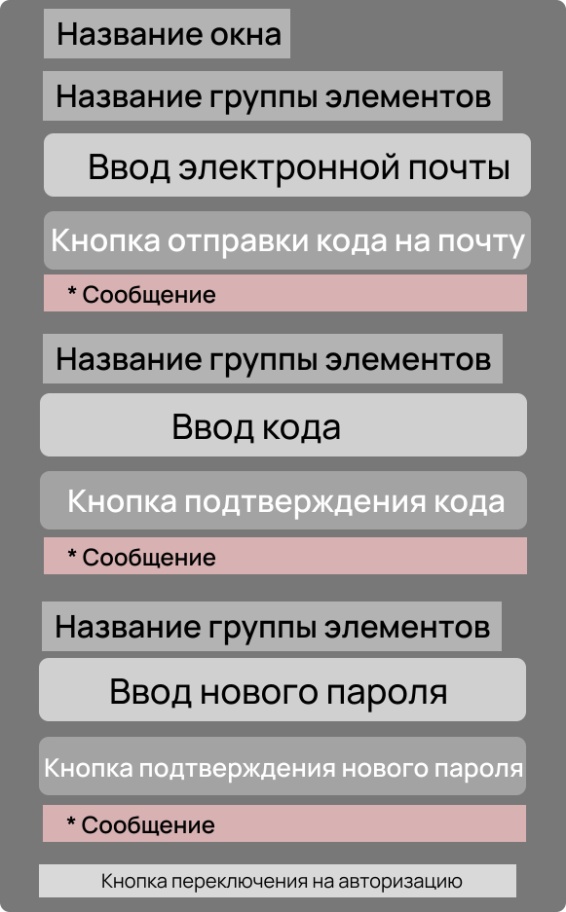


Рисунок 2.3– Макет окна восстановления пароля

1. Описание реализации программного средства
   1. Инструменты разработки и применяемые технологии

Инструментами разработки будут являться:

* операционная система Windows 10 x64;
* среда программирования Visual Studio 2022;
* язык программирования C#;
* технология Windows Presentation Foundation;
* технология Entity Framework;
* программное средство для создания UML диаграмм и схем Draw.io.

Программное приложение будет разрабатываться и тестироваться под управлением операционной системы Windows 10.

Windows 10 – операционная система, разработанная корпорацией Microsoft, используемая преимущественно для персональных компьютеров и рабочих станций, предоставляющая большой набор инструментов для разработчиков программного обеспечения. Данная операционная система является одной из самых распространенных на данный момент и обладает рядом существенных преимуществ:

Интегрированная среда разработки (IDE) — это многофункциональная программа, которая поддерживает многие аспекты разработки программного обеспечения. Интегрированная среда разработки Visual Studio — это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений []. Помимо стандартного редактора и отладчика, которые есть в большинстве сред IDE, Visual Studio включает в себя компиляторы, средства автозавершения кода, графические конструкторы и многие другие функции для улучшения процесса разработки.

C# является универсальным, безопасным в отношении типов, объектно–ориентированным языком программирования []. Хоть C# и является объектно–ориентированным языком, но поддерживает также и компонентно–ориентированное программирование, Разработка современных приложений все больше тяготеет к созданию программных компонентов в форме автономных и самоописательных пакетов, реализующих отдельные функциональные возможности. Главная особенность таких компонентов в том, что они представляют собой модель программирования со свойствами, методами и событиями, У них есть атрибуты, предоставляющие декларативные сведения о компоненте, Они включают в себя собственную документацию, C# предоставляет языковые конструкции, непосредственно поддерживающие такую концепцию работы. Благодаря этому C# подходит для создания и применения программных компонентов.

Windows Presentation Foundation (WPF) - это мощная и гибкая основа для создания настольных приложений с богатыми пользовательскими интерфейсами. WPF позволяет разработчикам создавать визуально потрясающие пользовательские интерфейсы, используя декларативную модель программирования и XAML, язык разметки для описания элементов пользовательского интерфейса и их свойств.

WPF предоставляет полный набор элементов управления и панелей компоновки, которые можно легко настраивать и стилизовать для создания уникальных и сложных пользовательских интерфейсов. Кроме того, WPF поддерживает привязку данных и мощную систему анимации, что позволяет разработчикам создавать динамичные и увлекательные пользовательские интерфейсы.

Одной из ключевых особенностей WPF является возможность отделить логику пользовательского интерфейса от базовой бизнес-логики с помощью паттерна Model-View-ViewModel (MVVM). Это облегчает сопровождение и тестирование кода приложения, а также улучшает общую масштабируемость и ремонтопригодность приложения.

Entity Framework - это каркас объектно-реляционного отображения (ORM), который позволяет разработчикам работать с реляционными базами данных, используя объекты .NET. Она является частью Microsoft .NET Framework и предоставляет набор библиотек и инструментов для создания приложений, управляемых данными.

Entity Framework предоставляет разработчикам ряд преимуществ, включая:

* абстракция базовой базы данных: Разработчики могут работать с объектами и не беспокоиться о схеме базы данных. Entity Framework автоматически сопоставляет объекты со схемой базы данных;
* автоматическая генерация SQL-запросов: Entity Framework генерирует SQL-запросы на основе операторов LINQ (Language Integrated Query), которые пишут разработчики. Это означает, что разработчикам не нужно писать сложные SQL-запросы вручную;
* поддержка нескольких баз данных: Entity Framework поддерживает множество систем баз данных, включая SQL Server, Oracle, MySQL и SQLite;
* поддержка миграции баз данных: Entity Framework предоставляет инструменты для управления изменениями схемы базы данных, включая создание, обновление и удаление таблиц и столбцов базы данных;
* поддержка кэширования: Entity Framework включает механизм кэширования, который может повысить производительность приложения за счет уменьшения количества запросов к базе данных;
* интеграция с другими технологиями .NET: Entity Framework интегрируется с другими технологиями .NET, такими как LINQ, WCF и ASP.NET.

Entity Framework имеет несколько компонентов, которые работают вместе для обеспечения его функциональности. К ним относятся:

* основной класс «ObjectContext» , представляющий контекст базы данных. Он управляет соединением с базой данных и предоставляет методы для запросов и обновления данных;
* набор классов «Entity Data Model», определяющих структуру базы данных и отображающих ее на объекты .NET. EDM включает три основных компонента: Тип сущности, Ассоциация и Свойство;
* интегрированный «LINQ toEntities» в язык язык запросов, который позволяет разработчикам запрашивать базу данных, используя объекты .NET и синтаксис LINQ;
* подход к разработке «Code First», который позволяет разработчикам создавать таблицы и столбцы базы данных непосредственно из классов и атрибутов .NET.

В целом, Entity Framework предоставляет мощный и гибкий набор инструментов для работы с реляционными базами данных в приложениях .NET. Его абстракция базы данных, автоматическая генерация SQL и поддержка нескольких баз данных делают его популярным выбором для создания приложений, управляемых данными.

Draw.io – инструмент, который позволяет создавать блок–схемы, сетевые диаграммы, интеллект–карты, отношения сущностей, программные блоки, UML, макеты и т.д []. Богатая функциональность Draw.io позволяет пользователям отслеживать и восстанавливать изменения, импортировать и экспортировать в PDF, PNG, XML, VSDX, HTML, а также автоматически публиковать и делиться работами.

С помощью сервиса можно создавать:

* графики;
* блок–схемы;
* диаграммы;
* ментальные карты;
* макеты.

Особенности сервиса:

* большой набор шаблонов;
* ментальные карты;
* быстрое создание блоков;
* интуитивно понятный интерфейс;
* экспорты в форматы JPG, PNG, SVG, VDSX;
* совместная работа;
* русскоязычный интерфейс;
* возможность скачать в виде программы для компьютера.
  1. Порядок авторизации пользователей

Аутентификация и авторизация - важнейшие аспекты любого программного инструмента, работающего с конфиденциальной информацией, такой как результаты тестирования. В этом разделе будут определены порядок авторизации пользователей в программном инструменте для тестирования знаний участников олимпиад.

Создание ролей. Первым шагом является создание необходимых ролей. В программе сущестует две роли – «Преподаватель» и «Учащийся». Роль «Преподаватель» имеет больше привилегий, чем роль «Учащийся». Поэтому очень важно точно определить характеристики ролей.

Предоставление ролям необходимых привилегий: Следующий шаг - наделение ролей необходимыми привилегиями. В программе роль «Учащийся» может создавать и настраивать викториныс вопросами, распределять викторины работы по категориям, добавлять учащихся в группы, назначать викторины и просматривать результаты учащихся. С другой стороны, роль «Учащийся» может только принимать тесты и просматривать результаты.

Пользователи с любой ролью должны сначала зарегистрироваться, введя свою электронную почту, имя, фамилию и пароль. После регистрации они могут войти в приложение, используя свою электронную почту и пароль.

Учитель может добавлять учеников и распределять их по группам. Кроме того, учитель может назначать викторины группам или отдельным ученикам. Ученик может принимать участие только в назначенных ему тестах и просматривать свои результаты.

Следует отметить, что четко определенная процедура аутентификации и авторизации имеет решающее значение для безопасного функционирования любого программного инструмента, работающего с конфиденциальной информацией. Создавая роли, предоставляя им необходимые привилегии и назначая роли пользователям, мы можем гарантировать, что пользователи смогут получить доступ только к той информации, которая относится к их ролям.

* 1. Организация данных

В качестве модели данных для хранения используется  реляционная система управления базами данных MySql. Так как приложение написано в интегрированной среде разработки Visual Studio, то для работы с базой данных необходимо в проект добавить Pomelo.EntityFrameworkCore.MySqlNuGet пакет. После чего в коде нужно добавить все необходимые сущности и связи между ними. В дальнейшем, для подключения базы необходимо добавить сервис для работы для подключения к данной базе данных.

В таблице «aspnetusers» описана информация о пользователях, структура которой представлена в таблице 3.1.

