Учреждение образования

«Гродненский государственный политехнический колледж»

**ОТЧЁТ  
ПО УЧЕБНОЙ ПРАКТИКЕ ПО ПРОГРАММИРОВАНИЮ**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Учащейся |  | 3 | | | | |  | курса, группы |  | ПЗТ-40 |
| специальности | | |  | 2 - 40 01 01 «Программное обеспечение информационных | | | | | | |
|  | | |  | технологий» | | | | | | |
| Место прохождения практики | | | | |  | УО «Гродненский государственный | | | | |
|  | | | | |  | политехнический колледж» | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Тема проекта: |  | «Разработка программного обеспечения «Спортивная викторина» | | |
| Ссылка на проект: | | |  | https://github.com/TsiupakovaKseniya/praktika-Csharp.git |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Выполнила |  |  |  | К.Ю.Тюпакова |
|  |  |  |  | (инициалы, фамилия) |
| Руководитель практики от колледжа |  |  |  | В.В.Хомич |
|  |  |  | (инициалы, фамилия) |

Содержание

[Введение. Описание структуры предприятия 3](#_Toc194324354)

[1 Анализ предметной области и формулировка требований к программе 5](#_Toc194324355)

[1.1 Исследование предметной области 5](#_Toc194324356)

[1.2 Инструменты разработки 6](#_Toc194324357)

[2 Проектирование 7](#_Toc194324358)

[2.1 Диаграмма вариантов использования 7](#_Toc194324359)

[2.2 Диаграмма деятельности 7](#_Toc194324360)

[2.3 Описание тестов 8](#_Toc194324361)

[3 Построение программы 13](#_Toc194324362)

[4 Тестирование 17](#_Toc194324363)

[5 Применение 18](#_Toc194324364)

[5.1 Назначение и условия применения программы 18](#_Toc194324365)

[5.2 Инсталляция 18](#_Toc194324366)

[5.3 Выполнение программы 20](#_Toc194324367)

[Заключение 25](#_Toc194324368)

[Список использованных источников 26](#_Toc194324369)

[Приложение A Листинг программы 27](#_Toc194324370)

[Приложение Б Диаграмма классов 38](#_Toc194324370)

Введение

На учебной практике была поставлена задача разработать программное обеспечение «Спортивная викторина».

Цель проекта – разработка программного обеспечения, которое позволит пользователям выбирать вид спорта и уровень сложности для прохождения викторины. Будут включены такие виды спорта как: лёгкая атлетика, настольный теннис, плавание, волейбол, футбол, баскетбол. Данное программное обеспечение будет иметь удобный и интуитивно понятный интерфейс, который позволит пользователям развивать критическое мышление, память и внимание.

Далее приведено краткое описание разделов пояснительной записки.

Первый раздел носит название «Анализ предметной области и формулировка требований к программе». В нем можно ознакомиться с постановкой задачи, которая включает в себя: исследование предметной области поставленной задачи, определение ее организационно-экономической сущности. Также в этом разделе можно узнать о том, как данная задача решается в настоящее время. Все входные и выходные данные тоже будут описаны в первом разделе. В подразделе «Инструменты разработки» будет рассмотрена среда, в которой создаётся данный проект. Здесь также будут установлены минимальные и оптимальные требования к аппаратным характеристикам, обеспечивающим правильное функционирование поставленной задачи.

В разделе «Проектирование задачи» будут рассмотрены основные аспекты разработки программного продукта. Здесь можно будет узнать об организации данных в контексте среды разработки. В данном разделе будет описан пользовательский интерфейс, составлены алгоритмы процесса обработки информации.

«Построение программы» – это третий раздел отчёта по практике, в котором описываются все элементы и объекты, которые будут использованы при реализации данного программного обеспечения. В этом разделе будут описаны функции пользователя и их структура. Здесь можно будет найти таблицу, в которой будет представлена полная аннотация файлов, используемых в данном проекте.

Четвёртый раздел – «Тестирование». В нем будет описано полное и функциональное тестирование данного проекта, т.е. будет протестирована каждая операция, которая выполняется приложением. Будут смоделированы все возможные действия пользователя при работе с данным приложением.

В разделе «Применение» будет описано назначение, область применения, среда функционирования программного обеспечения.

«Заключение» будет содержать краткую формулировку задачи, результаты проделанной работы, описание использованных методов и средств, описание степени автоматизации процессов на различных этапах разработки.

В разделе «Список использованных источников» будет приведён список используемых при разработке источников.

В приложении А к пояснительной записке будет приведен листинг программы.

В приложении Б будет представлена диаграмма классов.

1 Анализ предметной области и формулировка требований к программе

1.1 Исследование предметной области

Наименованием задачи является программное обеспечение «Спортивная викторина».

Цель разработки – это разработка программного обеспечения для изучения видов спорта таких как: лёгкая атлетика, настольный теннис, плавание, волейбол, футбол, баскетбол.

Назначение:данный программный продукт разрабатывается для пользователей, которые хотят подробнее изучить виды спорта, а также проверить свои знания.

В сети Интернет можно найти множество сайтов по данной тематике. Однако, десктопного приложения не разработано.

Создание программного обеспечения «Спортивная викторина» необходимо для того, чтобы предоставить пользователям увлекательный и познавательный способ изучения различных видов спорта. Этот программный продукт будет способствовать популяризации спорта, расширению словарного запаса и развитию спортивных интересов у пользователей всех возрастов.

Таким образом, создание программного обеспечения «Спортивная викторина» направлено на развитие и поддержку памяти и внимания, предоставление пользователям интересного способа изучения видов спорта и создание устойчивого проекта, который будет востребован широкой аудиторией.

Периодичность использования данного программного продукта неограниченна. Пользователь может в любое время установить, а также удалить приложение после провождения всех тем.

Цель данного программного обеспечения – предоставление увлекательного и образовательного инструмента для изучения и поддержки спортивных интересов, популяризации спорта, а также развитии критического мышления и внимания.

Программный продукт должен располагать своим интерфейсом и функционалом к пользователю, чтобы ему было комфортно его использовать.

Данный проект должен стать общедоступным для всех пользователей. В поставленной задаче необходимо реализовать максимально простой пользовательский интерфейс, позволяющий использовать проект пользователю, не обладающему дополнительными знаниями электронно-вычислительной техники.

1.2 Инструменты разработки

Для разработки проекта будет выбрана среда разработки Visual Studio 2022, которая является наиболее актуальной средой для создания программных продуктов данного типа.

Разработка будет производиться на языке программирования C#.

Иные инструменты, используемые при разработке и написании сопутствующей документации:

* Figma – будет использоваться для создания UI макетов проекта;
* web-ресурс DRAW.IO – будет использоваться для создания графической части и разработки UML-диаграмм;
* Microsoft Office Word 2021 – для написания документации к программному продукту.

При разработке данного программного продукта был использован компьютер со следующими характеристиками:

* процессор 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz 2.42 GHz;
* ОЗУ: 8Gb;
* память: HDD 512Gb;
* ОС: Windows 10.

2 Проектирование

2.1 Диаграмма вариантов использования

Диаграмма вариантов использования – диаграмма, отражающая отношения между актерами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью так называемых вариантов использования.

Применительно к разрабатываемому продукту можно выделить одного актера: пользователь.

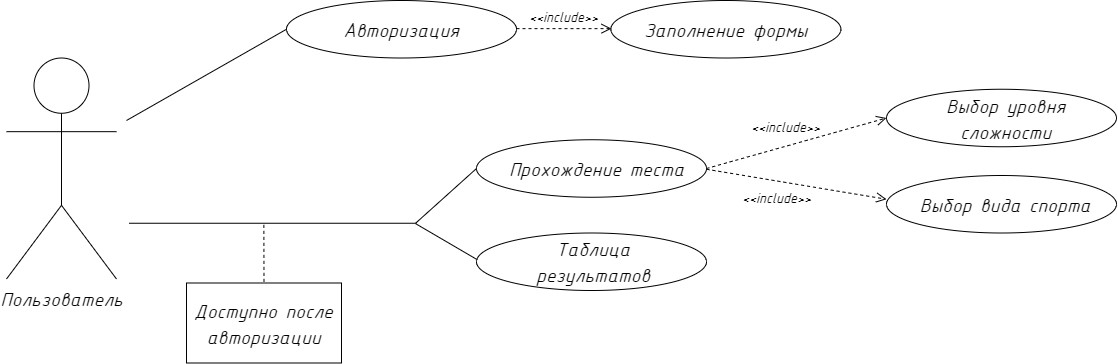
Для пользователя будут доступны следующие варианты использования: авторизация, выбор вида спорта, выбор уровня сложности, прохождение тестов и просмотр таблицы результатов прохождения теста.

