МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Программной инженерии

Специальность 1-40 01 01 Программное обеспечение информационных технологий

Специализация Программирование интернет-приложений

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине «Объектно-ориентированное программирование»

Тема Программное средство составления и прохождения тестов

Исполнитель

Студент (ка) 2 курса группы 6 Савенко А.В.

(Ф.И.О.)

Руководитель Сапотько Р.А.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Руководитель Сапотько Р.А.

(подпись)

Минск 2018

Оглавление

[**Введение** 3](#_Toc515050994)

[1.Теоретическое введение 4](#_Toc515050995)

[1.1 Объектно-ориентированное программирование 4](#_Toc515050996)

[1.2 C# 4](#_Toc515050997)

[1.3 ADO 5](#_Toc515050998)

[1.4 Архитектура платформы .NET Framework 5](#_Toc515050999)

[2. Архитектура проекта 7](#_Toc515051000)

[2.1 Проектирование классов приложения 7](#_Toc515051001)

[2.2 Проектирование базы данных 9](#_Toc515051002)

[3. Руководство пользователя 10](#_Toc515051003)

[4. Тестирование 16](#_Toc515051004)

[Заключение 18](#_Toc515051005)

[Список использованных источников 19](#_Toc515051006)

[Приложение А 20](#_Toc515051007)

**Введение**

Цифровые технологии сильно изменили жизнь человека. Компьютер предоставляет большие возможности для работы с разного вида информацией.

В настоящее время существует большое количество программ, которые позволяют создавать и проходить тесты. Компьютер как средство для тестирования имеет свои достоинства и недостатки. Однако в объективности оценки ему отказать трудно. Видимо, именно этот фактор способствует расширению сферы применения компьютерного тестирования — оно успешно используется в школах, вузах, при поступлении на работу, при сдаче экзаменов в ГИБДД и т.д.

Сегодня разработчики предлагают разнообразные компьютерные инструменты для подготовки тестовых заданий, начиная от различных редакторов и программ для разработки презентаций и до использования языков программирования и возможностей сети Интернет. Но разработка качественного тестового инструментария – длительный, трудоемкий и дорогостоящий процесс.

Целью данного курсового проекта является разработка программного средства, позволяющего создавать и проходить тесты. В качестве интерфейса прикладного программирования был выбран обширный API-интерфейс – Windows Presentation Foundation (WPF), предназначенный для создания настольных программ с графически насыщенным пользовательским интерфейсом. В основе графической технологии WPF лежит мощная инфраструктура, основанная на DirectX. Это является одним из основных отличий WPF от более ранней технологии создания пользовательских интерфейсов – Windows Forms.

Для работы с WPF использовался объектно-ориентированный язык программирования с С-подобным синтаксисом – С#, разработанный для создания приложений на платформе Microsoft .NET Framework. Кроме того, используемая в WPF модель разделения кода и дизайна предполагала работу с декларативным языком описания интерфейса –eXtensible Application Markup Language (XAML).

# 1.Теоретическое введение

## 1.1 Объектно-ориентированное программирование

Объектно-ориентированное программирование — это метод программирования, при использовании которого главными элементами программы являются понятия объектов и классов.

ООП возникло в результате развития идеологии процедурного программирования, где данные и подпрограммы (процедуры, функции) их обработки формально не связаны. В центре ООП находится понятие объекта.

**Объект** — это сущность, экземпляр класса, в которой можно посылать сообщения, и которая может на них реагировать, используя свои данные. Данные объекта скрыты от остальной программы. Сокрытие данных называется инкапсуляцией.

Наличие инкапсуляции достаточно для объектности языка программирования, но ещё не означает его объектной ориентированности — для этого требуется наличие наследования.

Но даже наличие инкапсуляции и наследования не делает язык программирования в полной мере объектным с точки зрения ООП. Основные преимущества ООП проявляются только в том случае, когда в языке программирования реализован полиморфизм; то есть возможность объектов с одинаковой спецификацией иметь различную реализацию.

**Абстрагирование** — это способ выделить набор значимых характеристик объекта, исключая из рассмотрения не значимые.  **Инкапсуляция** — это свойство системы, позволяющее объединить данные и методы, работающие с ними в классе, и скрыть детали реализации от пользователя. **Наследование** — это свойство системы, позволяющее описать новый класс на основе уже существующего с частично или полностью заимствующейся функциональностью. Класс, от которого производится наследование, называется базовым, родительским или суперклассом. Новый класс — потомком, наследником или производным классом. **Полиморфизм** — это свойство системы использовать объекты с одинаковым интерфейсом без информации о типе и внутренней структуре объекта.