Таблица 3.1 – Структура таблицы «aspnetusers»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер поля, байт | Описание |
| Id | VARCHAR | 255 | Уникальный идентификатор пользователя |
| TeacherId | VARCHAR | 255 | Уникальный идентификатор преподавателя из таблицы «aspnetusers» |
| FirstName | LONGTEXT | 4 | Имя пользователя |
| LastName | LONGTEXT | 4 | Имя пользователя |
| Email | VARCHAR | 255 | Адрес электронной почты пользователя |
| CreatedOn | DATETIME | 8 | Дата и время создания пользователя |
| ModifiedOn | DATETIME | 8 | Дата и время редактирования информации о пользователе |
| IsDeleted | TINYINT | 1 | Был ли удален пользователь |
| DeletedOn | DATETIME | 8 | Дата и время удаления пользователя |

Таблица 3.2 – Структура таблицы «questions»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер поля, байт | Описание |
| 1 | 2 | 3 | 4 |
| Id | VARCHAR | 255 | Уникальный идентификатор для пользователя |
| QuizId | VARCHAR | 255 | Уникальный идентификатор для викторины |
| Text | LONGTEXT | 4 | Содержание вопроса |
| Number | INT | 4 | Порядковый номер вопроса |

|  |  |  |  |
| --- | --- | --- | --- |
| Продолжение таблицы 3.2 | | | |
| 1 | 2 | 3 | 4 |
| CreatedOn | DATETIME | 8 | Дата и время создания вопроса |
| ModifiedOn | DATETIME | 8 | Дата и время редактирования вопроса |
| IsDeleted | TINYINT | 1 | Был ли удален вопрос |
| DeletedOn | DATETIME | 8 | Дата и время удаления вопроса |

Таблица 3.3 – Структура таблицы «answers»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер поля, байт | Описание |
| Id | VARCHAR | 255 | Уникальный идентификатор для пользователя |
| QuestionId | VARCHAR | 255 | Уникальный идентификатор для викторины |
| Text | LONGTEXT | 4 | Содержание вопроса |
| IsRightAnswer | TINYINT | 1 | Флаг для обозначения верного ответа |
| CreatedOn | DATETIME | 8 | Дата и время создания ответа |
| ModifiedOn | DATETIME | 8 | Дата и время редактирования ответа |
| IsDeleted | TINYINT | 1 | Был ли удален ответ |
| DeletedOn | DATETIME | 8 | Дата и время удаления ответа |

Таблица 3.4 – Структура таблицы «quizzes»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер поля, байт | Описание |
| Id | VARCHAR | 255 | Уникальный идентификатор для викторины |
| CreatorId | VARCHAR | 255 | Уникальный идентификатор для создателя |
| CaregoryId | VARCHAR | 255 | Уникальный идентификатор для категории |
| EventId | VARCHAR | 255 | Уникальный идентификатор для события |
| PasswordId | INT | 4 | Уникальный идентификатор для пароля |
| Name | VARCHAR | 255 | Название викторины |
| Description | LONGTEXT | 4 | Описание викторины |
| Timer | INT | 4 | Продолжительность викторины |
| CreatedOn | DATETIME | 8 | Дата и время создания ответа |
| ModifiedOn | DATETIME | 8 | Дата и время редактирования ответа |
| IsDeleted | TINYINT | 1 | Был ли удален ответ |
| DeletedOn | DATETIME | 8 | Дата и время удаления ответа |

Физическая схема базы данных представена на рисунке Б.16 в приложении Б.

* 1. Функции: логическая и физическая организация

Функция регистрации пользователя. Функция принимает объект ApplicationUser и пароль в качестве входных данных, устанавливает имя пользователя на его адрес электронной почты и создает пользователя. Объект userManager отвечает за фактическое создание пользователя в базе данных. Метод возвращает булево значение, указывающее, было ли создание пользователя успешным или нет. В общем, эта функция используется для регистрации нового пользователя с использованием фреймворка Identity.

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

{

newUser.UserName = newUser.Email;

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

return result.Succeeded;

}

Функция отправки токена для сброса пароля. Функция используется для отправки сообщения электронной почты для сброса пароля пользователю с указанным адресом электронной почты. Сначала она извлекает пользователя из базы данных с помощью метода FindByEmailAsync объекта userManager. Если пользователь существует, он генерирует уникальный токен сброса пароля для пользователя с помощью метода GeneratePasswordResetTokenAsync. Затем с помощью метода SendEmailAsync объекта emailSender отправляется письмо с маркером сброса на адрес электронной почты пользователя. Если письмо отправлено успешно, функция возвращает токен сброса. В противном случае возвращается null. В общем, эта функция позволяет пользователям сбросить свой пароль, отправив им электронное письмо с маркером сброса пароля.

public async Task<string> SendPasswordResetEmail(string email)

{

var user = await userManager.FindByEmailAsync(email);

if (user == null)

{

return null;

}

var resetToken = await userManager.GeneratePasswordResetTokenAsync(user);

var sendEmailResponse = await emailSender.SendEmailAsync(email, "Password reset token", resetToken);

if (!sendEmailResponse.IsSuccessStatusCode)

{

return null;

}

return resetToken;

}

Функция для отправки электронного письма с помощью почтовой службы SendGrid. Она принимает три параметра - адрес электронной почты получателя, тему письма и тело письма. Она создает новый объект SendGridMessage со свойствами From, Subject и PlainTextContent, установленными в указанные значения. Затем он добавляет адрес электронной почты получателя в сообщение с помощью метода AddTo. Наконец, она отправляет письмо с помощью почтовой службы SendGrid через объект sendGridClient и возвращает ответ службы. В целом, эта функция позволяет отправлять электронные письма с помощью SendGrid.

public async Task<Response> SendEmailAsync(string toEmail, string subject, string body)

{

var message = new SendGridMessage()

{

From = new EmailAddress(configuration.GetValue<string>("EmailSender:FromEmail")),

Subject = subject,

PlainTextContent = body

};

message.AddTo(new EmailAddress(toEmail));

return await sendGridClient.SendEmailAsync(message);

}

}

* 1. Входные и выходные данные

Данные, вводимые пользователем, включают личную информацию, такую как имя, фамилия, электронная почта, пароль. Входные данные и их описание представлены в таблице 3.5.

Таблица 3.5 – Описание входных данных

|  |  |
| --- | --- |
| Вид данных | Описание |
| FirstName | Имя пользователя |
| Вид данных | Описание |
| LastName | Фамилия пользователя |
| Еmai | Электронный адрес |
| Password | Пароль для аккаунта |

В дополнение к входным данным программа генерирует данные, которые используются для создания отчета по викторине. Выходные данные и их описание представлены в таблице 3.6, шаблон выходного документа представлен на рисунке Б.1 в приложении Б.

Таблица 3.6 – Описание выходных данных

|  |  |
| --- | --- |
| Вид данных | Описание |
| QuizName | Название викторины |
| Date | Дата прохождения |
| Question1 | Первый вопрос |
| Question2 | Второй вопрос |
| Question3 | Третий вопрос |

* 1. Функциональное тестирование

Данное тестирование проводится для выявления неполадок и недочетов программы на этапе ее сдачи в эксплуатацию.

Функциональное тестирование предполагает проверку выполнения определенных на этапе проектирования функций.

Начальное окно, оно же окно авторизации, представлено на рисунке Б.2 в приложении Б.

Протестируем функцию авторизации. Для этого, находясь в данном окне, введем логин и пароль, после чего нажмем на кнопку «Войти».

Тест кейсы для функции авторизации представлены в таблице 3.7.

Таблица 3.7 – Тест-кейсы для функции «Авторизация»

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модуль/функция | Шаги воспроизведения | Результат |
| 1 | Авторизация | 1. Заполнить поля данными:   * Email: «invalidMail123» * Пароль: 1234567 | Ожидаемый результат: уведомление о неверно введенных данных пользователя. |

Продолжение таблицы 3.7

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модуль/функция | Шаги воспроизведения | Результат |
|  |  |  | Фактический результат соответствует ожидаемому и представлен на рисунке Б.3 в приложении Б. |
| 22 | Авторизация | 1. Заполнить поля данными:   * Email: «matveyrfh@mail.ru» * Пароль: «fuieonbfwhi»   2. Нажать на кнопку «Войти» | Ожидаемый результат: уведомление о неверно введенных данных пользователя. |
| Фактический результат соответствует ожидаемому и представлен на рисунке Б.4 в приложении Б. |
| 23 | Авторизация | 1. Заполнить поля данными:   * Email: «matveyrfh@mail.ru» * Пароль: «matvey2003»   2. Нажать на кнопку «Войти» | Ожидаемый результат: уведомление об успешной авторизации. |
| Окно с правильно введенными данными представлено на рисунке Б.5 в приложении Б. Фактический результат соответствует ожидаемому и представлен на рисунке Б.6 в приложении Б. |

Протестируем функцию изменения пароля учетной записи. Для этого, находясь в окне авторизации, необходимо нажать на надпись «Забыльпароль», расположенную в нижней части экрана. После чего откроется окно изменения пароля учетной записи, представленное на рисунке Б.7 в приложении Б.

Тест-кейсы для функции изменения данных учетной записи представлены в таблице 3.8.

Таблица 3.8 – Тест-кейсы для функции «Изменение данных учетной записи»

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модуль/функция | Шаги воспроизведения | Результат |
| 4 | Изменение пароля учетной записи | 1. Заполнить полеEmail:   * [matveyrfh@mail.ru](mailto:matveyrfh@mail.ru)   2. Нажать на кнопку «Отправить»  3. Заполнить поле Токен:   * 563111   4. Нажать на кнопку «Подтвердить» | Ожидаемый результат: уведомление о неверно введенных данных. |
| Фактический результат соответствует ожидаемому и представлен на рисунке Б.8 в приложении Б. |
| 25 | Изменение пароля учетной записи | 1. Заполнить поле Email:   * [matveyrfh@mail.ru](mailto:matveyrfh@mail.ru)   2. Нажать на кнопку «Отправить» | Ожидаемый результат: уведомление об успешной отправке токена на почту. |
| Фактический результат соответствует ожидаемому и представлен на рисунке Б.9 в приложении Б. |
| 6 | Изменение пароля учетной записи | 1. Заполнить поле Email:   * [matveyrfh@mail.ru](mailto:matveyrfh@mail.ru)   2. Нажать на кнопку «Отправить»  3. Заполнить поле Токен:   * 602356   4. Нажать на кнопку «Подтвердить»  5. Заполнить поле «Пароль»   * Matvey2003   6. Нажать на кнопку «Поменять» | Ожидаемый результат: уведомление об успешной смене пароля. |
| Фактический результат соответствует ожидаемому и представлен на рисунке Б.10 в приложении Б. |

Протестируем функцию регистрацииучащегося. Для этого, находясь в окне авторизации, необходимо нажать на надпись «Регистрация учащегося», расположенную в нижней части экрана. После чего откроется окно регистрации, представленное на рисунке Б.11. в приложении Б.

Таблица 3.9 – Тест-кейсы для функции «Регистрация учащегося»

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модуль/функция | Шаги воспроизведения | Результат |
| 7 | Регистрация учащегося | 1. Заполнить поле Email:   * [matveyrfh@mail.ru](mailto:matveyrfh@mail.ru)   2. Заполнить поле Пароль:   * dfg34dg   3. Заполнить поле Повторить пароль:   * dfg34dg | Ожидаемый результат: уведомление о не введенных данных пользователя. |
| Фактический результат соответствует ожидаемому и представлен на рисунке Б.12 в приложении Б. |
| 28 | Регистрация учащегося | 1. Заполнить поле Email:   * [matveyrfh@mail.ru](mailto:matveyrfh@mail.ru)   2. Заполнить поле Имя:   * Матвей   3. Заполнить поле Фамилия:   * Быковский   4. Заполнить поле Пароль:   * 123456   5. Заполнить поле Повторить пароль:   * 123456   6. Нажать на кнопку «Регистрация» | Ожидаемый результат: уведомление об уже зарегистрированном почтовом адресе. |
| Фактический результат соответствует ожидаемому и представлен на рисунке Б.13 в приложении Б. |
| 9 | Регистрация учащегося | 1. Заполнить поле Email:   * [new6@mail.ru](mailto:new6@mail.ru)   2. Заполнить поле Имя:   * Новый   3. Заполнить поле Фамилия:   * Пользователь   4. Заполнить поле Пароль:   * 123456   5. Заполнить поле Повторить пароль:   * 123456   6. Нажать на кнопку «Регистрация» | Ожидаемый результат: уведомление об успешной регистрации нового пользователя. |
| Фактический результат соответствует ожидаемому и представлен на рисунке Б.14 в приложении Б. |

* 1. Описание справочной системы

Справочная система для приложения «QuizHut» - это руководство пользователя, которое содержит пошаговые инструкции по использованию различных функций приложения. Она предназначена для того, чтобы помочь пользователям разобраться в функциональных возможностях приложения и понять, как в нем ориентироваться.