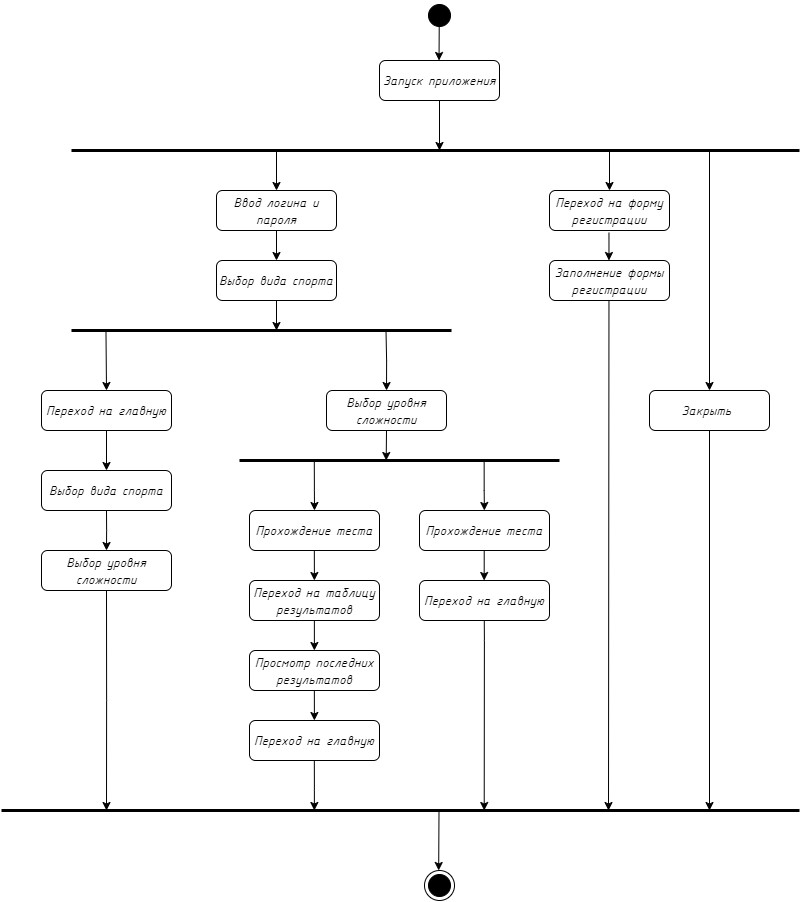
Диаграмма вариантов использования представлена на рисунке 1.

Рисунок 1 – Диаграмма вариантов использования

**2.2 Диаграмма деятельности**

Диаграмма деятельности представляет собой поведенческую диаграмму, которая иллюстрирует поток работы или действий в рамках системы или процесса. Она показывает последовательность шагов и возможные варианты выполнения работы, включая параллельные процессы и ветвления. Диаграмма деятельности включает в себя такие элементы, как начальные и конечные узлы, узлы действий, узлы решений, вилки и слияния, а также потоки управления, которые связывают эти узлы.

Диаграмма деятельности представлена на рисунке 2.

Рисунок 2 – Диаграмма деятельности

2.3 Описание тестов

При разработке проекта необходимо будет провести некоторые функциональные и нефункциональные тесты. Они должны охватывать различные аспекты функциональности, такие как авторизация, регистрация, тесты.

Таблица 1 – Тесты на использование

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | Приоритет | Заглавие и шаги выполнения | Ожидаемый результат |
| 1 | 2 | 3 | 4 |
| T\_01 | крайне высокий | **Регистрация, корректные данные, кнопка «ОК»**  1. Запустить приложение «Спортивная викторина».  2. Нажать на кнопку «Регистрация»  2. В поле «Имя» ввести «Ксения». | 1. Запускается приложение «Спортивная викторина».  2.Осуществляется переход на форму «Регистрация».  2. Введённое значение отображается в поле «Имя». |

Продолжение таблицы 1

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
|  |  | 3. В поле «Логин» ввести «ksuu».  4. В поле «Пароль» ввести «12345».  5. В поле «Подтверждение пароля» ввести «12345».  6. Нажать кнопку «Зарегистрироваться». | 3. Введённое значение отображается в поле «Логин».  4. Введённое значение отображается в поле «Пароль».  5. Введённое значение отображается в поле «Подтверждение пароля».  6. На форме появляется всплывающее окно «Регистрация прошла успешно». После нажатия на кнопку «ОК» осуществляется переход на форму «Авторизация». |
| T\_02 | крайне высокий | **Авторизация, корректные данные**  1. Запустить приложение «Спортивная викторина».  2. В поле «Логин» ввести «ksuu».  3. В поле «Пароль» ввести «12345».  4. Нажать кнопку «Войти». | 1. Запускается приложение «Спортивная викторина».  2. Введённое значение отображается в поле «Логин».  3. Введённое значение отображается в поле «Пароль».  4. На форме появляется всплывающее окно «Успешный вход!». После нажатия на кнопку «ОК» осуществляется переход на форму «Главная». |
| T\_03 | низкий | **Регистрация, окно «Успех», кнопка «Х»**  1. Запустить приложение «Спортивная викторина».  2. В поле «Имя» ввести «Ксения».  3. В поле «Логин» ввести «ksuuuu».  4. В поле «Пароль» ввести «12345».  5. В поле «Подтверждение пароля» ввести «12345».  6. Нажать кнопку «Зарегистрироваться». | 1. Запускается приложение «Спортивная викторина».  2. Введённое значение отображается в поле «Имя».  3. Введённое значение отображается в поле «Логин».  4. Введённое значение отображается в поле «Пароль».  5. Введённое значение отображается в поле «Подтверждение пароля».  6. На форме появляется всплывающее окно |

Продолжение таблицы 1

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
|  |  |  | «Регистрация прошла успешно». После нажатия на кнопку «Х» осуществляется переход на форму «Авторизация». |
| Т\_04 | средний | **Регистрация, некорректный логин**   * Пользователь с логином «ksuu» существует.   1. Запустить приложение «Спортивная викторина».  2. В поле «Имя» ввести «Ксения».  3. В поле «Логин» ввести «ksuu».  4. В поле «Пароль» ввести «12345».  5. В поле «Подтверждение пароля» ввести «12345».  6. Нажать кнопку «Зарегистрироваться». | 1. Запускается приложение «Спортивная викторина».  2. Введённое значение отображается в поле «Имя».  3. Введённое значение отображается в поле «Логин».  4. Введённое значение отображается в поле «Пароль».  5. Введённое значение отображается в поле «Подтверждение пароля».  6. На форме появляется всплывающее окно «Пользователь с таким логином уже существует». |
| Т\_05 | средний | **Авторизация, некорректный логин**  1. Запустить приложение «Спортивная викторина».  2. В поле «Логин» ввести «k».  3. В поле «Пароль» ввести «12345».  4. Нажать кнопку «Войти». | 1. Запускается приложение «Спортивная викторина».  2. Введённое значение отображается в поле «Логин».  3. Введённое значение отображается в поле «Пароль».  4. На форме появляется всплывающее окно «Неверный логин или пароль». |
| Т\_06 | средний | **Авторизация, некорректный пароль**  1. Запустить приложение «Спортивная викторина».  2. В поле «Логин» ввести «ksuu».  3. В поле «Пароль» ввести «1».  4. Нажать кнопку «Войти». | 1. Запускается приложение «Спортивная викторина».  2. Введённое значение отображается в поле «Логин».  3. Введённое значение отображается в поле «Пароль».  4. На форме появляется всплывающее окно |

Продолжение таблицы 1

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
|  |  |  | «Неверный логин или пароль». |
| Т\_07 | средний | **Авторизация, некорректный логин и пароль**  1. Запустить приложение «Спортивная викторина».  2. В поле «Логин» не вводить текст  3. В поле «Пароль» не вводить текст  4. Нажать кнопку «Войти». | 1. Запускается приложение «Спортивная викторина».  2. Введённое значение отображается в поле «Логин».  3. Введённое значение отображается в поле «Пароль».  4. На форме появляется всплывающее окно «Введите логин и пароль» |
| Т\_08 | средний | **Вид спорта «Лёгкая атлетика»**   * Пользователь авторизирован.   1. Нажать на изображение «Лёгкая атлетика».  2. Нажать на кнопку «Лёгкий». | 1. Загружается форма «Выбор сложности».  2. Загружается форма с вопросами с уровнем сложности «Лёгкий». |
| T\_09 | средний | **Вид спорта «Настольный теннис»**   * Пользователь авторизирован.   1. Нажать на изображение «Настольный теннис».  2. Нажать на кнопку «Средний». | 1. Загружается форма «Выбор сложности».  2. Загружается форма с вопросами с уровнем сложности «Средний». |
| T\_10 | средний | **Вид спорта «Плавание»**   * Пользователь авторизирован.   1. Нажать на изображение «Плавание».  2. Нажать на кнопку «Средний». | 1. Загружается форма «Выбор сложности».  2. Загружается форма с вопросами с уровнем сложности «Средний». |
| T\_11 | средний | **Вид спорта «Волейбол»**   * Пользователь авторизирован.   1. Нажать на изображение «Волейбол».  2. Нажать на кнопку «Сложный». | 1. Загружается форма «Выбор сложности».  2. Загружается форма с вопросами с уровнем сложности «Сложный». |
| T\_12 | средний | **Вид спорта «Футбол»**   * Пользователь авторизирован. | 1. Загружается форма «Выбор сложности». |

|  |  |  |  |
| --- | --- | --- | --- |
| Продолжение таблицы 1 | | | |
| 1 | 2 | 3 | 4 |
|  |  | 1. Нажать на изображение «Футбол».  2. Нажать на кнопку «Лёгкий». | 2. Загружается форма с вопросами с уровнем сложности «Лёгкий». |
| T\_13 | средний | **Вид спорта «Баскетбол»**   * Пользователь авторизирован.   1. Нажать на изображение «Баскетбол».  2. Нажать на кнопку «Средний». | 1. Загружается форма «Выбор сложности».  2. Загружается форма с вопросами с уровнем сложности «Средний». |
| Т\_14 | высокий | **Главная**   * Пользователь авторизирован.   1. Нажать на изображение «Волейбол».  2. Нажать на кнопку «Сложный».  3. После прохождения теста нажать кнопку «Главная». | 1. Загружается форма «Выбор сложности».  2. Загружается форма с вопросами с уровнем сложности «Сложный».  3. Осуществляется переход на форму «Главная». |
| T\_15 | высокий | **Таблица результатов**   * Пользователь авторизирован.   1. Нажать на изображение «Волейбол».  2. Нажать на кнопку «Сложный».  3. После прохождения теста нажать кнопку «Таблица результатов». | 1. Загружается форма «Выбор сложности».  2. Загружается форма с вопросами с уровнем сложности «Сложный».  3. Осуществляется переход на форму «Таблица результатов». |

3 Построение программы

Данный программный продукт представляет собой программное обеспечение на языке программирования C#.