Первым языком программирования, в котором были предложены принципы объектной ориентированности, была Симула.

## 1.2 C#

C# – язык программирования, сочетающий объектно-ориентированные и контекстно-ориентированные концепции. Разработан в 1998-2001 годах группой инженеров под руководством Андерсa Хейлсбергa в компании Microsoft как основной язык разработки приложений для платформы Microsoft .NET. Компилятор с C# входит в стандартную установку самой .NET, поэтому программы на нём можно создавать и компилировать даже без инструментальных средств вроде Visual Studio.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет строгую статическую типизацию, поддерживает полиморфизм, перегрузку операторов, указатели на функции-члены классов, атрибуты, события, свойства, исключения, комментарии в формате XML. Переняв многое от своих предшественников – языков C++, Delphi, Modula и Smalltalk – С#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем: так, C# не поддерживает множественное наследование классов (в отличие от C++).

## 1.3 [ADO](javascript:void(0))

Сегодня большое значение имеет работа с данными. Для хранения данных используются различные системы управления базами данных: MS SQL Server, Oracle, MySQL и так далее. И большинство крупных приложений так или иначе используют для хранения данных эти системы управления базами данных. Однако, чтобы осуществлять связь между базой данных и приложением на C# необходим посредник. И именно таким посредником является технология ADO.NET.

ADO.NET предоставляет собой технологию работы с данными, которая основана на платформе .NET Framework. Эта технология представляет нам набор классов, через которые мы можем отправлять запросы к базам данных, устанавливать подключения, получать ответ от базы данных и производить ряд других операций.

Основу интерфейса взаимодействия с базами данных в ADO.NET представляет ограниченный круг объектов: Connection, Command, DataReader, DataSet и DataAdapter. С помощью объекта Connection происходит установка подключения к источнику данных. Объект Command позволяет выполнять операции с данными из БД. Объект DataReader считывает полученные в результате запроса данные. Объект DataSet предназначен для хранения данных из БД и позволяет работать с ними независимо от БД. И объект DataAdapter является посредником между DataSet и источником данных. Главным образом, через эти объекты и будет идти работа с базой данных.

## 1.4 [Архитектура платформы .NET Framework](javascript:void(0))

Программа на языке C# выполняется в среде .NET Framework – интегрированном компоненте Windows, содержащем виртуальную систему выполнения (среда CLR) и унифицированный набор библиотек классов. Среда CLR представляет собой коммерческую реализацию Майкрософт инфраструктуры CLI (common language infrastructure), международного стандарта, основы сред выполнения и разработки с тесным взаимодействием языков и библиотек.

Исходный код, написанный на языке C#, компилируется в промежуточный язык (IL) в соответствии со спецификацией CLI. Код IL и ресурсы, такие как растровые изображения и строки, хранятся на диске в исполняемом файле, называемом сборкой, с расширением EXE или DLL в большинстве случаев. Сборка содержит манифест со сведениями о типах сборки, версии, языке и региональных параметрах и требованиях безопасности.

При выполнении программы на C# сборка загружается в среду CLR в зависимости от сведений в манифесте. Далее, если требования безопасности соблюдены, среда CLR выполняет JIT-компиляцию для преобразования кода IL в инструкции машинного кода. Среда CLR также предоставляет другие службы, относящиеся к автоматическому сбору мусора, обработке исключений и управлению ресурсами. Код, выполняемый средой CLR, иногда называют "управляемым кодом" в противопоставление "неуправляемому коду", который компилируется в машинный код, предназначенный для определенной системы. Далее показаны отношения во время компиляции и время выполнения между файлами с исходным кодом C#, библиотеками классов .NET Framework, сборками и средой CLR.

Взаимодействие между языками является ключевой особенностью .NET Framework. Поскольку код на промежуточном языке (IL), создаваемый компилятором C#, соответствует спецификации CTS, код IL на основе C# может взаимодействовать с кодом, создаваемым версиями языков Visual Basic, Visual C++, Visual J# платформы .NET Framework и еще более чем 20 CTS-совместимых языков. В одной сборке может быть несколько модулей, написанных на разных языках платформы .NET Framework, и типы могут ссылаться друг на друга, как если бы они были написаны на одном языке.