На данный момент справочная система охватывает три функции приложения:

* регистрация;
* восстановление пароля;
* авторизация.

Каждая функция подробно объясняется с помощью четких и кратких инструкций, чтобы пользователи могли легко понять, как их использовать. Справочная система может также включать скриншоты и видео, чтобы помочь пользователям визуализировать различные шаги, связанные с использованием приложения.

Цель справочной системы - предоставить пользователям необходимые знания и навыки для эффективного управления отелями с помощью приложения. Справочная система представлена на рисунке Б.15 в приложении Б.

1. Применение
   1. Назначение программного средства

Программа «QuizHut.exe» предназначена для автоматизации сбора и обработки данных при подготовке и проведении академических олимпиад. Программа предназначена для использования организаторами академических олимпиад - школами, колледжами и университетами.

Основная цель программного обеспечения - оптимизировать процесс подготовки и проведения олимпиад путем автоматизации нескольких бизнес-процессов, включая регистрацию, создание заданий, оценку заданий и формирование отчетов. Это поможет организаторам управлять всем процессом более эффективно и точно, снижая риск ошибок и экономя время.

Область применения программного обеспечения - академические олимпиады в разных направлениях. Она предназначена для использования организаторами этих мероприятий для управления всем процессом, начиная с регистрации и заканчивая формированием итогового отчета.

Программное обеспечение призвано решить несколько проблем, связанных с ручным процессом подготовки и проведения олимпиад. Эти проблемы включают в себя трудоемкий сбор и обработку данных, сложное создание заданий, а также субъективное и подверженное ошибкам оценивание заданий.

Ограничения сферы применения программного обеспечения заключаются в том, что оно разработано специально для академических олимпиад и может не подходить для других типов мероприятий или соревнований. Оно также ограничено требованиями к аппаратному и программному обеспечению, необходимым для его правильного функционирования, а также любыми техническими ограничениями, которые могут возникнуть в процессе разработки программного обеспечения.

* 1. Условия применения

Минимальные системные требования для оптимальной работы программного средства является персональный компьютер (ПК) со следующими характеристиками:

* 32 – разрядный (x86) или 64 – разрядный (x64) процессор с тактовой частотой 1 ГГц или выше;
* 1 ГБ (для 32 – разрядного процессора) или 2 ГБ (для 64 – разрядного процессора) ОЗУ;
* 16 ГБ (для 32 – разрядной системы) или 20 ГБ (для 64 – разрядной системы) свободного места на жестком диске;
* графическое устройство DirectX 9 с драйвером WDDM 1.0 или более поздней версии.

Для использования программы необходим компьютер под управлением операционной системы Windows 10. Программное средство «QuizHut.exe» предназначено для работы на этой операционной системе.

Пользователю необходимо запустить установочный файл и запустить мастер установки. Мастер проведет пользователя через весь процесс установки и предложит выбрать папку назначения для установки.

После завершения установки пользователь может запустить программу, нажав на иконку. Программа откроется и предложит пользователю авторизоваться в приложении. После этого пользователь может начать использовать программу.

1. Охрана труда и окружающей среды

Статическое электричество часто рассматривается как скрытый враг специалистов по охране труда, технике безопасности и обычных работников особенно в информационной и технологической промышленности.Будучи невидимой силой, она часто не идентифицируется как проблема до тех пор, пока не произойдет инцидент. Однако последствия могут быть серьезными как для персонала, так и для компании. Специалисты-практики должны защитить благополучие своих сотрудников от возможности поражения электрическим током и последующих травм, а также обеспечить надлежащую защиту рабочей среды от риска пожара и взрыва, вызванного электростатическим разрядом.

В 600 году до н.э. философ и математик Фалес Милетский впервые описал, как он натирал янтарем шерсть кошки, после чего она могла собирать перья. Это первый известный рассказ о природной силе статического электричества, и было введено слово электростатика, означающее "электричество в состоянии покоя".Когда материал имеет чистый электрический заряд, положительный или отрицательный, говорят, что он имеет статический заряд. Во многих случаях этот заряд медленно уменьшается со временем - фактическая продолжительность зависит от сопротивления материала. Для практических целей можно привести два крайних примера: пластик и металл. Пластмассы обычно обладают высоким сопротивлением, что позволяет им сохранять статический заряд в течение длительного времени. Металлы, с другой стороны, имеют низкое сопротивление, что означает, что заземленный металлический предмет разряжается практически мгновенно.

Примерно 50 лет назад статическое электричество не было серьезной проблемой в промышленных процессах, поскольку многие из ключевых элементов, влияющих на создание и поддержание статического заряда - такие как тип материала, температура окружающей среды, влажность и повторяющиеся действия, такие как трение или разделение - не присутствовали в производственной среде. Например, на заре производства текстиля необработанные натуральные материалы - шерсть и хлопок - впитывали воду из атмосферы и поэтому имели очень низкий статический заряд. Кроме того, старая фабричная среда с ее паровыми трубами, обеспечивающими тепло и влажность, означала, что вероятность образования статического заряда была низкой.Однако за последние 50 лет сырье и производственные процессы существенно изменились, как и фабрика. Во-первых, искусственные материалы, такие как нейлон, полиэстер и различные пластмассы, теперь доминируют и более доступны, чем натуральные материалы.Кроме того, повсеместное внедрение систем кондиционирования воздуха и сухого воздушного отопления привело к созданию более сухой атмосферы и снижению уровня влажности, что выдвинуло на первый план проблемы статики. Промышленные процессы в большей или меньшей степени страдают от проблем, связанных со статическим электричеством: загрязнение продукции, низкая скорость работы оборудования, поражение операторов током, риск пожара или взрыва.

Многие люди испытывали неприятное ощущение "удара током". Такие повседневные действия, как закрытие двери автомобиля, передвижение тележки по супермаркету или закрытие посудомоечной машины, могут привести к напряжению более 10 000 вольт. Однако, поскольку ток протекает в течение короткого времени, серьезные повреждения случаются редко.

В заводских условиях оператор оборудования может получить удар током непосредственно от продукта, с которым он работает, или зарядиться за счет индукции, находясь в электрическом поле заряженного объекта - например, во время работы за персональным компьютером, рядом электрическими кабелямиили полотном в процессе размотки и перемотки. Заряд накапливается до тех пор, пока оператор не коснется заземленной части машины, а затем разряжается, что приводит к поражению током. Опять же, это вряд ли приведет к серьезному ущербу, но если удар достаточно силен, чтобы вызвать реакцию отдачи, то могут произойти несчастные случаи, когда оператор столкнется с коллегой, с рамой машины или наткнется на другие компоненты, что приведет к травме.

Устранение риска статического удара важно для того, чтобы избежать травм и проявить должную заботу и внимание к благополучию персонала. Однако есть и другие важные аспекты. Во-первых, если кто-то получает удар током, необходимо остановить машину для выяснения причины. Это приведет к простою и потере производства. Если удар был достаточно сильным, чтобы пострадавшему потребовалась медицинская помощь, то станок нельзя запускать до тех пор, пока его не проверят специалисты по охране труда и технике безопасности.Если такие потрясения происходят с операторами регулярно, это может еще больше сказаться на производстве, поскольку люди будут интуитивно работать медленнее, если будут считать, что им грозит потрясение. Никто не любит испытывать боль, и постоянная угроза потрясений приведет к неудовлетворенности и недовольству персонала. Кроме того, в сегодняшнюю эпоху судебных разбирательств неспособность компании "обеспечить безопасные условия труда" может привести к судебному иску.

Одним из самых первых способов контроля статического заряда в промышленности было использование "пассивной" ионизации в виде проводящих материалов, таких как щетки из углеродного волокна. Этот метод снижал уровень статического заряда, но не снимал его полностью.Контроль статического электричества сделал большой шаг вперед в 1960-х годах с появлением "активных" систем электрической ионизации. В этих системах обычное сетевое напряжение переменного тока повышалось до 5 кВ через специальный трансформатор и подавалось на острые контакты эмиттера, где высокая энергия генерировала большое количество отрицательных и положительных ионов. Статически заряженная поверхность любой полярности, проходящая вблизи этого облака ионов, затем нейтрализуется. Такие системы работали хорошо, но были ограничены в плане рабочего расстояния (обычно 20 мм). Кроме того, они имели относительно короткий срок службы и вызывали поражение электрическим током при случайном прикосновении к контактам. Постепенное совершенствование этой технологии переменного тока улучшило характеристики системы.

Сравнительно недавнее внедрение импульсной технологии постоянного тока для статического контроля не только стало эффективной альтернативой системам переменного тока, но и позволило оптимизировать решения для специфических материалов и более сложных процессов.

Для контроля статического электричества на рабочем месте имеется широкий спектр специального оборудования. Токопроводящие браслеты и напольные коврики хорошо работают во многих областях, но непрактичны в условиях, когда работникам приходится много передвигаться.

Ионизационные воздуходувки широко используются в технологической промышленности для предотвращения поражения электрическим током. Они работают благодаря сочетанию ионизирующей головки со встроенной системой вентиляторов и могут генерировать мощный поток ионизированного воздуха в широкой рабочей зоне, обеспечивая комфорт и отсутствие статического электричества.

Ионизаторы могут устанавливаться непосредственно на оборудование для контроля статического электричества и защиты работников. Они различаются по производительности и диапазону ионизации в зависимости от конкретного применения. Специалисты по статическому контролю смогут посоветовать оптимальное решение и расположение ионизирующей планки для достижения наилучших результатов.

Монтаж ионизирующего оборудования во взрывоопасных средах требует особого внимания. Это особенно важно в заводских условиях, где растворители используются как часть технологического процесса. Например, в полиграфической промышленности краски на основе растворителей являются потенциальным источником пожара, а высокоскоростные рулонные печатные машины заряжаются при прохождении через последовательные печатные головки, что приводит к опасности статического разряда, воспламеняющего легковоспламеняющиеся растворители и пары. В этих обстоятельствах компании должны убедиться, что их решения по статическому контролю сертифицированы уполномоченным испытательным органом.

Термоформовка - еще один производственный процесс, в котором существует повышенный риск того, что неконтролируемое статическое электричество приведет к потенциально смертельному инциденту. Присутствуют легковоспламеняющиеся растворители и газы, что удваивает риск пожара, который может быть легко вызван искрой, возникшей в результате статического разряда.

Ввиду высоких затрат, которые могут быть понесены в результате пожаров или несчастных случаев, связанных со статическим электричеством, применяемые системы контроля статического электричества должны быть наиболее эффективными. На этапе экструзии процесса термоформования безопасность может быть еще больше повышена благодаря новой разработке, которая автоматически контролирует и оптимизирует снятие статического заряда.

Поэтому заказчикам необходимо получить рекомендации от поставщиков статического оборудования, чтобы убедиться, что они могут провести различие между продуктами, которые просто сообщают о статике, и системами, которые могут оптимизировать снятие статического заряда путем определения остаточного заряда на целевом материале и реагирования на него. Это означает, что система статического контроля может работать с оптимальной производительностью в различных условиях без необходимости вмешательства оператора.

Поскольку производственные процессы продолжают развиваться, связанные с ними эффекты статического заряда будут представлять все большие проблемы. Если раньше основным направлением статического контроля было повышение производительности и рентабельности промышленности за счет снижения накопления статического заряда на обрабатываемых материалах, то теперь в промышленности более открыто обсуждаются вопросы охраны здоровья и безопасности. Это хорошие признаки, поскольку компании, занимающиеся статическим контролем, теперь работают вместе с профессионалами в области охраны труда и техники безопасности, чтобы помочь снизить риск опасности от статических разрядов и предотвратить поражение операторов статическим электричеством.