Класс «выбор сложности»:

* \_easyMax (тип int): максимальный результат на лёгком уровне;
* \_mediumMax (тип int): максимальный результат на среднем уровне;
* \_hardMax (тип int): максимальный результат на сложном уровне;
* TestName (тип string): название теста.

Класс «главная»:

– currentUser: текущий пользователь (Пользователь), полученный из App.Current.

Класс «загрузка»:

* \_timer – таймер (DispatcherTimer), который обновляет загрузку.
* \_progress (тип int): переменная отслеживания прогресса (от 0 до 100).

Класс «таблица\_лидеров»:

– \_testId (тип int): Id теста;

– currentUser: текущий пользователь (Пользователь), полученный из App.Current.

Класс «тест»:

– currentUser: текущий пользователь, полученный из App.Current;

* \_ answered (тип bool): флаг, показывающий, был ли дан ответ на текущий вопрос;
* Category (тип string): категория теста;
* Level (тип string): уровень сложности теста;
* TestId(тип int): идентификатор теста;
* \_questions: коллекция (ObservableCollection<вопрос>) для хранения загруженных вопросов;
* \_currentIndex(тип int): текущий индекс вопроса;
* \_timer – таймер (DispatcherTimer), отсчитывающий время для ответа;
* \_initialTime (тип int): начальное время на ответ;
* \_currentQuestion: текущий активный вопрос;
* \_timeRemaining (тип int): оставшееся время на ответ;
* Score (тип int): количество правильных ответов;
* SelectedAnswer: выбранный пользователем ответ;

– dataFolder (тип string): путь к папке, где хранится база данных;

– dbPath (тип string): путь к файлу базы данных;

* connStr: строка подключения к базе данных.

Класс «Пользователь»:

* Id (тип int): идентификатор пользователя;
* Name (тип string): имя пользователя;
* Login (тип string): логин пользователя;
* Password (тип string): пароль пользователя.

Класс «вопрос»:

* идентификатор (тип int): идентификатор вопроса;
* содержание(тип string): содержание вопроса;
* вес (тип int): вес вопроса;
* ответы: коллекция возможных ответов (ObservableCollection<ответ>), создаётся при инициализации.

Класс «ответ»:

* идентификатор (тип int): идентификатор ответов;
* содержание (тип string): содержание ответа;
* комментарий (тип string): дополнительная информация или объяснение к ответу;
* правильный (тип bool): указывает, является ли ответ правильным;
* IsSelected (тип bool): показывает, был ли этот ответ выбран пользователем.

Класс «авторизация»:

– dataFolder (тип string): путь к папке, где хранится база данных;

– dbPath (тип string): путь к файлу базы данных;

– connStr (тип string): строка подключения к базе данных.

Класс «регистрация»:

– dataFolder (тип string): путь к папке, где хранится база данных;

– dbPath (тип string): путь к файлу базы данных;

– connString (тип string): строка подключения к базе данных.

Каждый класс имеет свои собственные поля для хранения данных, а также конструкторы, которые могут быть использованы для инициализации объектов.

Приложение имеет следующую структуру:

Методы в классе «выбор сложности»:

– RaisePropertyChanged(string propName = null) – уведомляет интерфейс об изменении значения свойства;

– выбор\_сложности(string name) – конструктор, который инициализирует окно, устанавливает название теста и загружает результаты;

– LoadMaxScores(string testName) – загружает максимальные баллы пользователя по каждому уровню сложности из базы данных;

– Difficulty\_Click(object sender, MouseButtonEventArgs e) – обработчик клика, вызывающий RouteToTest(level);

– RouteToTest(string level) – создаёт окно теста с выбранной сложностью;

– BackButton\_Click(object sender, RoutedEventArgs e) – возвращает пользователя на главное окно.

Методы в классе «главная»:

* главная() – конструктор, который инициализирует главное окно и загружает текущего пользователя;
* Tile\_MouseLeftButtonUp(object sender, MouseButtonEventArgs e) – обработчик клика мыши, который извлекает название теста и вызывает RouteToTest(testName);
* RouteToTest(string testName) – метод, открывающий окно выбора сложности (выбор\_сложности) и закрывающий текущее окно (this.Close()).

Методы в классе «загрузка»:

* загрузка() – конструктор, который инициализирует окно, запускает таймер и задаёт интервал его работы;
* Timer\_Tick(object sender, EventArgs e) – метод, вызываемый при каждом обновлении таймера: Если \_progress >= 100, таймер останавливается и открывается окно авторизация; в противном случае \_progress увеличивается, обновляется ProgressBar и ProgressText.

Методы в классе «таблица\_лидеров»:

* таблица\_лидеров(int testId) – конструктор, который инициализирует окно, сохраняет идентификатор теста и загружает данные таблицы лидеров;
* LoadLeaderboard() – загружает список лидеров из базы данных;
* HomeButton\_Click(object sender, RoutedEventArgs e) – обработчик кнопки возврата на главное окно.

Методы в классе «тест»:

* тест(string category, string level) – конструктор, который инициализирует окно теста, загружает вопросы и устанавливает таймер;
* LoadQuestionsFromDb(int testId) – загружает вопросы и ответы из базы данных;
* ShowCurrentQuestion() – отображает текущий вопрос и запускает таймер;
* Timer\_Tick(object sender, EventArgs e) – обновляет оставшееся время и автоматически завершает вопрос при окончании времени;
* NextButton\_Click(object sender, RoutedEventArgs e) – обработчик кнопки "Далее": если ответ не был дан, вызывается SubmitAnswer(); иначе переход к следующему вопросу.
* SubmitAnswer() – проверяет ответ: останавливает таймер; проверяет, является ли выбранный ответ правильным; показывает сообщение с правильным ответом; меняет кнопку "Ответить" на "Далее".
* FinishTest() – завершает тест: считает итоговые баллы; скрывает элементы теста; показывает окно результатов; сохраняет результат в базу данных;
* OnPropertyChanged(string prop) – обновляет UI при изменении данных;
* HomeButton\_Click(object sender, RoutedEventArgs e) – переход на главное окно;
* LeaderboardButton\_Click(object sender, RoutedEventArgs e) – открывает таблицу лидеров (таблица\_лидеров).

Методы в классе «авторизация»:

* авторизация() – конструктор, который инициализирует окно авторизации;
* LoginButton\_Click(object sender, RoutedEventArgs e) – обработчик кнопки входа:проверяет введённые логин и пароль; подключается к базе данных и проверяет пользователя; при успешном входе сохраняет данные пользователя и открывает главное окно (главная); при неудаче выводит сообщение об ошибке;
* Registration\_Click(object sender, RoutedEventArgs e) – обработчик кнопки регистрации: показывает сообщение о переходе на форму регистрации; открывает окно регистрации (регистрация).

Методы в классе «регистрация»:

* регистрация() – конструктор, который инициализирует окно регистрации;
* RegisterButton\_Click(object sender, RoutedEventArgs e) – обработчик кнопки "Зарегистрироваться": валидирует введённые данные; проверяет, существует ли пользователь с таким логином; если пользователь новый, записывает его в базу данных и открывает окно авторизации; если данные невалидны, выводит сообщение об ошибке;
* BackToLogin\_Click(object sender, RoutedEventArgs e) – переход к форме авторизации (авторизация).

Данная структура программы представляет собой программное обеспечение с модульной организацией кода, разделяющим логику управления данными и сами данные по сущностям. Что в последствии позволит расширять программный продукт.

Диаграмма классов представлена в приложении Б.

4 Тестирование

При разработке данной программы многие возникающие ошибки и недоработки были исправлены на этапе реализации проекта. После завершения этапа написания программы было проведено тщательное функциональное тестирование.

Элементы программы были проверены, и было установлено, что все они работают правильно и выполняют задачи, указанные в классах.

Расписание проведения и время, затраченное на тестирование, описано в таблице 2.