При разработке приложения были использованы:

* интегрированная среда разработки Microsoft Visual Studio 2017;
* программная платформа .NET Framework 5.0;
* язык программирования C#;
* расширяемый язык разметки XAML;
* технология WPF;
* ADO.NET;
* WpfAnimatedGif Library;

# 2. Архитектура проекта

## 2.1 Проектирование классов приложения

Диаграмма классов приложения представлена на рисунке 2.1.

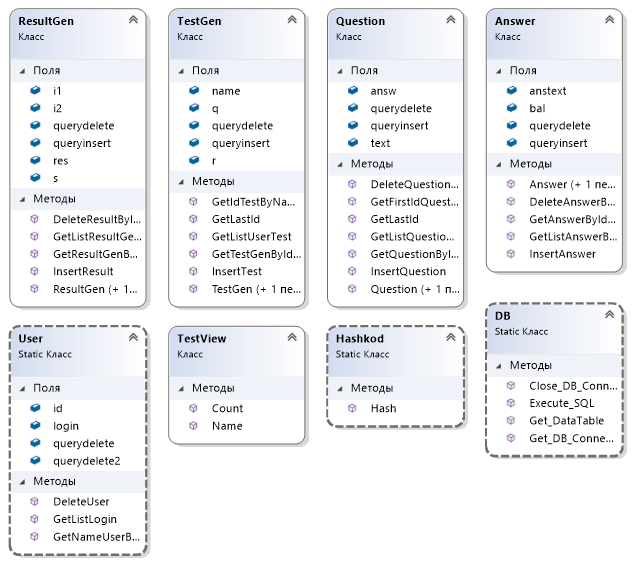


Рисунок 2.1 – Диаграмма классов

Класс DB содержит методы работы с базой данных и данные о текущем подключении к БД.

В классе Hashkod хранится метод для хеширования пароля.

Класс User содержит информацию об одном пользователе, под именем которого осуществляется вход в приложение. Также этот класс хранит метод для удаления пользователя и ещё несколько методов.

TestGen – это класс для работы с тестами. В нём хранятся методы, такие как вставка теста, получение ID теста по его имени и т.д. Также хранятся данные, необходимые для взаимодействия с записями из таблицы, хранящей данные о тестах. Внутри каждого экземпляра этого класса находятся списки Question и ResultGen.

Question – это класс, содержащий информацию и методы, связанные с получением вопросов. Внутри каждого экземпляра лежит список Answer.

Класс Answer хранит данные и методы получения текстовых переменных, содержащие данные об ответе на определённый вопрос, и количество баллов, полученных за выбор данного ответа.

В классе ResultGen находятся данные и методы получения результатов тестов в зависимости от набранного количества баллов, полученных в результате прохождения всего теста.

Класс TestView содержит два метода: подсчёт пользовательских тестов и имена этих тестов.

Также в проекте есть 11 классов окон. Классы окон представлены на рисунке 2.2.

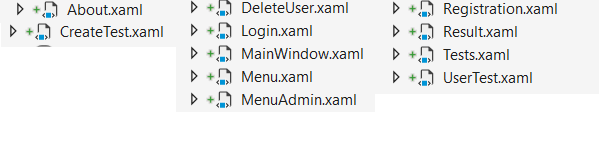


Рисунок 2.2 – Классы окон

Все файлы логических ресурсов подключаются для использования в разметке XAML в файле App.xaml в качестве логических с помощью разметки, представленной в Приложении А (Листинг 1).

Также в Приложении А(Листинг 2) представлены стили, использованные в данном курсовом проекте.