# **Заключение**

Во время прохождения производственной преддипломной была начата работа над проектом «Средство по автоматизации сбора и обработки данных для подготовки и проведения олимпиады». При прохождении практики, для разработки проекта использовалась среда разработки Visual Studio 2022. Был применен язык программирования C# 10 вместе с платформой .NET 7.

Были получены как теоретические так и практические знания в области разработки WPF приложений. Освоение нового паттерна проектирования и архитектуры построения приложений.

Cледует отметить, что цель разработки декстоп-приложения по автоматизации сбора и обработки данных для подготовки и проведения олимпиады была успешно достигнута. Проект был реализован с использованием C# и Visual Studio, с применением трехуровневой архитектуры, которая позволяет четко разделить обязанности и упростить обслуживание. В проекте использовались слой данных, слой бизнес-логики и слой графического интерфейса. Выбор «MySql» в качестве базы данных в сочетании с использованием Entity Framework обеспечил эффективное и безопасное управление данными о прокате автомобилей. Такая установка также облегчила расширение и масштабирование базы данных в будущем при необходимости. Кроме того, реализация паттернов Repository и Unit of Work в слое данных еще больше улучшила организацию и сопровождаемость кода. Эти архитектурные паттерны широко приняты и используются при разработке программного обеспечения, они облегчают сопровождение и расширение кодовой базы. Разработанные сервисы и «вью-модели» позволяют обрабатывать и реагировать на ввод пользователя. В целом, проект можно считать успешным, поскольку в ходе его разработки не возникло никаких трудностей. Поэтому данное приложение можно использовать в качестве эффективного и надежного решения по автоматизации сбора и обработки данных для подготовки и проведения олимпиады.

# Список информационных источников

1. Багласова, Т.Г. Методические указания по выполнению дипломного проекта для учащихся по специальности 2–40 01 01 «Программное обеспечение технологий» / Т.Г. Багласова. – Минск : КБП, 2021. – 30 c.
2. Багласова, Т.Г. Методические указания по оформлению курсовых и дипломных проектов / Т.Г. Багласова, К.О. Якимович. – Минск : КБП, 2022. – 29 c.
3. Орлов, С. А. Технологии разработки программного обеспечения: Учебник для вузов. 4–е изд. / С. А. Орлов, Б. Я. Цилькер. – СПб. : Питер, 2012. – 608 с.
4. Михнюк, Т.Ф. Охрана труда / Т.Ф. Михнюк. – Минск : ИВЦ Минфина, 2009. – 365 с.
5. Общие требования к тестовым документам : ГОСТ 2.105–95. – Введ. 01.01.1996. – Минск :Межгос. совет по стандартизации, метрологии и сертификации, 1995. – 84 с.
6. 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.
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. Htmlhelpworkshop [Электронный ресурс]. – Режим доступа:http://word2003.ru/8/8–opisanie–vozmozhnostej–html–help–workshop.htm. – Дата доступа : 22.04.2023.
9. User documentationArchives - draw.io [Электронный ресурс]. – Режим доступа : https://drawio-app.com/tag/user-documentation/. – Дата доступа : 19.04.2023.
10. Windows Presentation Foundation for .NET 7 documentation [Электронныйресурс]. – Режимдоступа: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/?view=netdesktop-7.0– датадоступа 18.04.2023.
11. Общие сведения о VisualStudio [Электронный ресурс] / Microsoft, Документация по VisualStudio. – Режим доступа :https://docs.microsoft.com/ru–ru/visualstudio/ide/visual–studio–ide. – Дата доступа 15.04.2023.
12. Руководство по программированию на C# [Электронный ресурс]. – Microsoft, 2023. – Режим доступа : http://docs.microsoft.com/ru–ru/dotnet/csharp/programming–guide/. – Дата доступа : 09.04.2023.

# **Приложение А (обязательное)**

**Текст программных модулей**

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

{

private static readonly MethodInfo SetIsDeletedQueryFilterMethod =

typeof(ApplicationDbContext).GetMethod(

nameof(SetIsDeletedQueryFilter),

BindingFlags.NonPublic | BindingFlags.Static);

public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)

: base(options)

{

}

public DbSet<Answer> Answers { get; set; }

public DbSet<Question> Questions { get; set; }

public DbSet<Quiz> Quizzes { get; set; }

public DbSet<Password> Passwords { get; set; }

public DbSet<Category> Categories { get; set; }

public DbSet<Group> Groups { get; set; }

public DbSet<Event> Events { get; set; }

public DbSet<Result> Results { get; set; }

public DbSet<StudentGroup> StudentsGroups { get; set; }

public DbSet<EventGroup> EventsGroups { get; set; }

public DbSet<ScheduledJob> ScheduledJobs { get; set; }

public override int SaveChanges() => this.SaveChanges(true);

public override int SaveChanges(bool acceptAllChangesOnSuccess)

{

this.ApplyAuditInfoRules();

return base.SaveChanges(acceptAllChangesOnSuccess);

}

public override Task<int> SaveChangesAsync(CancellationToken cancellationToken = default) =>

this.SaveChangesAsync(true, cancellationToken);

public override Task<int> SaveChangesAsync(

bool acceptAllChangesOnSuccess,

CancellationToken cancellationToken = default)

{

this.ApplyAuditInfoRules();

return base.SaveChangesAsync(acceptAllChangesOnSuccess, cancellationToken);

}

protected override void OnModelCreating(ModelBuilder builder)

{

base.OnModelCreating(builder);

builder.ApplyConfigurationsFromAssembly(typeof(ApplicationDbContext).Assembly);

EntityIndexesConfiguration.Configure(builder);

var entityTypes = builder.Model.GetEntityTypes().ToList();

var deletableEntityTypes = entityTypes

.Where(et => et.ClrType != null && typeof(IDeletableEntity).IsAssignableFrom(et.ClrType));

foreach (var deletableEntityType in deletableEntityTypes)

{

var method = SetIsDeletedQueryFilterMethod.MakeGenericMethod(deletableEntityType.ClrType);

method.Invoke(null, new object[] { builder });

}

var foreignKeys = entityTypes

.SelectMany(e => e.GetForeignKeys().Where(f => f.DeleteBehavior == DeleteBehavior.Cascade));

foreach (var foreignKey in foreignKeys)

{

foreignKey.DeleteBehavior = DeleteBehavior.Restrict;

}

}

private static void SetIsDeletedQueryFilter<T>(ModelBuilder builder)

where T : class, IDeletableEntity

{

builder.Entity<T>().HasQueryFilter(e => !e.IsDeleted);

}

private void ApplyAuditInfoRules()

{

var changedEntries = this.ChangeTracker

.Entries()

.Where(e =>

e.Entity is IAuditInfo &&

(e.State == EntityState.Added || e.State == EntityState.Modified));

foreach (var entry in changedEntries)

{

var entity = (IAuditInfo)entry.Entity;

if (entry.State == EntityState.Added && entity.CreatedOn == default)

{

entity.CreatedOn = DateTime.UtcNow;

}

else

{

entity.ModifiedOn = DateTime.UtcNow;

}

}

}

}

public class DbQueryRunner : IDbQueryRunner

{

public DbQueryRunner(ApplicationDbContext context)

{

this.Context = context ?? throw new ArgumentNullException(nameof(context));

}

public ApplicationDbContext Context { get; set; }

public Task RunQueryAsync(string query, params object[] parameters)

{

return this.Context.Database.ExecuteSqlRawAsync(query, parameters);

}

public void Dispose()

{

this.Dispose(true);

GC.SuppressFinalize(this);

}

protected virtual void Dispose(bool disposing)

{

if (disposing)

{

this.Context?.Dispose();

}

}

}

internal static class EfExpressionHelper

{

private static readonly Type StringType = typeof(string);

private static readonly MethodInfo ValueBufferGetValueMethod =

typeof(ValueBuffer).GetRuntimeProperties().Single(p => p.GetIndexParameters().Any()).GetMethod;

private static readonly MethodInfo EfPropertyMethod =

typeof(EF).GetTypeInfo().GetDeclaredMethod(nameof(Property));

public static Expression<Func<TEntity, bool>> BuildByIdPredicate<TEntity>(

DbContext dbContext,

object[] id)

where TEntity : class

{

if (id == null)

{

throw new ArgumentNullException(nameof(id));

}

var entityType = typeof(TEntity);

var entityParameter = Expression.Parameter(entityType, "e");

var keyProperties = dbContext.Model.FindEntityType(entityType).FindPrimaryKey().Properties;

var predicate = BuildPredicate(keyProperties, new ValueBuffer(id), entityParameter);

return Expression.Lambda<Func<TEntity, bool>>(predicate, entityParameter);

}

private static BinaryExpression BuildPredicate(

IReadOnlyList<IProperty> keyProperties,

ValueBuffer keyValues,

ParameterExpression entityParameter)

{

var keyValuesConstant = Expression.Constant(keyValues);

BinaryExpression predicate = null;

for (var i = 0; i < keyProperties.Count; i++)

{

var property = keyProperties[i];

var equalsExpression = Expression.Equal(

Expression.Call(

EfPropertyMethod.MakeGenericMethod(property.ClrType),

entityParameter,

Expression.Constant(property.Name, StringType)),

Expression.Convert(

Expression.Call(keyValuesConstant, ValueBufferGetValueMethod, Expression.Constant(i)),

property.ClrType));

predicate = predicate == null ? equalsExpression : Expression.AndAlso(predicate, equalsExpression);

}

return predicate;

}

}

internal static class EntityIndexesConfiguration

{

public static void Configure(ModelBuilder modelBuilder)

{

var deletableEntityTypes = modelBuilder.Model

.GetEntityTypes()

.Where(et => et.ClrType != null && typeof(IDeletableEntity).IsAssignableFrom(et.ClrType));

foreach (var deletableEntityType in deletableEntityTypes)

{ modelBuilder.Entity(deletableEntityType.ClrType).HasIndex(nameof(IDeletableEntity.IsDeleted));

}

}

}

public class EfDeletableEntityRepository<TEntity> : EfRepository<TEntity>, IDeletableEntityRepository<TEntity>

where TEntity : class, IDeletableEntity

{

public EfDeletableEntityRepository(ApplicationDbContext context)

: base(context)

{

}

public override IQueryable<TEntity> All() => base.All().Where(x => !x.IsDeleted);

public override IQueryable<TEntity> AllAsNoTracking() => base.AllAsNoTracking().Where(x => !x.IsDeleted);

public IQueryable<TEntity> AllWithDeleted() => base.All().IgnoreQueryFilters();

public IQueryable<TEntity> AllAsNoTrackingWithDeleted() => base.AllAsNoTracking().IgnoreQueryFilters();

public Task<TEntity> GetByIdWithDeletedAsync(params object[] id)

{

var getByIdPredicate = EfExpressionHelper.BuildByIdPredicate<TEntity>(this.Context, id);

return this.AllWithDeleted().FirstOrDefaultAsync(getByIdPredicate);

}

public void HardDelete(TEntity entity) => base.Delete(entity);

public void Undelete(TEntity entity)

{

entity.IsDeleted = false;

entity.DeletedOn = null;

this.Update(entity);

}

public override void Delete(TEntity entity)