Таблица 2 – Расписание работ над проектом

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Дата | Деятельность | Продолжительность, ч |
| Тюпакова Ксения | 07.04.2025 | Составление вопросов для каждого вида спорта | 5 |
| Тюпакова Ксения | 11.04.2025 | Разработка тестов | 4 |
| Тюпакова Ксения | 13.04.2025 | Тестирование приложения | 2 |
| Тюпакова Ксения | 16.04.2025 | Исправление найденных ошибок | 6 |
| Тюпакова Ксения | 25.04.2025 | Проведение регрессионного тестирования | 3 |
| Тюпакова Ксения | 30.04.2025 | Составление отчета о результатах тестирования | 3 |

Элементы программы были проверены, и было установлено, что все они работают правильно и выполняют задачи, указанные в процедурах.

Статистика по всем дефектам представлена в таблице 3.

Таблица 3 – Статистика по всем дефектам

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Статус | Количество | Важность | | | |
| Низкая | Средняя | Высокая | Критическая |
| Найдено | 0 | 0 | 0 | 0 | 0 |
| Исправлено | 0 | 0 | 0 | 0 | 0 |
| Проверено | 0 | 0 | 0 | 0 | 0 |
| Открыто заново | 0 | 0 | 0 | 0 | 0 |
| Отклонено | 0 | 0 | 0 | 0 | 0 |

5 Применение

5.1 Назначение и условия применения программы

Цель данного проекта заключается в создании увлекательного и образовательного инструмента для изучения спортивных направлений, популяризации спорта, развитии памяти и внимания.

Создаваемое приложение будет рассчитано на любого пользователя.

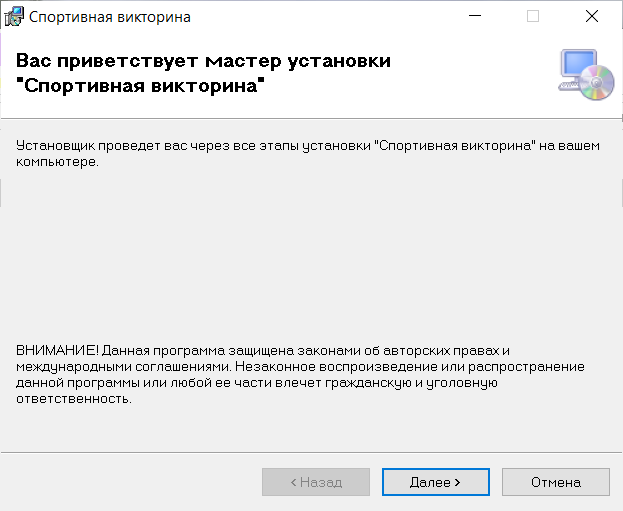
Быстродействие любой программы во многом зависит от характеристик выбранного персонального компьютера: рабочей частоты процессора, объема оперативной памяти и т.д. Несмотря на все реализованные в ней задачи, она легко запускается и функционирует на любых машинах.

Программа разработана на ПК со следующими характеристиками:

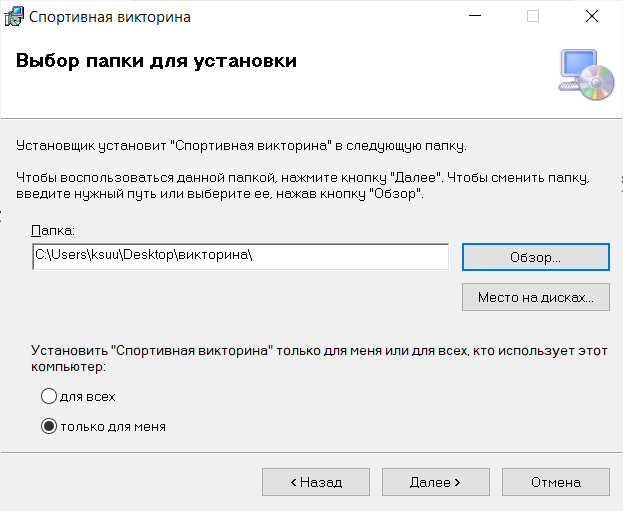
* процессор 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz 2.42 GHz;
* ОЗУ: 8Gb;
* память: HDD 512Gb;
* ОС: Windows 10.

5.2 Инсталляция

Для инсталляции приложения необходимо запустить setup.exe. Открывается «Мастер установки», представленный на рисунке 3.

Рисунок 3 – Мастер установки

После нажатия кнопки «Далее», появляется возможность выбора места для установки программного продукта. Окно выбора папки для установки представлено на рисунке 4.

Рисунок 4 – Выбор папки для установки

После нажатия на кнопку «Далее» открывается форма подтверждения установки, представленная на рисунке 5.

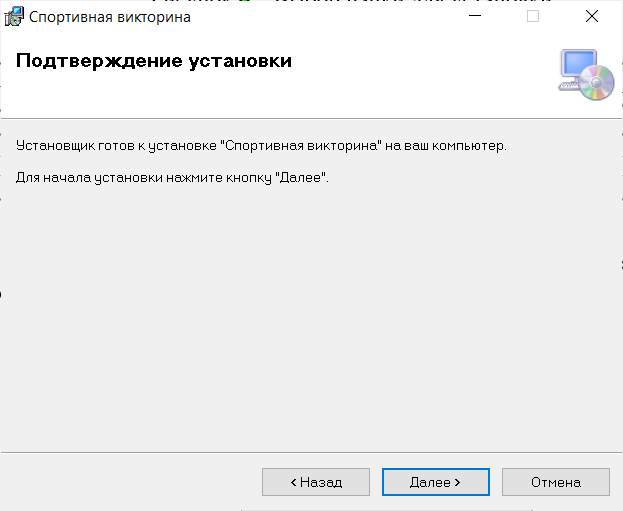


Рисунок 5 – Подтверждение установки

Окно, представленное на рисунке 6, информирует об успешной установке.

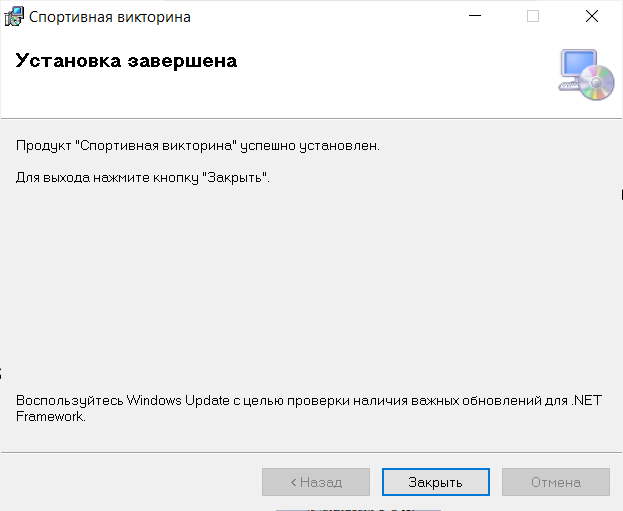


Рисунок 6 – «Установка завершена»

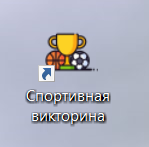
После установки на «Рабочем столе» появится ярлык для запуска приложения, который представлен на рисунке 7.

Рисунок 7 – Ярлык приложения

**5.3 Выполнение программы**

После запуска приложения открывается форма «Загрузка», представленная на рисунке 8.

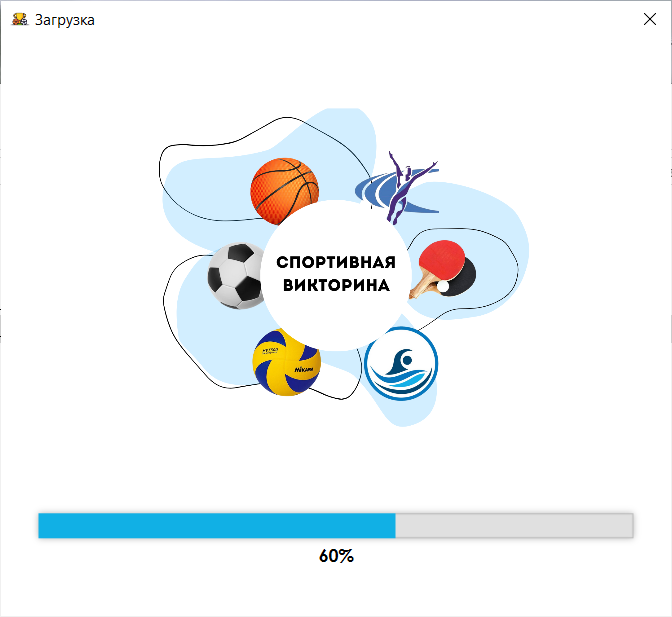


Рисунок 8 – Форма «Загрузка»

После загрузки приложения открывается форма «Авторизация». Форма авторизации представлена на рисунке 9.