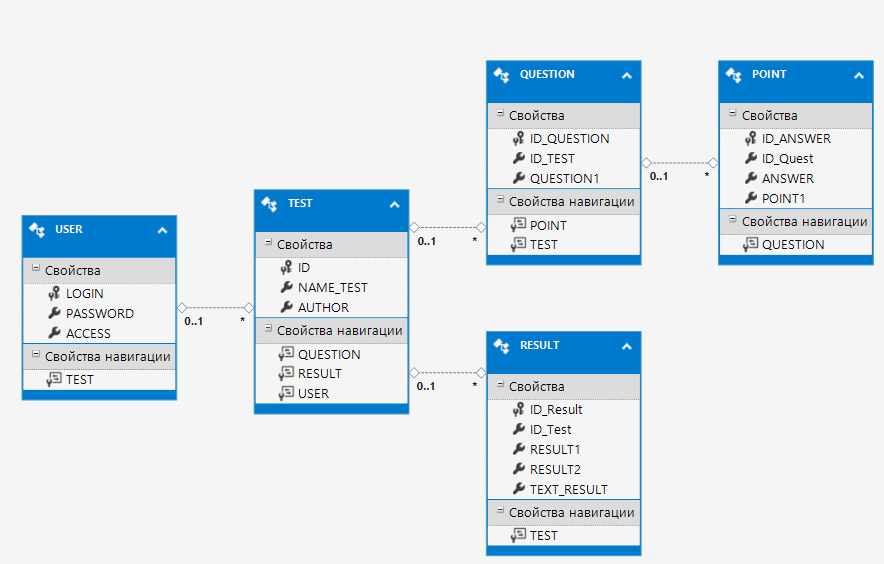
Стартовым окном программы является «Login». В нём пользователь может ввести входные данные и, в случае успешной авторизации, перейти к окну «Menu». Также пользователь может перейти к окну «Registration» по нажатию соответствующей кнопки, если у него ещё нет своего аккаунта. Так, оказавшись в окне регистрации пользователю нужно ввести необходимые данные, и подтвердить регистрацию нажатием соответствующей кнопки. В случае успешной регистрации пользователь автоматически переходит к окну «Menu».

К основным функциональным окнам также относятся окна:

* «CreateTest»,
* «Tests»,
* «UserTest»,
* «GeleteUser»,
* «About».

## 2.2 Проектирование базы данных

Диаграмма базы данных, спроектированной в ходе разработки приложения, представлена на рисунке 2.3.

Рисунок 2.3 – Диаграмма базы данных

База данных приложения состоит из 5 таблиц, взаимосвязанных друг с другом внешними ключами. Главной из данных таблиц является таблица «USER», с которой так или иначе связана каждая таблица базы данных. Эта таблица содержит всех пользователей. С ней напрямую связана таблица «TEST», содержащая информацию о названии теста и его создателе. Далее создаются таблицы «QUESTION» и «RESULT», связанные с таблицей «TEST». В таблице «QUESTION» хранятся вопросы и ID тестов, которым они принадлежат. В таблице «RESULT» находятся текстовый результат, границы цифровых значений, которым соответствует тот или иной результат и ID теста. С таблицей «QUESTION» напрямую связана таблица «POINT», хранящая ID вопроса, ответы на него и баллы за эти ответы.

# 3. Руководство пользователя

Начальной страницей приложения является окно авторизации, представленное на рисунке 3.1.

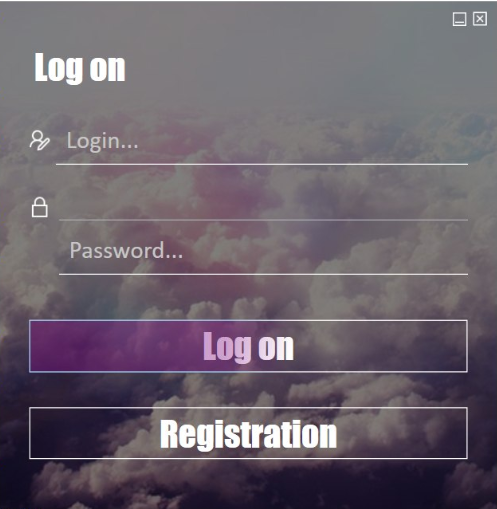


Рисунок 3.1 – Начальная страница

Если пользователь ещё не авторизован, то следует нажать кнопку «Registration» и откроется окно, представленное на рисунке 3.2. После заполнения всех полей и нажатия кнопки «Create» система перекинет пользователя на окно регистрации, представленное на рисунке 3.1.