{

entity.IsDeleted = true;

entity.DeletedOn = DateTime.UtcNow;

this.Update(entity);

}

}

public class EfRepository<TEntity> : IRepository<TEntity>

where TEntity : class

{

public EfRepository(ApplicationDbContext context)

{

this.Context = context ?? throw new ArgumentNullException(nameof(context));

this.DbSet = this.Context.Set<TEntity>();

}

protected DbSet<TEntity> DbSet { get; set; }

protected ApplicationDbContext Context { get; set; }

public virtual IQueryable<TEntity> All() => this.DbSet;

public virtual IQueryable<TEntity> AllAsNoTracking() => this.DbSet.AsNoTracking();

public virtual Task AddAsync(TEntity entity) => this.DbSet.AddAsync(entity).AsTask();

public virtual void Update(TEntity entity)

{

var entry = this.Context.Entry(entity);

if (entry.State == EntityState.Detached)

{

this.DbSet.Attach(entity);

}

entry.State = EntityState.Modified;

}

public virtual void Delete(TEntity entity) => this.DbSet.Remove(entity);

public Task<int> SaveChangesAsync() => this.Context.SaveChangesAsync();

public void Dispose()

{

this.Dispose(true);

GC.SuppressFinalize(this);

}

protected virtual void Dispose(bool disposing)

{

if (disposing)

{

this.Context?.Dispose();

}

}

}

public interface IAuditInfo

{

DateTime CreatedOn { get; set; }

DateTime? ModifiedOn { get; set; }

}

public abstract class BaseModel<TKey> : IAuditInfo

{

[Key]

public TKey Id { get; set; }

public DateTime CreatedOn { get; set; }

public DateTime? ModifiedOn { get; set; }

}

public abstract class BaseDeletableModel<TKey> : BaseModel<TKey>, IDeletableEntity

{

public bool IsDeleted { get; set; }

public DateTime? DeletedOn { get; set; }

}

public class EmailSenderService : IEmailSenderService

{

private readonly ISendGridClient sendGridClient;

private readonly IConfiguration configuration;

public EmailSenderService(ISendGridClient sendGridClient, IConfiguration configuration)

{

this.sendGridClient = sendGridClient;

this.configuration = configuration;

}

public async Task<Response> SendEmailAsync(string toEmail, string subject, string body)

{

var message = new SendGridMessage()

{

From = new EmailAddress(configuration.GetValue<string>("EmailSender:FromEmail")),

Subject = subject,

PlainTextContent = body

};

message.AddTo(new EmailAddress(toEmail));

return await sendGridClient.SendEmailAsync(message);

}

}

public class StudentService : IStudentService

{

private readonly IDeletableEntityRepository<ApplicationUser> userRepository;

private readonly IExpressionBuilder expressionBuilder;

public StudentService(IDeletableEntityRepository<ApplicationUser> userRepository, IExpressionBuilder expressionBuilder)

{

this.userRepository = userRepository;

this.expressionBuilder = expressionBuilder;

}

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

{

var user = await userRepository

.AllAsNoTracking()

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

.FirstOrDefaultAsync();

if (user != null)

{

var teacher = await userRepository

.AllAsNoTracking()

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

.FirstOrDefaultAsync();

user.TeacherId = teacherId;

teacher.Students.Add(user);

userRepository.Update(user);

userRepository.Update(teacher);

await userRepository.SaveChangesAsync();

return true;

}

return false;

}

public async Task DeleteFromTeacherListAsync(string studentId, string teacherId)

{

var studentToRemove = await userRepository

.AllAsNoTracking()

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

.FirstOrDefaultAsync();

var teacher = await userRepository

.AllAsNoTracking()

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

.FirstOrDefaultAsync();

studentToRemove.TeacherId = null;

teacher.Students.Remove(studentToRemove);

userRepository.Update(studentToRemove);

userRepository.Update(teacher);

await userRepository.SaveChangesAsync();

}

public async Task<IList<T>> GetAllStudentsAsync<T>(string teacherId = null, string groupId = null)

{

var query = userRepository.AllAsNoTracking();

if (groupId != null)

{

query = query.Where(x => !x.StudentsInGroups.Select(x => x.GroupId).Contains(groupId));

}

if (teacherId != null)

{

query = query.Where(x => x.TeacherId == teacherId);

}

return await query.Where(x => !x.Roles.Any()).To<T>().ToListAsync();

}

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

int page,

int countPerPage,

string searchCriteria = null,

string searchText = null,

string roleId = null)

{

var query = userRepository.AllAsNoTracking();

query = roleId != null

? query.Where(x => x.Roles.Any(x => x.RoleId == roleId))

: query.Where(x => x.Roles.Any());

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

{

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

query = query.Where(filter);

}

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

.Skip(countPerPage \* (page - 1))

.Take(countPerPage)

.To<T>()

.ToListAsync();

}

public async Task<int> GetAllInRolesCountAsync(string searchCriteria = null, string searchText = null, string roleId = null)

{

var query = userRepository.AllAsNoTracking().Where(x => x.Roles.Any());

query = roleId != null

? query.Where(x => x.Roles.Any(x => x.RoleId == roleId))

: query.Where(x => x.Roles.Any());

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

{

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

query = query.Where(filter);

}

return await query.CountAsync();

}

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

{

return await userRepository

.AllAsNoTracking()

.Where(x => x.StudentsInGroups.Select(x => x.GroupId).Contains(groupId))

.To<T>()

.ToListAsync();

}

public async Task<int> GetAllStudentsCountAsync(string teacherId = null, string searchCriteria = null, string searchText = null)

{

var query = userRepository.AllAsNoTracking().Where(x => !x.Roles.Any());

if (teacherId != null)

{

query = query.Where(x => x.TeacherId == teacherId);

}

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

{

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

query = query.Where(filter);

}

return await query.CountAsync();

}

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

int page,

int countPerPage,

string teacherId = null,

string searchCriteria = null,

string searchText = null)

{

var query = userRepository.AllAsNoTracking().Where(x => !x.Roles.Any());

if (teacherId != null)

{

query = query.Where(x => x.TeacherId == teacherId);

}

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

{

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

query = query.Where(filter);

}

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

.Skip(countPerPage \* (page - 1))

.Take(countPerPage)

.To<T>()

.ToListAsync();

}

}

public class UserAccountService : IUserAccountService

{

private readonly UserManager<ApplicationUser> userManager;

private readonly IEmailSenderService emailSender;

public UserAccountService(UserManager<ApplicationUser> userManager, IEmailSenderService emailSender)

{

this.userManager = userManager;

this.emailSender = emailSender;

}

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

{

newUser.UserName = newUser.Email;

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

return result.Succeeded;

}

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);

return isRightPassword;

}

public async Task<string> SendPasswordResetEmail(string email)

{

var user = await userManager.FindByEmailAsync(email);

if (user == null)

{

return null;

}

var resetToken = await userManager.GeneratePasswordResetTokenAsync(user);

var sendEmailResponse = await emailSender.SendEmailAsync(email, "Password reset token", resetToken);

if (!sendEmailResponse.IsSuccessStatusCode)

{

return null;

}

return resetToken;

}

# **Приложение Б (cправочное)**

# **https://sun9-69.userapi.com/impg/yaaoofrVSb5VP7B0GhyBWA29-lfzlrTaZbL7ig/bScjHBJD9PE.jpg?size=917x705&quality=96&sign=66d4e0b4211f0d45ec5be6f7143bfd26&type=album****Формы выходных документов**

Рисунок Б.1

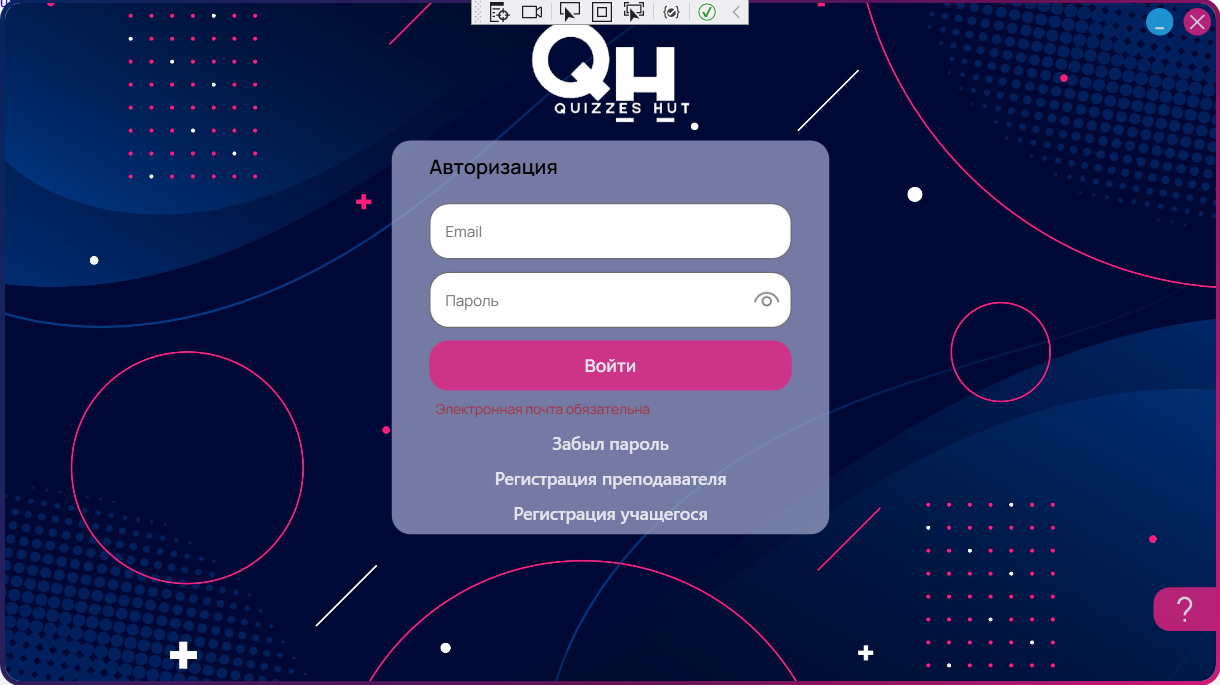
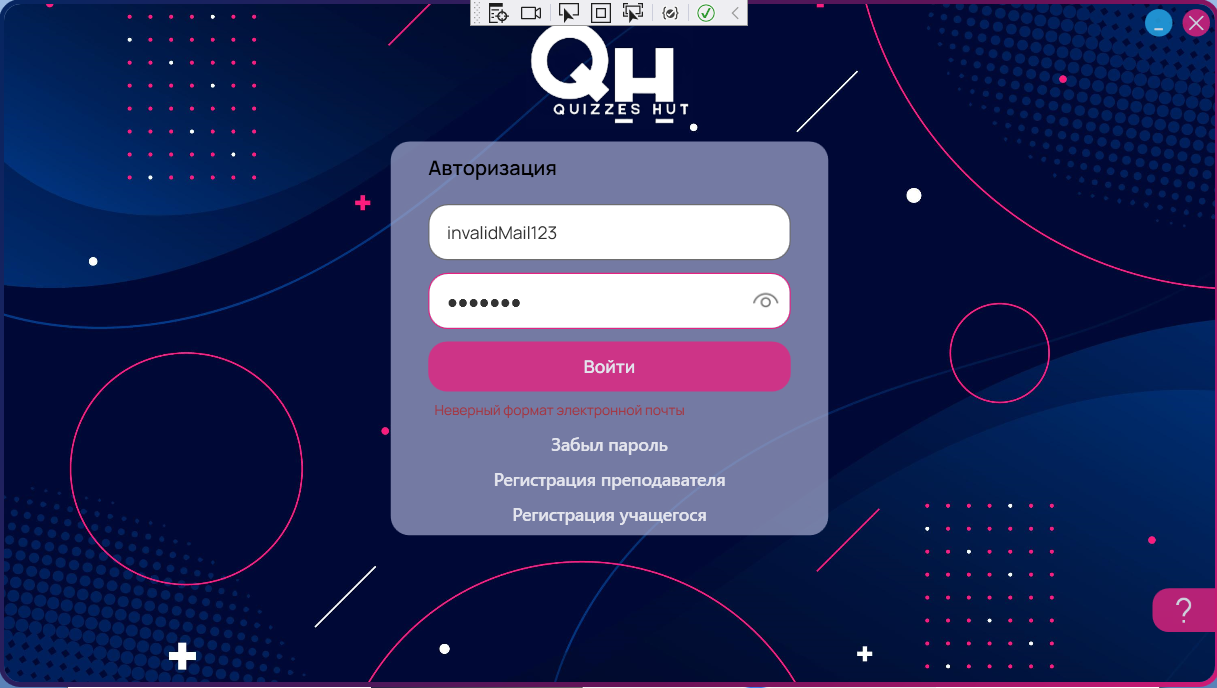