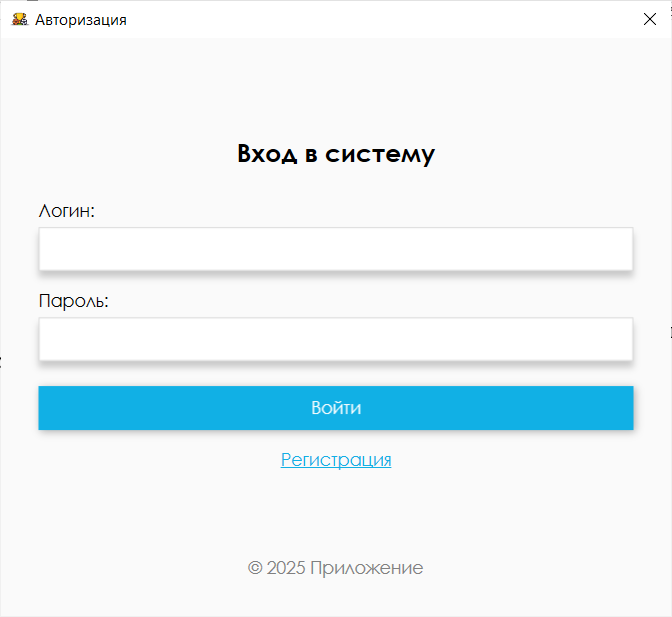


Рисунок 9 – Форма «Авторизация»

Если пользователь не авторизован, необходимо перейти на форму «Регистрация» и ввести данные. После успешной регистрации, выполнить переход на форму «Авторизация». На рисунке 10 представлена форма регистрации.

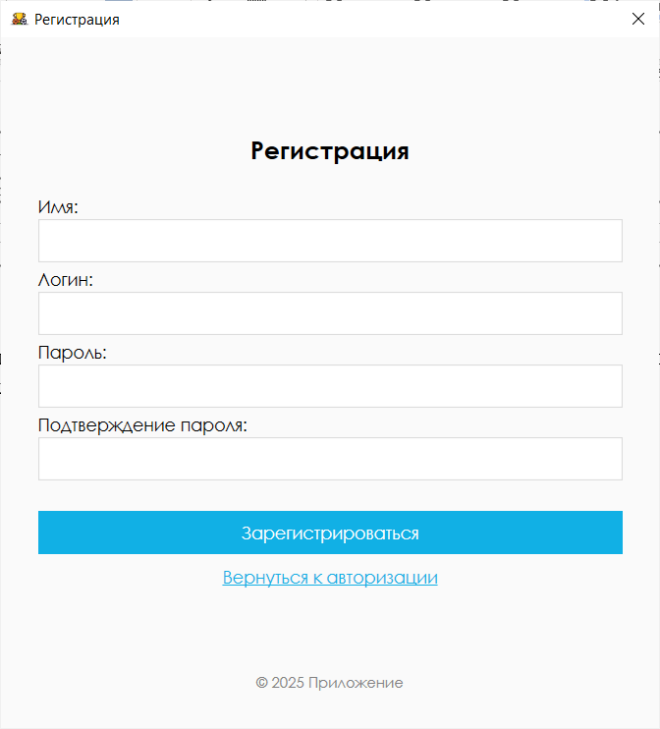
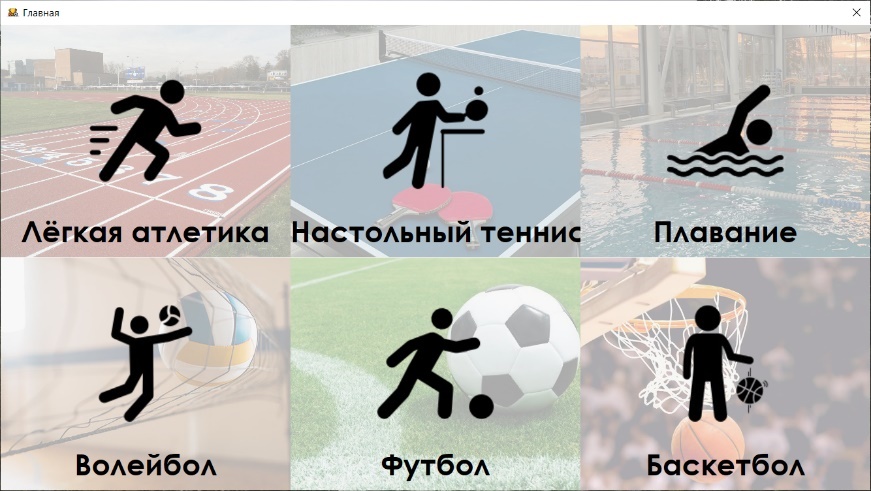
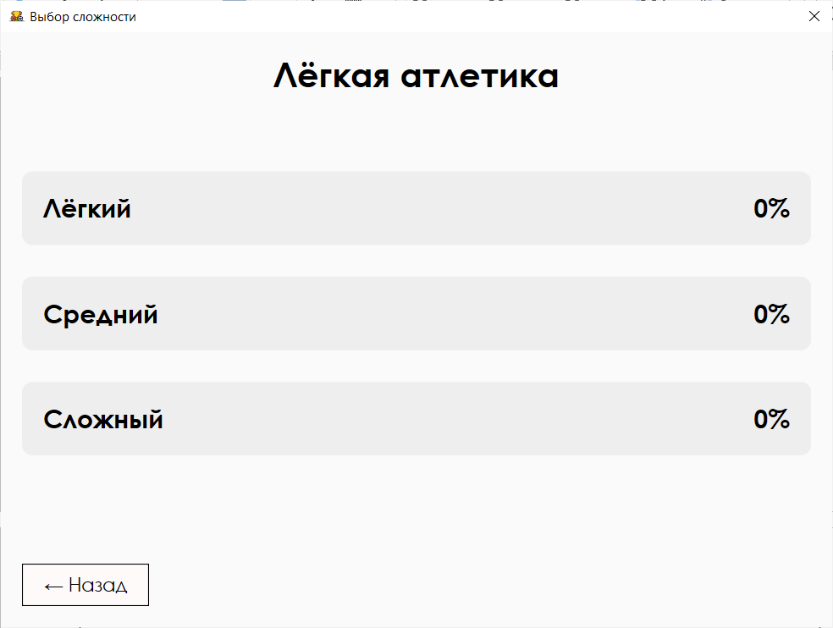


Рисунок 10 – Форма «Регистрация»

Авторизовавшись, пользователь перенаправляется на форму «Главная», которая представлена на рисунке 11. Она включает в себя выбор вида спорта.

Рисунок 11 – Форма «Главная»

Форма «Лёгкая атлетика» включает в себя выбор уровня сложности для прохождения данного теста. Форма «Выбор сложности» представлена на рисунке 12.

Рисунок 12 – Форма «Выбор сложности»

После нажатия на кнопку «Лёгкий» осуществится переход на форму «Тест», где будут подгружены вопросы с вариантами ответов на тему «Лёгкая атлетика», а также установлен таймер. На каждый вопрос даётся 30 секунд. Если время истекло, то ответ считается неверным. Таймер увеличивается на среднем и сложном уровне сложности до 45 и 60 секунд соответственно. На рисунке 13 представлена форма «Тест».

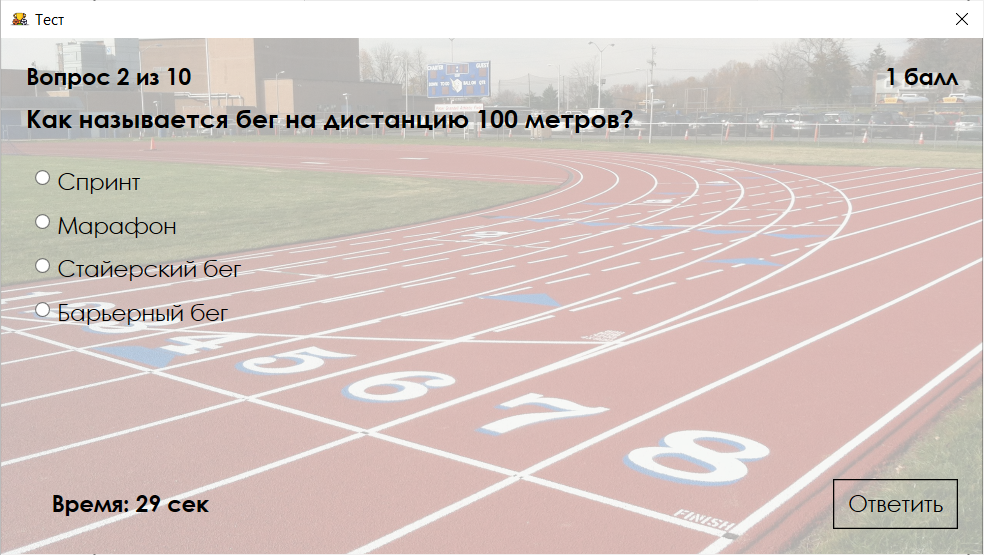


Рисунок 13 – Форма «Тест»

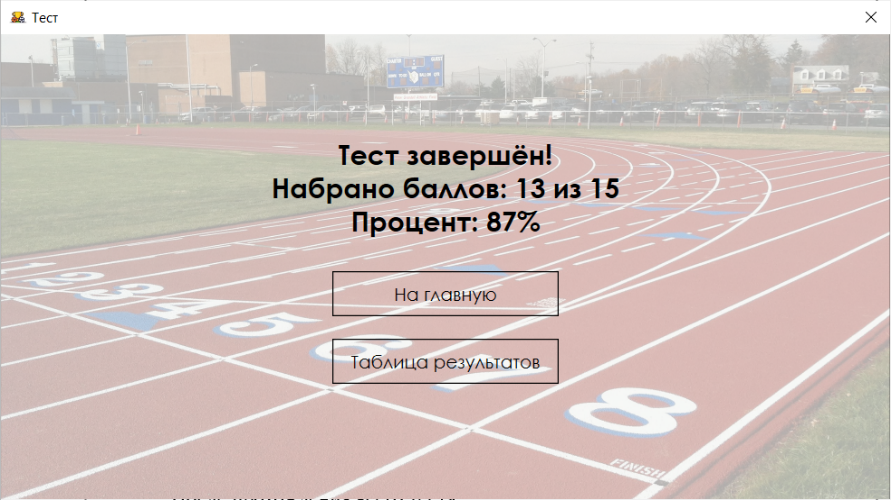
После прохождения всего теста, выводятся результаты в баллах и процентах. На рисунке 14 представлены результаты пройденного теста.