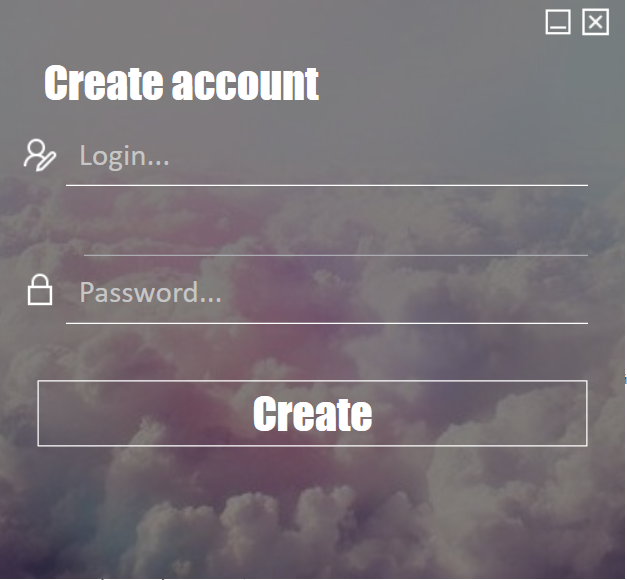


Рисунок 3.2 – Создание аккаунта

Далее, в зависимости от того, кто входит в приложение (пользователь или администратор), открывается соответствующее меню (Рис. 3.3– 3.4).

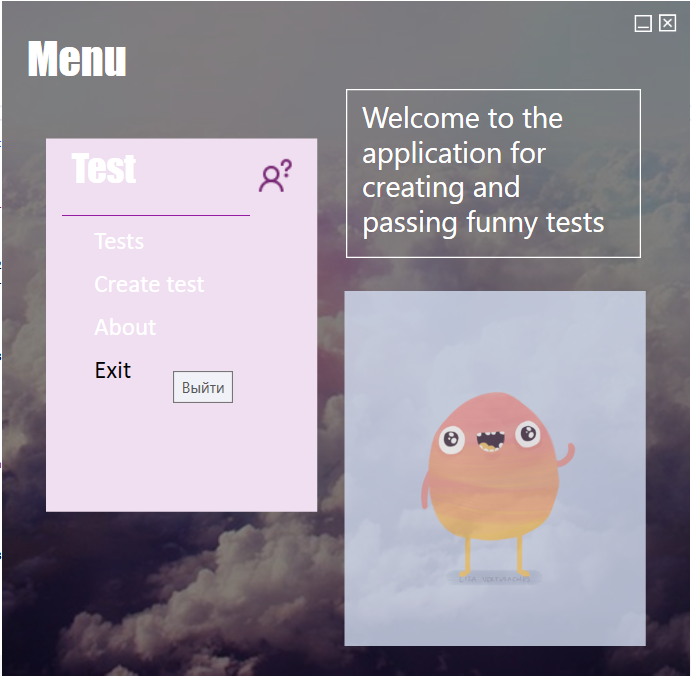


Рисунок 3.3 – Меню пользователя

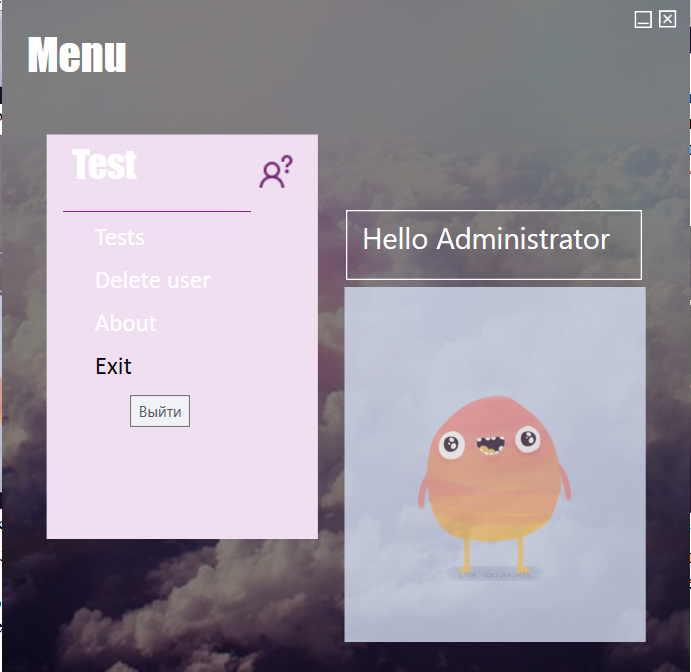


Рисунок 3.4 – Меню администратора

Различие меню состоит в том, что пользователь может создавать и проходить тесты. Администратор может удалять пользователей и созданные ими тесты. Далее подробно описано содержание каждого пункта меню.