Рисунок Б.2

Рисунок Б.3

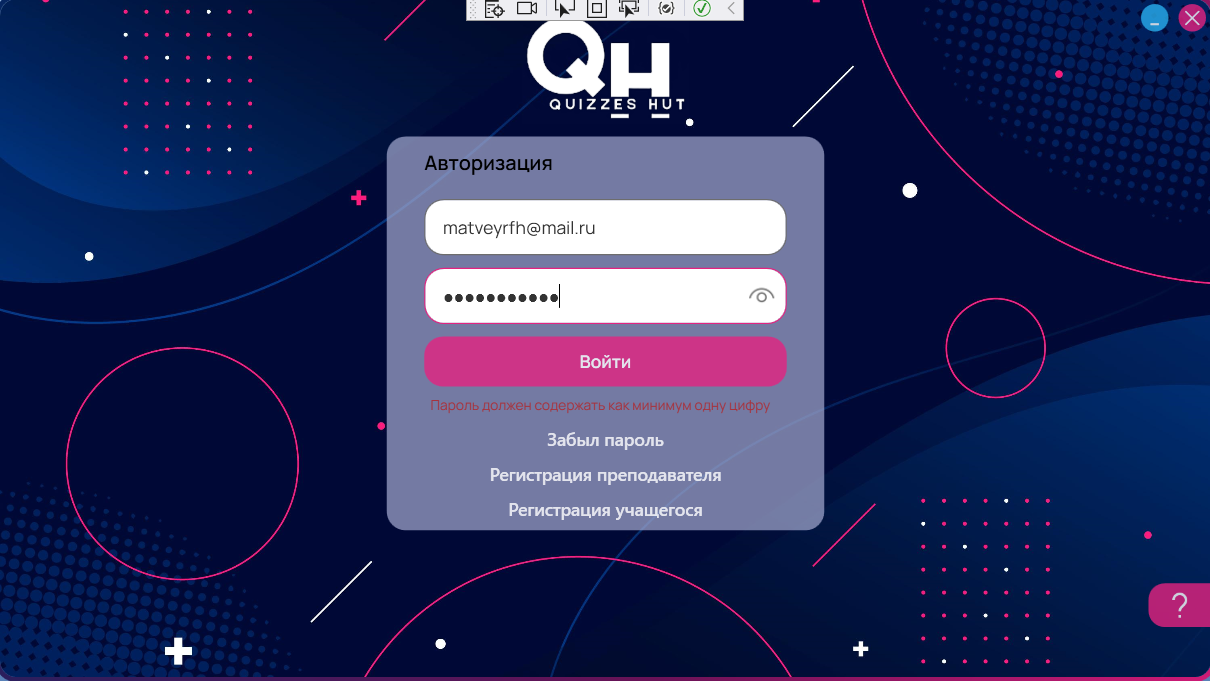


Рисунок Б.4

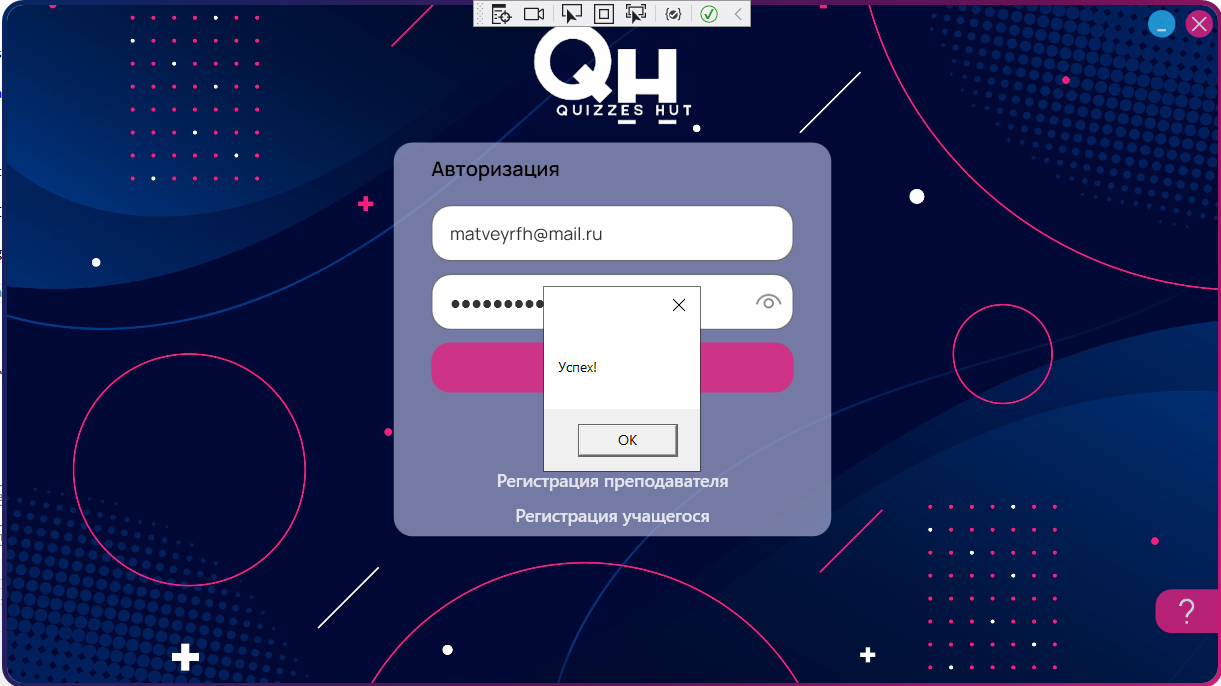


Рисунок Б.5

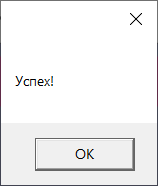


Рисунок Б.6

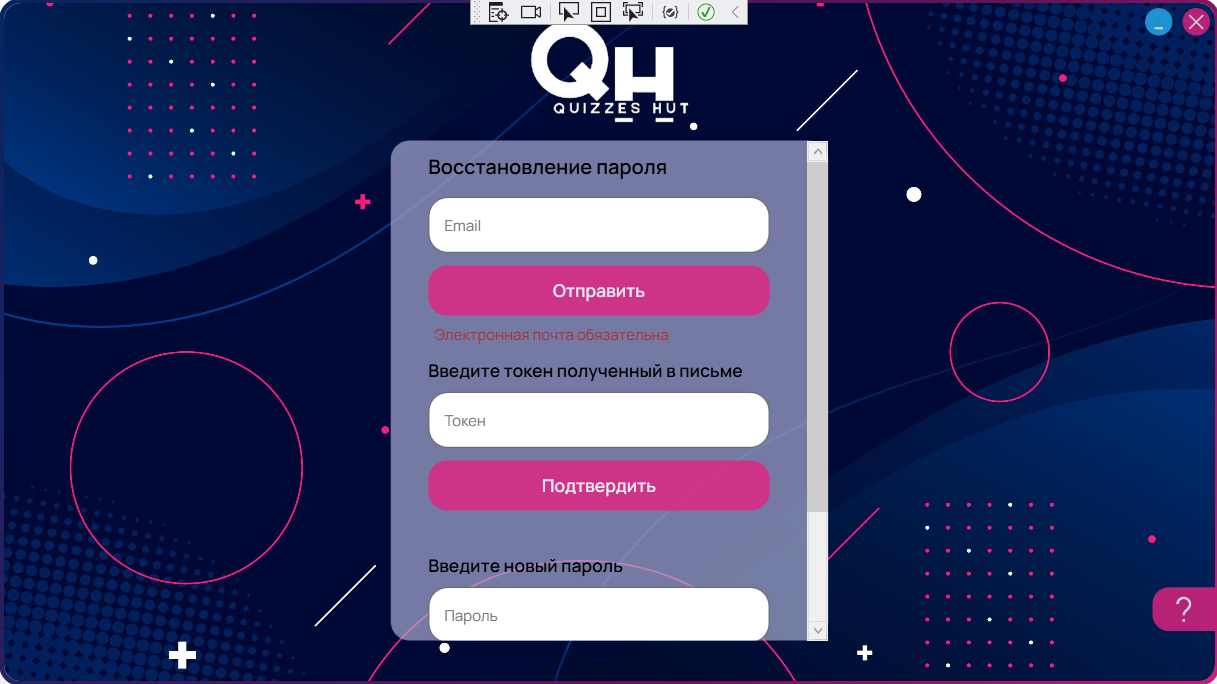


Рисунок Б.7

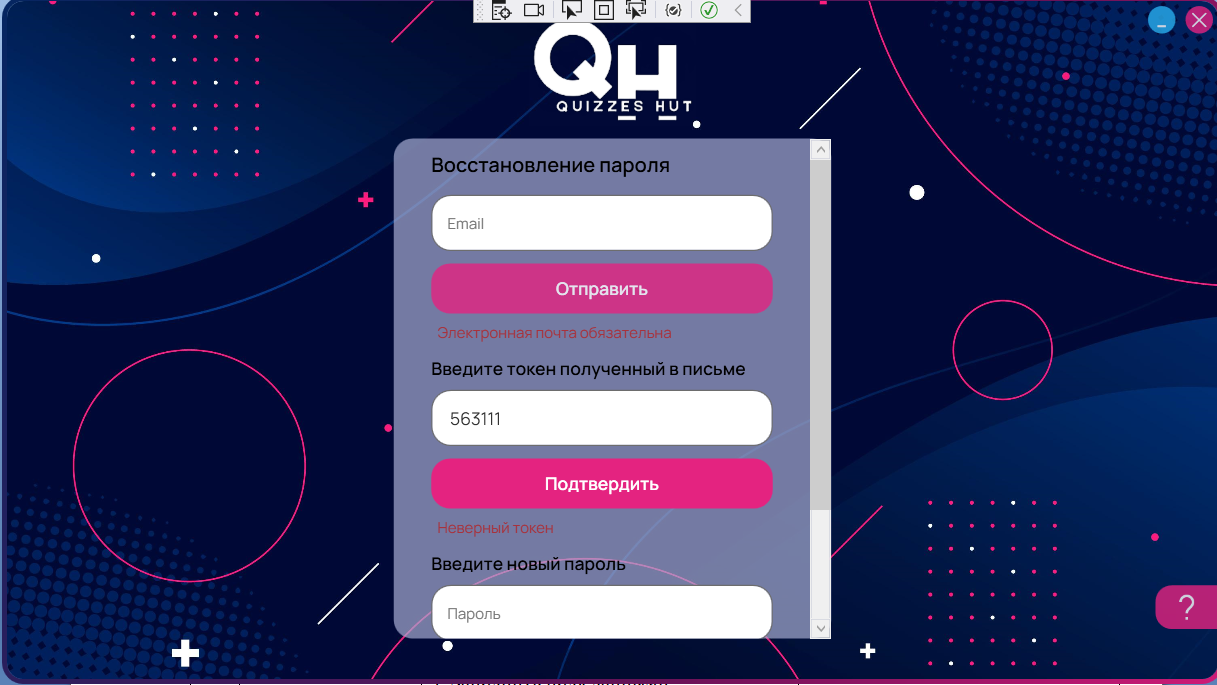
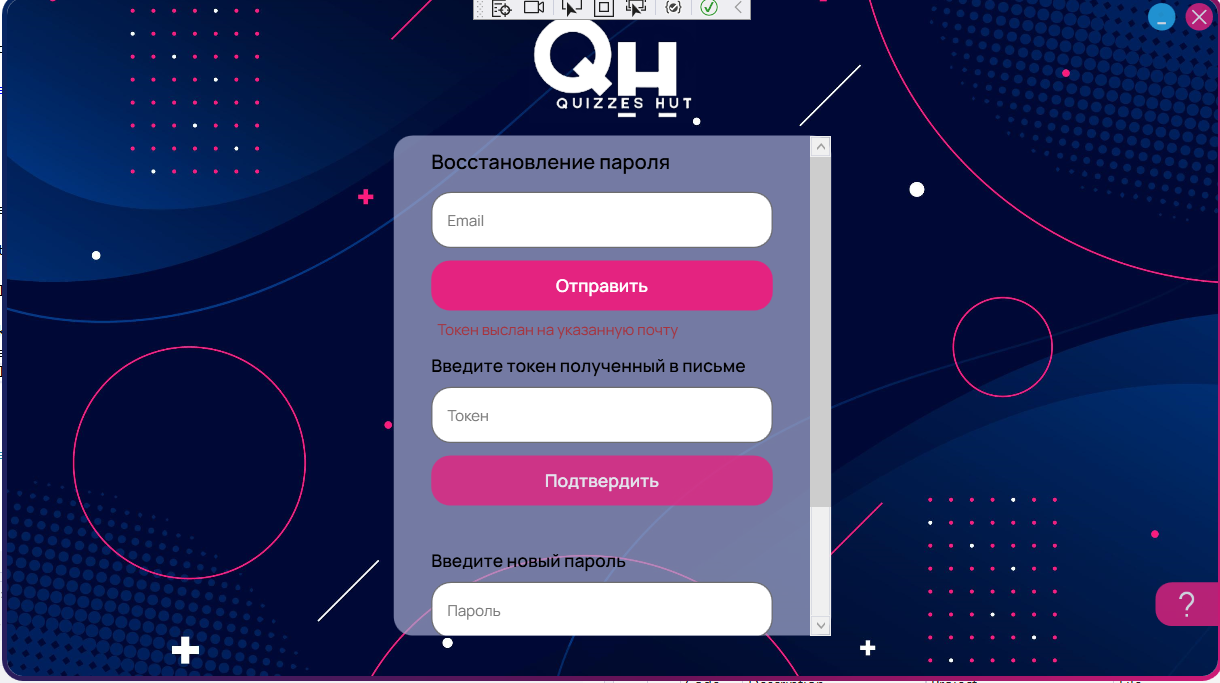