Рисунок 14 – Результаты теста

После нажатия на кнопку «На главную» осуществляется переход на форму с выбором вида спорта. Форма «Главная» представлена на рисунке 11.

После нажатия на кнопку «Таблица результатов» осуществляется переход на таблицу с последними тремя результатами пройденного теста. Форма «Таблица результатов» представлена на рисунке 15.



Рисунок 15 – Форма «Таблица результатов»

Формы «Настольный теннис», «Плавание», «Волейбол», «Футбол», «Баскетбол» имеют идентичный вид, представленный на рисунках 12, 13, 14.

**Заключение**

Целью данного проекта была разработка программного обеспечения «Спортивная викторина». В ходе тестирования не было выявлено исключительных ситуаций. Проект работает без сбоев и ошибок. В процессе разработки программного продукта были улучшены навыки использования C#, закреплены умения создания собственных функций, и обработки всех исключительны ситуации. Все поставленные требования были выполнены и проект реализован успешно.

Особое внимание было уделено пользовательскому опыту, что позволило создать интуитивно понятный и удобный интерфейс, который делает программный продукт удобным и доступным для широкой аудитории. В ходе тестирования приложение продемонстрировало высокую стабильность и надежность, что подтверждает его готовность к использованию конечными пользователями.

Также стоит отметить, что проект «Спортивная викторина» помогает поддерживать спортивный интерес, развивать критическое мышление, память и внимание.

Подводя итог, можно сказать, что разработка данного программного обеспечения является отличным примером успешного выполнения всех этапов разработки программного обеспечения, от первоначального планирования до окончательного тестирования и запуска продукта.

Список использованных источников

1. Марченко, А. Л. Основы программирования на С# 2.0. / А.Л. Марченко. – Москва: 2011. – 552 c.
2. Албахари Д. C# 5.0. Справочник. Полное описание языка / Д. Албахари, Б. Албахари – М.: Вильямс, 2013. – 1008 с.
3. Троелсен, Эндрю Язык программирования C# 5.0 и платформа .NET 4.5 / Эндрю Троелсен. - М.: Вильямс, 2015. – 486 c.
4. Руководство по C# [Электронный ресурс]. – Режим доступа: <https://vscode.ru/prog-lessons/sozdanie-redaktirovanie-i-udalenie-dannyih-xml-fayla-c-sharp.html> – Дата доступа: 10.04.2025.
5. Руководство по социализации XML в C# [Электронный ресурс]. – Режим доступа: <https://metanit.com/sharp/tutorial/6.4.php> – Дата доступа: 13.04.2025.

Приложение A  
Листинг программы

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data.OleDb;

using System.Linq;

using System.Reflection.Emit;

using System.Runtime.CompilerServices;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace Спорт\_тесты

{

/// <summary>

/// Логика взаимодействия для выбор\_сложности.xaml

/// </summary>

public partial class выбор\_сложности : Window, INotifyPropertyChanged

{

private int \_easyMax;

public int EasyMax

{

get => \_easyMax;

set { \_easyMax = value; RaisePropertyChanged(); }

}

private int \_mediumMax;

public int MediumMax

{

get => \_mediumMax;

set { \_mediumMax = value; RaisePropertyChanged(); }

}

private int \_hardMax;

public int HardMax

{

get => \_hardMax;

set { \_hardMax = value; RaisePropertyChanged(); }

}

private string TestName{ get; set; }

private void RaisePropertyChanged([CallerMemberName] string propName = null) =>

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propName));

public event PropertyChangedEventHandler PropertyChanged;

public выбор\_сложности(string name)

{

InitializeComponent();

DataContext = this;

TestName = name;

NameBox.Text = TestName;

LoadMaxScores(TestName);

}

private void LoadMaxScores(string testName)

{

var user = ((App)Application.Current).CurrentUser;

if (user == null) return;

string dataFolder = System.IO.Path.Combine(

Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData),

"SportTestsDB");

string dbPath = System.IO.Path.Combine(dataFolder, "db.accdb");

string connStr = $@"Provider=Microsoft.ACE.OLEDB.12.0;Data Source={dbPath};Persist Security Info=False;";

try

{

using (var conn = new OleDbConnection(connStr))

{

conn.Open();

foreach (var level in new[] { "Лёгкий", "Средний", "Сложный" })

{

try

{

// 1) Находим testId

using (var cmdT = new OleDbCommand("SELECT идентификатор FROM Тесты WHERE [вид\_спорта]=? AND [сложность]=?", conn))

{

cmdT.Parameters.AddWithValue("?", testName);

cmdT.Parameters.AddWithValue("?", level);

var o = cmdT.ExecuteScalar();

if (o == null) continue;

int testId = Convert.ToInt32(o);

// 2) Берём MAX(баллы)

int maxScore = 0;

using (var cmdR = new OleDbCommand("SELECT MAX(баллы) FROM результаты WHERE идентификатор\_пользователя=? AND идентификатор\_теста=?", conn))

{

cmdR.Parameters.AddWithValue("?", user.Id);

cmdR.Parameters.AddWithValue("?", testId);

var r = cmdR.ExecuteScalar();

if (r != DBNull.Value && r != null)

maxScore = Convert.ToInt32(r);

}

switch (level)

{

case "Лёгкий": EasyMax = maxScore; break;

case "Средний": MediumMax = maxScore; break;

case "Сложный": HardMax = maxScore; break;

}

}

}

catch (Exception exLevel)

{

MessageBox.Show($"Ошибка уровня «{level}»: {exLevel.Message}", "БД", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при загрузке результатов: {ex.Message}", "БД", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private void Difficulty\_Click(object sender, MouseButtonEventArgs e)

{

if (sender is FrameworkElement fe && fe.Tag is string level)

{

// Тут ваша логика: переход к тесту с выбранным уровнем

RouteToTest(level);

}

}

private void RouteToTest(string level)

{

var testWindow = new тест(TestName, level);

testWindow.Show();

this.Close();

}

private void BackButton\_Click(object sender, RoutedEventArgs e)

{

// Логика возврата, например:

var mainWindow = new главная();

mainWindow.Show();

this.Close();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace Спорт\_тесты

{

/// <summary>

/// Логика взаимодействия для главная.xaml

/// </summary>

public partial class главная : Window

{

private readonly Пользователь currentUser;

public главная()

{

InitializeComponent();

this.currentUser = ((App)Application.Current).CurrentUser;

}

private void Tile\_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)

{

if (sender is FrameworkElement fe && fe.Tag is string testName)

{

RouteToTest(testName);

}

}

private void RouteToTest(string testName)

{

выбор\_сложности wind = new выбор\_сложности(testName);

wind.Show();

this.Close();

}

}

}

using System;

using System.Windows;

using System.Windows.Threading;

namespace Спорт\_тесты

{

/// <summary>

/// Логика взаимодействия для загрузка.xaml

/// </summary>

public partial class загрузка : Window

{

private readonly DispatcherTimer \_timer;

private int \_progress;

public загрузка()

{

InitializeComponent();

\_timer = new DispatcherTimer

{

Interval = TimeSpan.FromMilliseconds(5)

};

\_timer.Tick += Timer\_Tick;

\_timer.Start();

}

private void Timer\_Tick(object sender, EventArgs e)

{

if (\_progress >= 100)

{

\_timer.Stop();

авторизация auth = new авторизация();

auth.Show();

this.Close();

}

else

{

\_progress++;

ProgressBar.Value = \_progress;

ProgressText.Text = $"{\_progress}%";

}

}

}

}

using System;

using System.Data;

using System.Data.OleDb;

using System.Windows;

namespace Спорт\_тесты

{

/// <summary>

/// Логика взаимодействия для таблица\_лидеров.xaml

/// </summary>

public partial class таблица\_лидеров : Window

{

private readonly int \_testId;

private readonly Пользователь currentUser;

public таблица\_лидеров(int testId)

{

InitializeComponent();

\_testId = testId;

this.currentUser = ((App)Application.Current).CurrentUser;

LoadLeaderboard();

}

private void LoadLeaderboard()

{

string dataFolder = System.IO.Path.Combine(

Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData),

"SportTestsDB");

string dbPath = System.IO.Path.Combine(dataFolder, "db.accdb");

string connString = $@"Provider=Microsoft.ACE.OLEDB.12.0;Data Source={dbPath};Persist Security Info=False;";

try

{

using (var conn = new OleDbConnection(connString))

{

conn.Open();

// Ваш обновлённый запрос

string sql = @"SELECT TOP 10

п.имя\_пользователя AS Логин,

р.баллы AS Балл

FROM

результаты р

INNER JOIN пользователи п ON р.идентификатор\_пользователя = п.идентификатор

WHERE

р.идентификатор\_пользователя = ?