Рассмотрим меню пользователя. Оно содержит в себе такие вкладки, как: тесты, создание теста, о программе и выход.

Для прохождения теста следует перейти на соответствующую вкладку(«Tests») в меню. Откроется окно, представленное на рисунке 3.5.

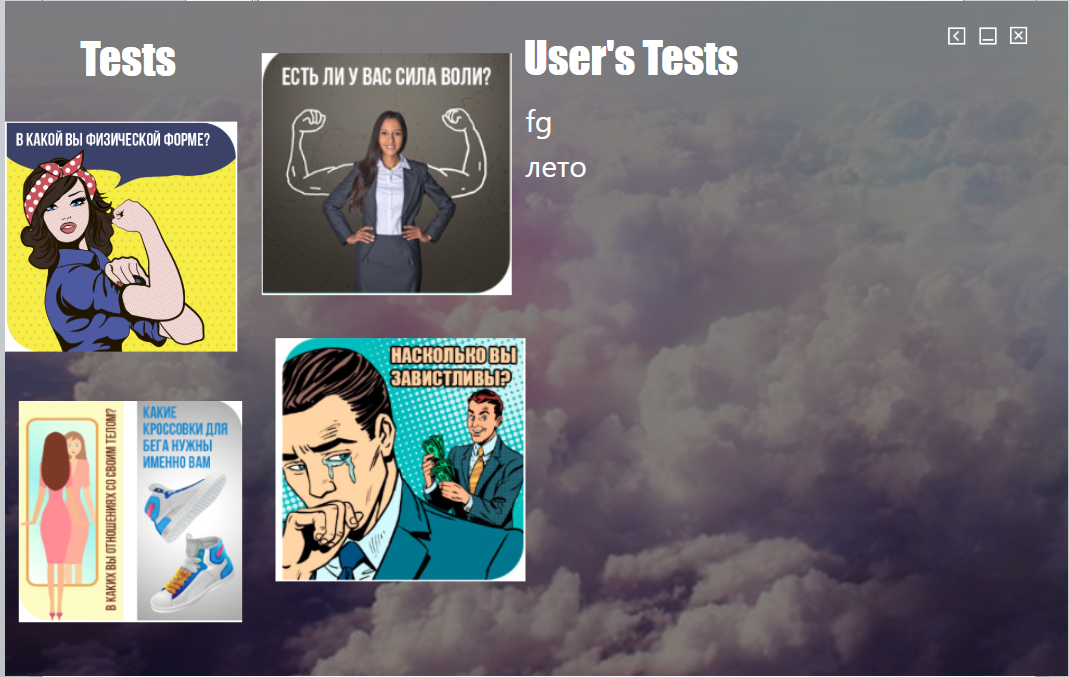


Рисунок 3.5 – Страница с тестами

Оно разделено на две части. С левой стороны есть 4 теста, созданных администратором для того, чтобы пользователь мог по образцу создавать свои тесты. С правой стороны список тестов, созданных пользователями. При выборе одного из тестов администратора или пользователей и нажатия на картинку откроется само поле теста (Рис.3.6).