Рисунок Б.8

Рисунок Б.9

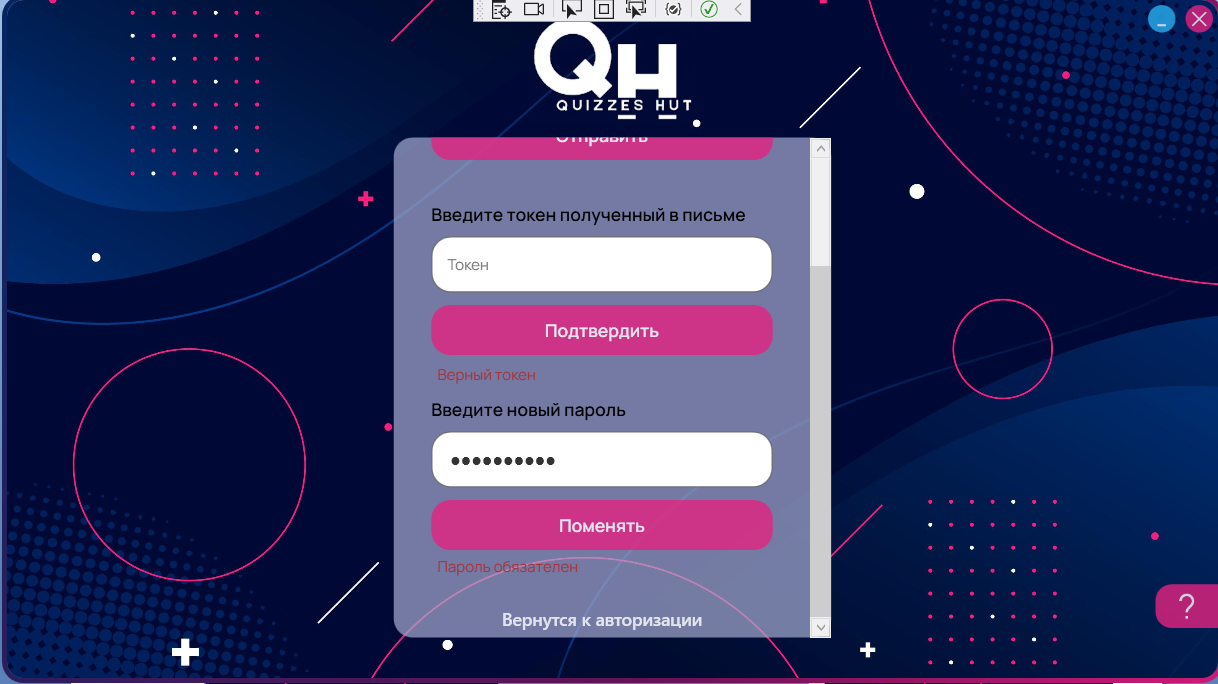


Рисунок Б.10

Рисунок Б.11

Рисунок Б.12

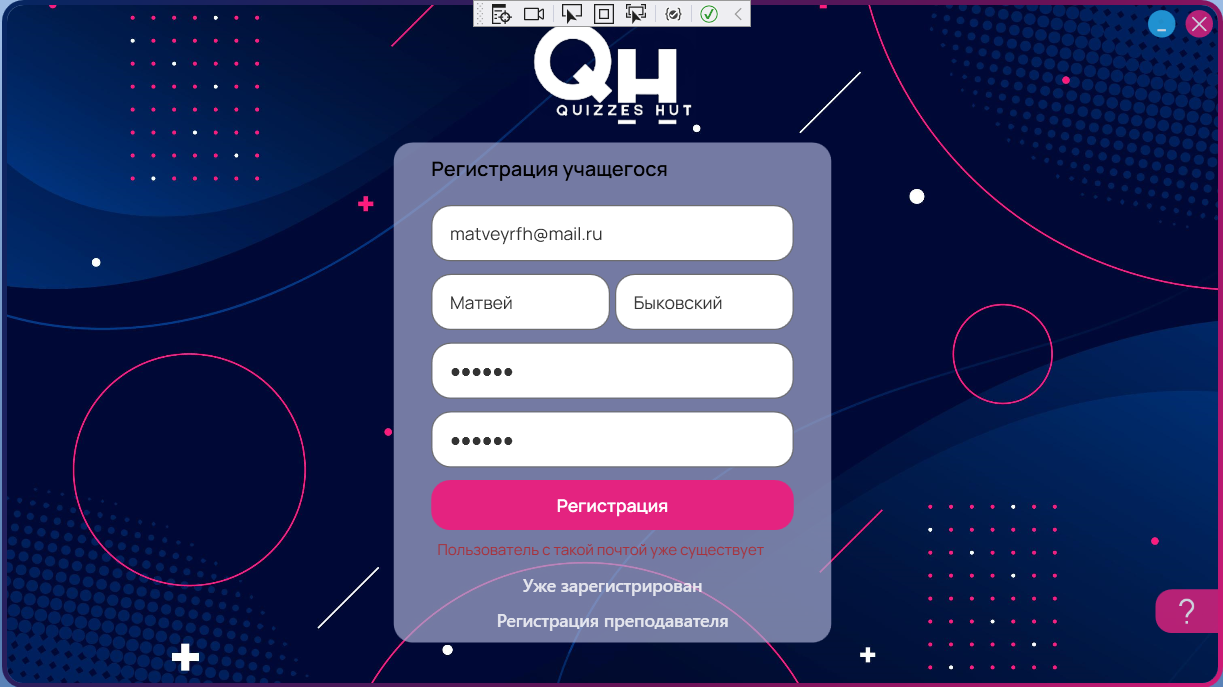


Рисунок Б.13

Рисунок Б.14

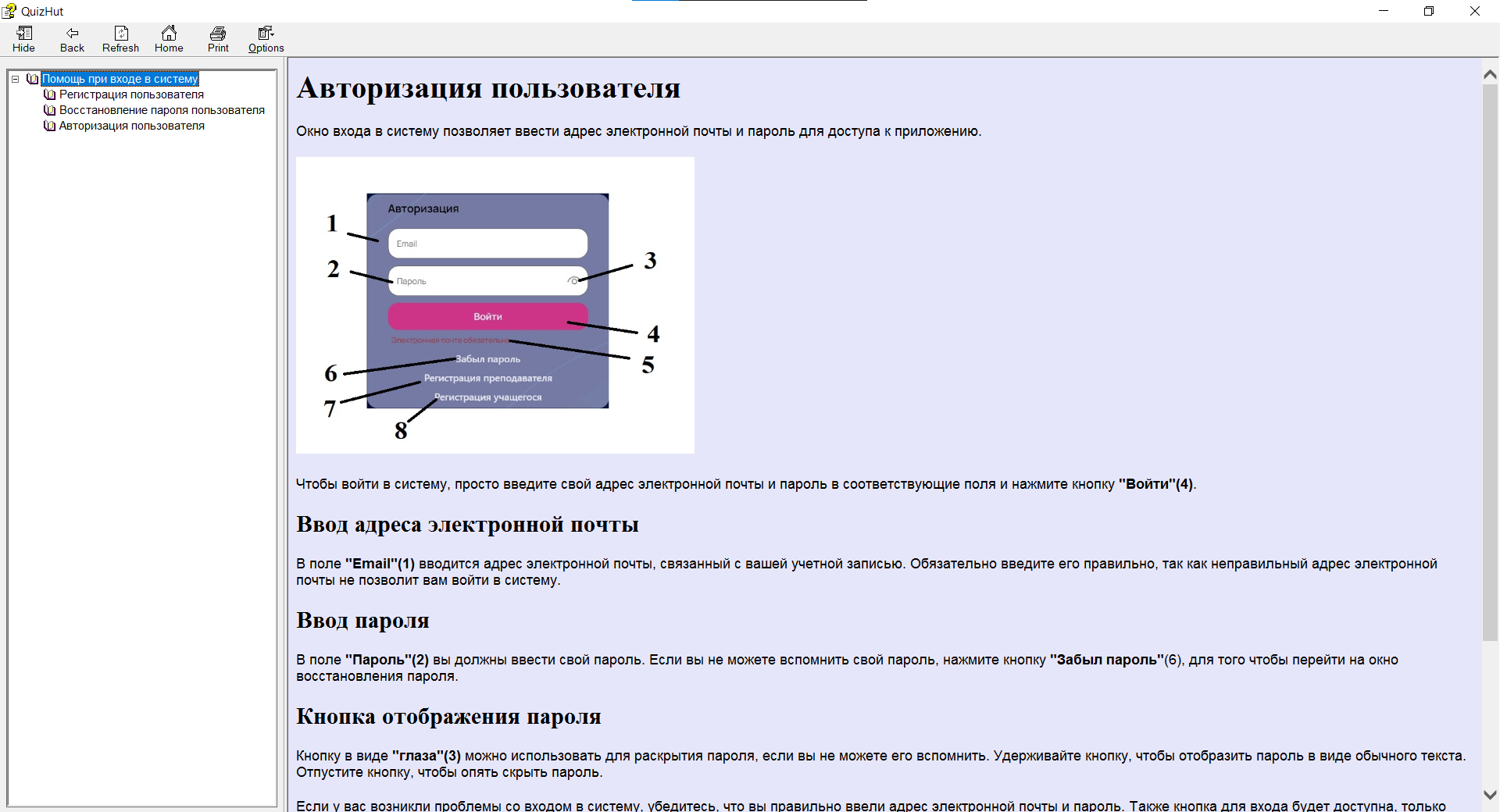


Рисунок Б.15

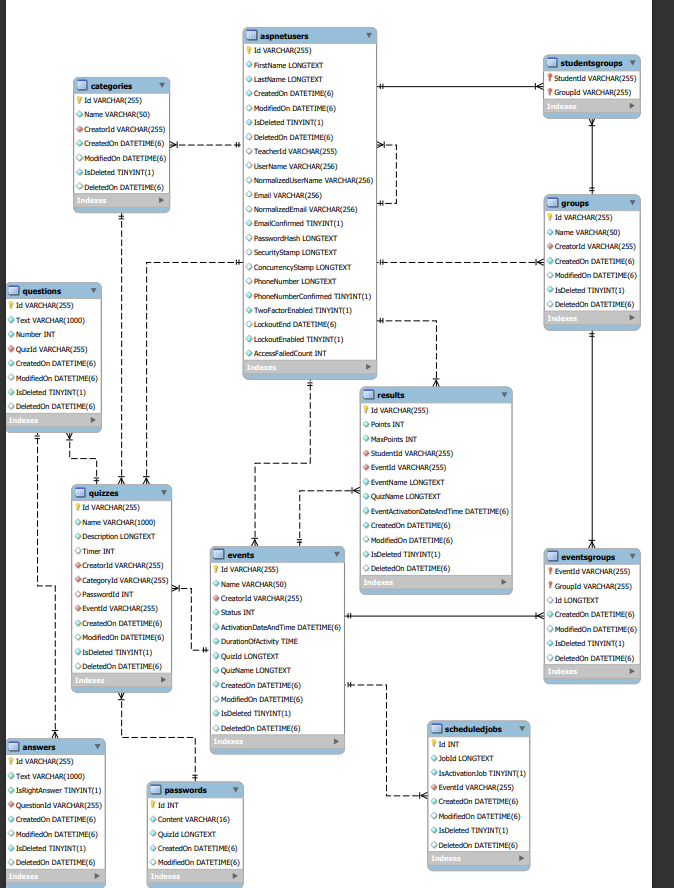


Рисунок Б.16



ОП Т.992002

Т. Контр.

Н.Контр.

Утверд.

КБиП

Листов 3

Провер.

Разраб.

Подп. и дата

Инв.№дубл.

№ докум.

Подпись

Дата

Изм.

Лист

Взам.инв.№

Подп. и дата

*Отчет по преддипломной практике*

*Диаграмма вариантов использования*

У

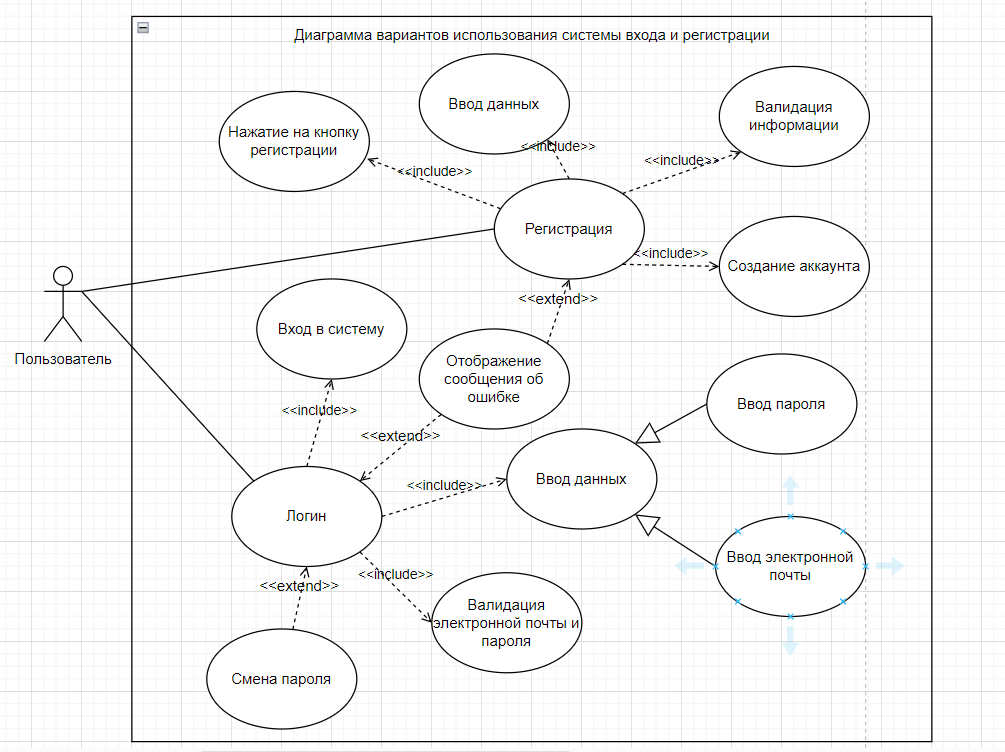
Масса

Лит.

Масштаб

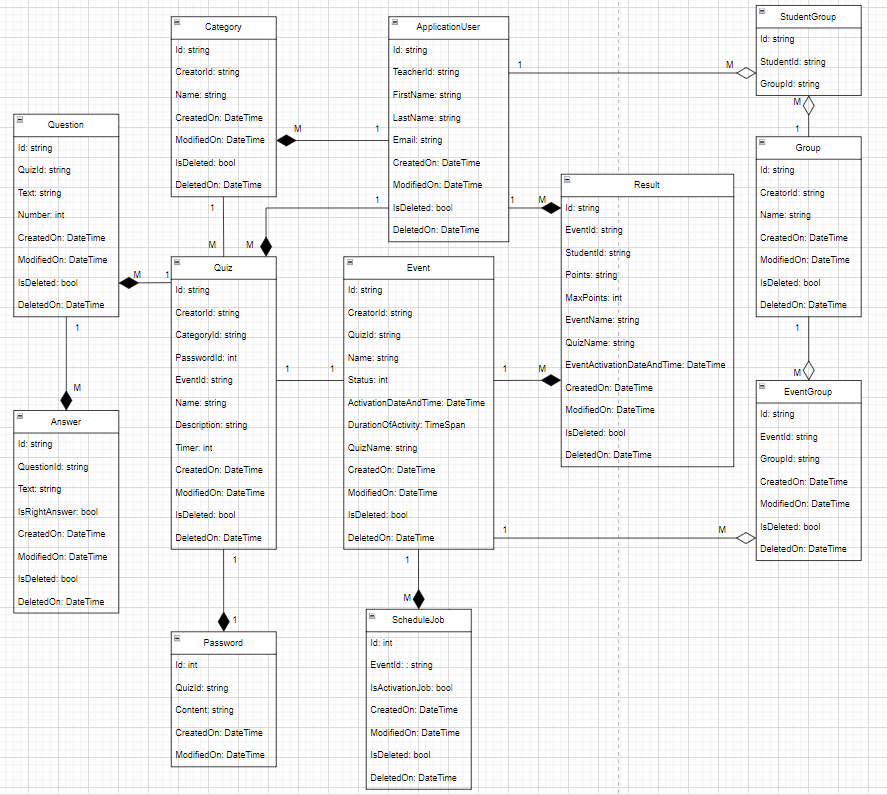
Лист 1

Реценз.