AND

р.идентификатор\_теста = ?

ORDER BY

р.идентификатор DESC

";

var cmd = new OleDbCommand(sql, conn);

cmd.Parameters.AddWithValue("?", currentUser.Id); // \_userId — идентификатор пользователя

cmd.Parameters.AddWithValue("?", \_testId); // \_userId — идентификатор пользователя

var adapter = new OleDbDataAdapter(cmd);

var dt = new DataTable();

adapter.Fill(dt);

// Добавляем колонку с нумерацией

dt.Columns.Add("Рейтинг", typeof(int));

dt.Columns.Add("Баллы", typeof(string));

// Форматируем баллы (например, добавляем %)

for (int i = 0; i < dt.Rows.Count; i++)

{

dt.Rows[i]["Рейтинг"] = i + 1;

dt.Rows[i]["Баллы"] = $"{dt.Rows[i]["Балл"]}%";

}

dt.Columns.Remove("Балл");

// Перемещаем колонку "Рейтинг" в начало (опционально)

dt.Columns["Рейтинг"].SetOrdinal(0);

dt.Columns["Баллы"].SetOrdinal(2);

LeaderboardGrid.ItemsSource = dt.DefaultView;

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка загрузки таблицы результатов:\n{ex.Message}",

"Ошибка",

MessageBoxButton.OK,

MessageBoxImage.Error);

}

}

private void HomeButton\_Click(object sender, RoutedEventArgs e)

{

главная mainWindow = new главная();

mainWindow.Show();

this.Close();

}

}

}

using System;

using System.Windows;

using System.Data.OleDb;

using System.IO;

using System.Linq;

using System.Collections.ObjectModel;

using System.ComponentModel;

using System.Windows.Threading;

using System.Runtime.CompilerServices;

using System.Windows.Controls.Primitives;

using System.Windows.Media.Imaging;

using System.Windows.Media;

namespace Спорт\_тесты

{

public partial class тест : Window, INotifyPropertyChanged

{

private readonly Пользователь currentUser;

private bool \_answered;

private string Category { get; set; }

private string Level { get; set; }

private int TestId { get; set; }

private readonly ObservableCollection<вопрос> \_questions = new ObservableCollection<вопрос>();

private int \_currentIndex;

private DispatcherTimer \_timer;

private int \_initialTime;

// Текущий вопрос

private вопрос \_currentQuestion;

public вопрос CurrentQuestion

{

get => \_currentQuestion;

set

{

if (\_currentQuestion == value) return;

\_currentQuestion = value;

OnPropertyChanged(nameof(CurrentQuestion));

OnPropertyChanged(nameof(WeightText));

}

}

public string WeightText

{

get

{

if (CurrentQuestion == null) return "";

int w = CurrentQuestion.вес;

string suffix = (w % 10 == 1 && w % 100 != 11) ? "балл"

: (w % 10 >= 2 && w % 10 <= 4

&& (w % 100 < 10 || w % 100 >= 20)) ? "балла"

: "баллов";

return $"{w} {suffix}";

}

}

// Оставшееся время

private int \_timeRemaining;

public int TimeRemaining

{

get => \_timeRemaining;

set

{

if (\_timeRemaining == value) return;

\_timeRemaining = value;

OnPropertyChanged(nameof(TimeRemaining));

}

}

// Сколько ответов угадано верно

public int Score { get; private set; }

public event PropertyChangedEventHandler PropertyChanged;

public ответ SelectedAnswer { get; set; }

static string dataFolder = System.IO.Path.Combine(

Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData),

"SportTestsDB");

static string dbPath = System.IO.Path.Combine(dataFolder, "db.accdb");

string connStr = $@"Provider=Microsoft.ACE.OLEDB.12.0;Data Source={dbPath};Persist Security Info=False;";

public тест(string category, string level)

{

InitializeComponent();

this.currentUser = ((App)Application.Current).CurrentUser;

DataContext = this;

Category = category;

Level = level;

if (Level == "Сложный")

{

\_initialTime = 60;

}

else if (Level == "Средний")

{

\_initialTime = 45;

}

else

{

\_initialTime = 30;

}

try

{

using (var conn = new OleDbConnection(connStr))

{

conn.Open();

string sql = @"

SELECT COUNT(\*)

FROM Тесты

WHERE [сложность] = ? AND [вид\_спорта] = ?

";

using (var cmd = new OleDbCommand(sql, conn))

{

cmd.Parameters.AddWithValue("?", Level);

cmd.Parameters.AddWithValue("?", Category);

int count = (int)cmd.ExecuteScalar();

if (count > 0)

{

var selectCmd = new OleDbCommand(

"SELECT [идентификатор] FROM Тесты WHERE [сложность] = ? AND [вид\_спорта] = ?",

conn);

selectCmd.Parameters.AddWithValue("?", Level);

selectCmd.Parameters.AddWithValue("?", Category);

using (var reader = selectCmd.ExecuteReader())

{

if (reader.Read())

{

TestId = reader.GetInt32(0);

}

}

}

else

{

MessageBox.Show("Тест не найден");

}

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при подключении к БД:\n{ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

try

{

// Путь к файлу Assets/<Category>\_bg.png

var relativePath = $"/Assets/{Category}\_bg.png";

// создаём URI ресурса, упакованного в приложение

var uri = new Uri($"pack://application:,,,{relativePath}", UriKind.Absolute);

var img = new BitmapImage(uri);

this.Background = new ImageBrush(img)

{

Stretch = Stretch.UniformToFill,

Opacity = 1.0

};

}

catch

{

MessageBox.Show("Картинка фона не найдена");

}

LoadQuestionsFromDb(TestId);

// 3) Готовим таймер

\_timer = new DispatcherTimer { Interval = TimeSpan.FromSeconds(1) };

\_timer.Tick += Timer\_Tick;

// 4) Запускаем первый вопрос

\_currentIndex = 0;

ShowCurrentQuestion();

}

private void LoadQuestionsFromDb(int testId)

{

using (var conn = new OleDbConnection(connStr))

{

conn.Open();

// 2.1. Читаем вопросы

string qSql = "SELECT идентификатор, содержание, вес FROM вопросы WHERE идентификатор\_теста = ? ORDER BY идентификатор";

try

{

using (var qCmd = new OleDbCommand(qSql, conn))

{

qCmd.Parameters.AddWithValue("?", testId);

using (var qReader = qCmd.ExecuteReader())

{

while (qReader.Read())

{

var q = new вопрос

{

идентификатор = qReader.GetInt32(0),

содержание = qReader.GetString(1),

вес = qReader.GetInt32(2)

};

// 2.2. Читаем ответы для этого вопроса

string aSql = "SELECT идентификатор, содержание, правильный, комментарий FROM Ответы WHERE идентификатор\_вопроса = ?";

using (var aCmd = new OleDbCommand(aSql, conn))

{

aCmd.Parameters.AddWithValue("?", q.идентификатор);

using (var aReader = aCmd.ExecuteReader())

{

while (aReader.Read())

{

string comment = aReader.IsDBNull(3)

? string.Empty

: aReader.GetString(3);

q.ответы.Add(new ответ

{

идентификатор = aReader.GetInt32(0),

содержание = aReader.GetString(1),

правильный = aReader.GetBoolean(2),

комментарий = comment

});

}

}

}

\_questions.Add(q);

}

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при подключении к БД:\n{ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

private void ShowCurrentQuestion()

{

if (\_currentIndex >= \_questions.Count)

{

FinishTest();

return;

}

CurrentQuestion = \_questions[\_currentIndex];

QuestionNumberTextBlock.Text = $"Вопрос {\_currentIndex + 1} из {\_questions.Count}";

// Сброс выделения

foreach (var ans in CurrentQuestion.ответы) ans.IsSelected = false;

// Запускаем таймер

TimeRemaining = \_initialTime;

\_timer.Start();

FeedbackTextBlock.Visibility = Visibility.Collapsed;

AnswersListBox.IsEnabled = true;

\_answered = false;

}

private void Timer\_Tick(object sender, EventArgs e)

{

TimeRemaining--;

if (TimeRemaining <= 0)

{

\_timer.Stop();

SubmitAnswer(); // автоматический пропуск (считается неверным)

}

}

private void NextButton\_Click(object sender, RoutedEventArgs e)

{

if (!\_answered)

{

// Если ещё не проверили — проверяем

SubmitAnswer();

}

else

{

// Иначе — переходим к следующему

\_currentIndex++;

ShowCurrentQuestion();

AnswersListBox.IsHitTestVisible = true;

AnswersListBox.Focusable = true;

ActionButton.Content = "Ответить";

}

}

private void SubmitAnswer()

{

\_timer.Stop();

// Находим выбранный и правильный ответы

var selected = CurrentQuestion.ответы.FirstOrDefault(a => a.IsSelected);

var correct = CurrentQuestion.ответы.FirstOrDefault(a => a.правильный);

bool isCorrect = selected != null && selected.правильный;

if (isCorrect) Score += CurrentQuestion.вес;

// Выводим сообщение

FeedbackTextBlock.Text = isCorrect

? "Правильно!"