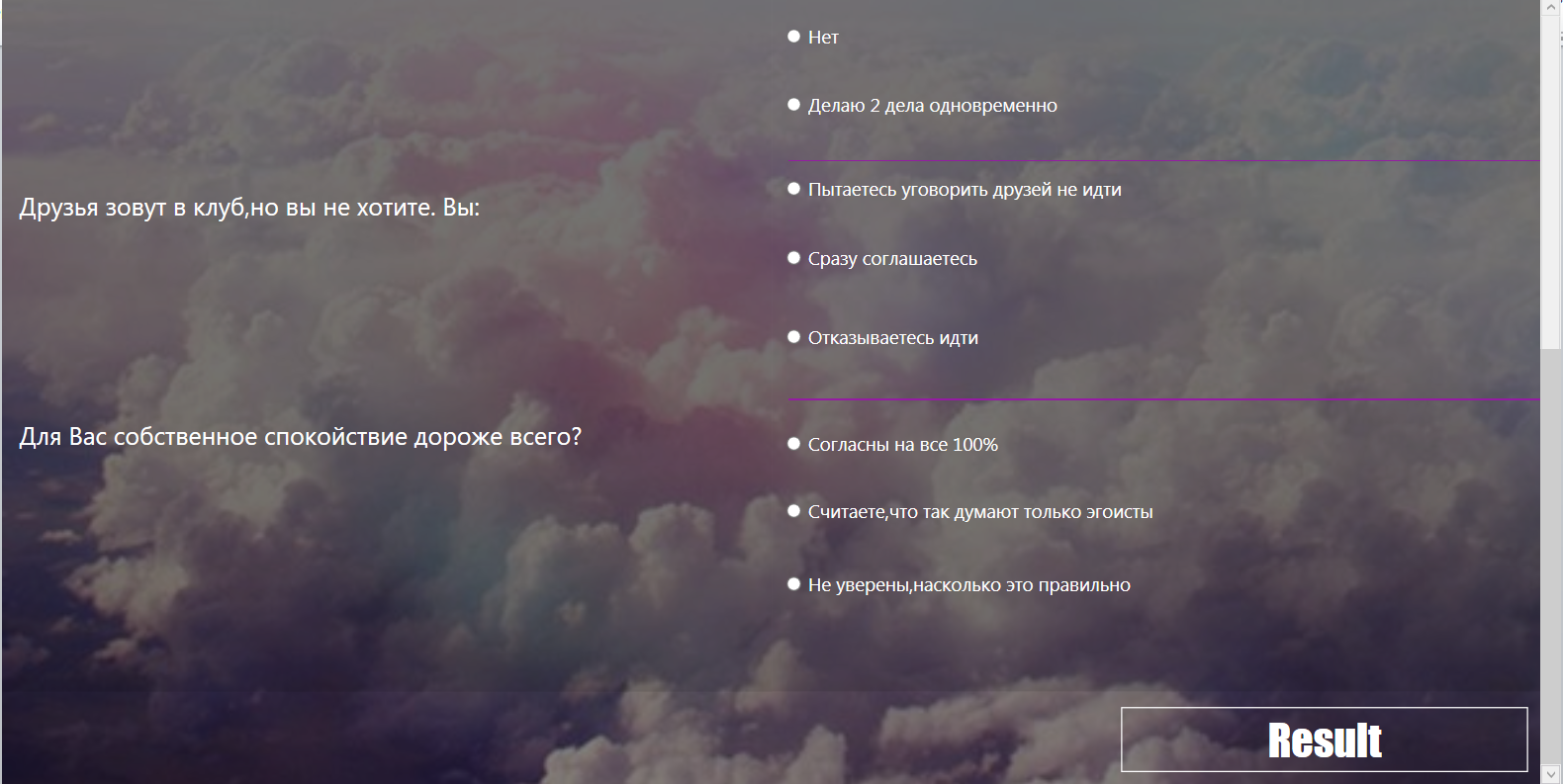


Рисунок 3.6 – Страница прохождения теста

После того, как пользователь ответит на все вопросы, следует нажать кнопку «Result». Появляется окно с результатом пройденного теста (Рис. 3.8).

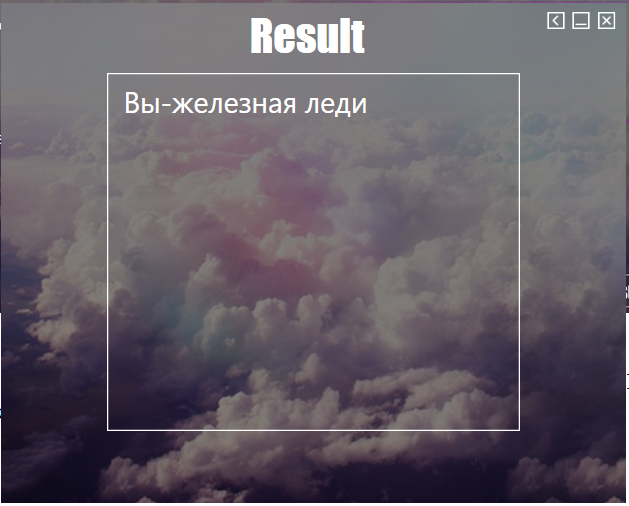


Рисунок 3.8 – Результат теста

Для создания теста необходимо открыть вкладку «Create test» в главном меню. Откроется страница для создания нового теста (Рис. 3.9).