Коропа Е.Н.

Быковский М.М.



Быковский М.М.

Коропа Е.Н.

ОП Т.992002

Т. Контр.

Н.Контр.

Утверд.

КБиП

Листов 3

Провер.

Разраб.

Подп. и дата

Инв.№дубл.

№ докум.

Подпись

Дата

Изм.

Лист

Взам.инв.№

Подп. и дата

*Отчет по преддипломной практике*

*Диаграмма классов*

У

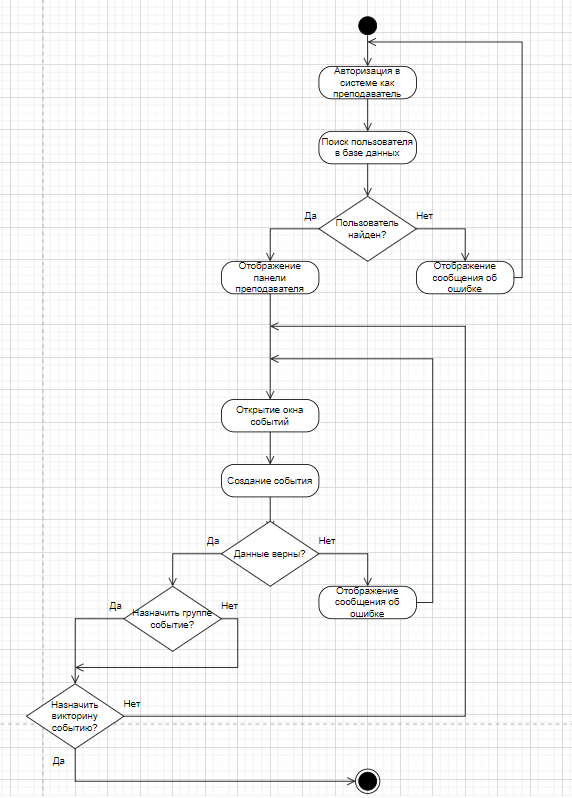
Масса

Лит.

Масштаб

Лист 2

Реценз.



Коропа Е.Н.

Быковский М.М.

ОП Т.992002

Т. Контр.

Н.Контр.

Утверд.

КБиП

Листов 3

Провер.

Разраб.

Подп. и дата

Инв.№дубл.

№ докум.

Подпись

Дата

Изм.

Лист

Взам.инв.№

Подп. и дата

*Отчет по преддипломной практике*

*Диаграмма деятельности*

У

Масса

Лит.

Масштаб

Лист 3

Реценз.