: $"Неправильно. Правильный ответ: {correct?.комментарий}";

FeedbackTextBlock.Visibility = Visibility.Visible;

// Пометим, что ответ дан — дальше ждём нажатия «Далее»

\_answered = true;

AnswersListBox.IsHitTestVisible = false;

AnswersListBox.Focusable = false;

ActionButton.Content = "Далее";

}

private void FinishTest()

{

// сумма всех весов вопросов

int totalWeight = \_questions.Sum(q => q.вес);

// процент от максимума

int totalScore = (int)Math.Round(Score / (double)totalWeight \* 100);

QuestionNumberTextBlock.Visibility = Visibility.Collapsed;

QuestionTextBlock.Visibility = Visibility.Collapsed;

AnswersListBox.Visibility = Visibility.Collapsed;

FeedbackTextBlock.Visibility = Visibility.Collapsed;

ActionButton.Visibility = Visibility.Collapsed;

TimerBox.Visibility = Visibility.Collapsed;

WeightTexBox.Visibility = Visibility.Collapsed;

ResultPanel.Visibility = Visibility.Visible;

FinalScoreText.Text =

$"Тест завершён!\n" +

$"Набрано баллов: {Score} из {totalWeight}\n" +

$"Процент: {totalScore}%";

// Запись в БД

try

{

using (var conn = new OleDbConnection(connStr))

{

conn.Open();

string sql = @"INSERT INTO результаты

(идентификатор\_пользователя, идентификатор\_теста, баллы)

VALUES (?, ?, ?)";

using (var cmd = new OleDbCommand(sql, conn))

{

cmd.Parameters.AddWithValue("?", currentUser.Id);

cmd.Parameters.AddWithValue("?", TestId);

cmd.Parameters.AddWithValue("?", totalScore);

cmd.ExecuteNonQuery();

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при сохранении результата:\n{ex.Message}",

"Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

private void OnPropertyChanged(string prop) =>

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(prop));

private void HomeButton\_Click(object sender, RoutedEventArgs e)

{

главная main = new главная();

main.Show();

this.Close();

}

private void LeaderboardButton\_Click(object sender, RoutedEventArgs e)

{

// Создайте окно таблицы лидеров

таблица\_лидеров leaderboard = new таблица\_лидеров(TestId);

leaderboard.Show();

this.Close();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Спорт\_тесты

{

public class Пользователь

{

public int Id { get; set; }

public string Name { get; set; }

public string Login { get; set; }

public string Password { get; set; }

}

}

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Спорт\_тесты

{

public class вопрос

{

public int идентификатор { get; set; }

public string содержание { get; set; }

public int вес { get; set; }

public ObservableCollection<ответ> ответы { get; set; } = new ObservableCollection<ответ>();

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Спорт\_тесты

{

public class ответ

{

public int идентификатор { get; set; }

public string содержание { get; set; }

public string комментарий { get; set; }

public bool правильный { get; set; }

private bool \_isSelected;

public bool IsSelected

{

get => \_isSelected;

set

{

if (\_isSelected == value) return;

\_isSelected = value;

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(nameof(IsSelected)));

}

}

public event PropertyChangedEventHandler PropertyChanged;

}

}

using System;

using System.Windows;

using System.Data.OleDb;

using System.IO;

using System.Text;

namespace Спорт\_тесты

{

public partial class авторизация : Window

{

public авторизация()

{

InitializeComponent();

}

private void LoginButton\_Click(object sender, RoutedEventArgs e)

{

string dataFolder = System.IO.Path.Combine(

Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData),

"SportTestsDB");

string dbPath = System.IO.Path.Combine(dataFolder, "db.accdb");

string connStr = $@"Provider=Microsoft.ACE.OLEDB.12.0;Data Source={dbPath};Persist Security Info=False;";

// Считываем введённые логин/пароль

string username = UsernameBox.Text.Trim();

string password = PasswordBox.Password;

// Если поля пустые — сразу ошибка

if (string.IsNullOrEmpty(username) || string.IsNullOrEmpty(password))

{

MessageBox.Show("Введите логин и пароль", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

return;

}

try

{

using (var conn = new OleDbConnection(connStr))

{

conn.Open();

string sql = @"

SELECT COUNT(\*)

FROM Пользователи

WHERE [имя\_пользователя] = ? AND [пароль] = ?

";

using (var cmd = new OleDbCommand(sql, conn))

{

cmd.Parameters.AddWithValue("?", username);

cmd.Parameters.AddWithValue("?", password);

int count = (int)cmd.ExecuteScalar();

if (count > 0)

{

MessageBox.Show("Успешный вход!", "Успех", MessageBoxButton.OK, MessageBoxImage.Information);

var selectCmd = new OleDbCommand(

"SELECT идентификатор, [имя], [имя\_пользователя] FROM пользователи WHERE [имя\_пользователя]=? AND [пароль]=?",

conn);

selectCmd.Parameters.AddWithValue("?", username);

selectCmd.Parameters.AddWithValue("?", password);

using (var reader = selectCmd.ExecuteReader())

{

if (reader.Read())

{

Пользователь user = new Пользователь

{

Id = reader.GetInt32(0),

Name = reader.GetString(1),

Login = reader.GetString(2)

};

// Сохраняем в Application

((App)Application.Current).CurrentUser = user;

}

}

главная main = new главная();

main.Show();

this.Close();

}

else

{

MessageBox.Show("Неверный логин или пароль!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при подключении к БД:\n{ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private void Registration\_Click(object sender, RoutedEventArgs e)

{

MessageBox.Show("Переход на форму регистрации", "Регистрация", MessageBoxButton.OK, MessageBoxImage.Information);

регистрация regWindow = new регистрация();

regWindow.Show();

this.Close();

}

}

}

using System.Data.OleDb;

using System;

using System.Windows;

namespace Спорт\_тесты

{

public partial class регистрация : Window

{

public регистрация()

{

InitializeComponent();

}

private void RegisterButton\_Click(object sender, RoutedEventArgs e)

{

string name = NameBox.Text.Trim();

string loginl = loginBox.Text.Trim();

string password = PasswordBox.Password;

string confirmPassword = ConfirmPasswordBox.Password;

// Валидация полей

if (string.IsNullOrWhiteSpace(name) ||

string.IsNullOrWhiteSpace(loginl) ||

string.IsNullOrWhiteSpace(password) ||

string.IsNullOrWhiteSpace(confirmPassword))

{

MessageBox.Show("Заполните все поля!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

return;

}

if (password != confirmPassword)

{

MessageBox.Show("Пароли не совпадают!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

return;

}

string dataFolder = System.IO.Path.Combine( Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData),

"SportTestsDB");

string dbPath = System.IO.Path.Combine(dataFolder, "db.accdb");

string connString = $@"Provider=Microsoft.ACE.OLEDB.12.0;Data Source={dbPath};Persist Security Info=False;";

// Составляем строку подключения к Access (ACE).

// Файл БД должен лежать рядом с exe и иметь Build Action=Content, Copy if newer.

/\*string dbPath = System.IO.Path.Combine(

AppDomain.CurrentDomain.BaseDirectory,

"db.accdb");

string connString = $@"Provider=Microsoft.ACE.OLEDB.12.0;Data Source={dbPath};Persist Security Info=False;";\*/

try

{

using (var conn = new OleDbConnection(connString))

{

conn.Open();

// Сначала проверим, нет ли уже такого логина/почты

using (var checkCmd = new OleDbCommand(

@"SELECT COUNT(\*)

FROM Пользователи

WHERE [имя\_пользователя] = ?", conn))

{

checkCmd.Parameters.AddWithValue("?", loginl);

int exists = (int)checkCmd.ExecuteScalar();

if (exists > 0)

{

MessageBox.Show("Пользователь с таким логином уже существует",

"Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

return;

}

}

// Вставляем нового пользователя

using (var insertCmd = new OleDbCommand(

@"INSERT INTO Пользователи ([имя], [имя\_пользователя], [пароль])

VALUES (?, ?, ?)", conn))

{

insertCmd.Parameters.AddWithValue("?", name);

insertCmd.Parameters.AddWithValue("?", loginl);

insertCmd.Parameters.AddWithValue("?", password);

int rows = insertCmd.ExecuteNonQuery();

if (rows > 0)

{

MessageBox.Show("Регистрация прошла успешно!", "Успех",

MessageBoxButton.OK, MessageBoxImage.Information);

авторизация auth = new авторизация();

auth.Show();

this.Close();

}

else

{

MessageBox.Show("Не удалось добавить пользователя", "Ошибка",

MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при работе с базой данных:\n{ex.Message}",

"Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private void BackToLogin\_Click(object sender, RoutedEventArgs e)

{

// Переход на форму авторизации

авторизация authWindow = new авторизация();

authWindow.Show();

this.Close();

}

}

}