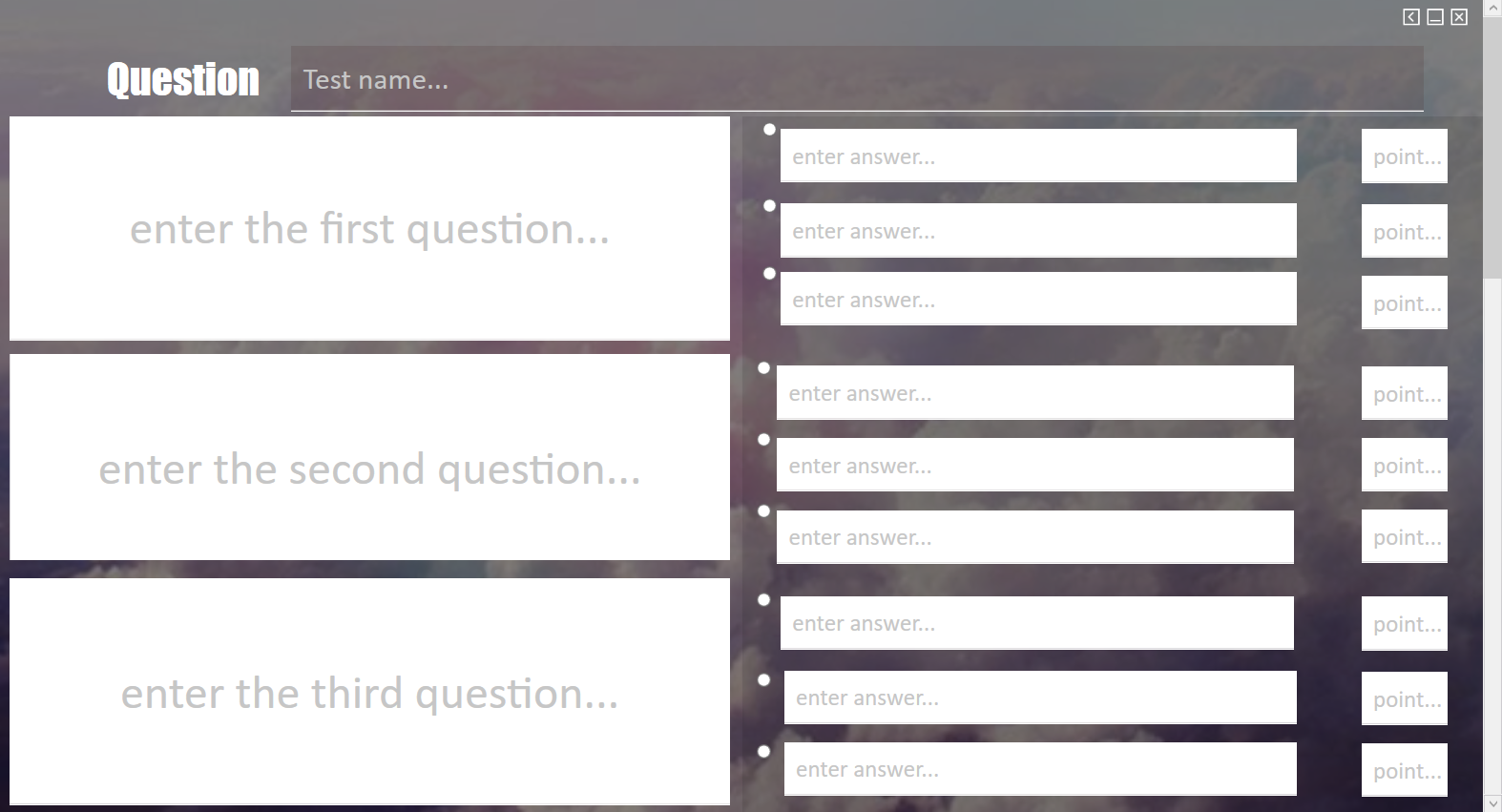


Рисунок 3.9 – Создание теста

В самом верху есть поле, предназначенное для ввода названия теста. Ниже располагаются 5 пустых полей, предназначенных для ввода вопросов, а напротив – по 3 поля для ввода вариантов ответа и баллов, которые даются за каждый ответ.

В конце страницы есть поля для добавления текстового результата и две кнопки. Одна для добавления текстовых результатов(«Add result»), а другая для создания теста(«Create») (Рис. 3.10).

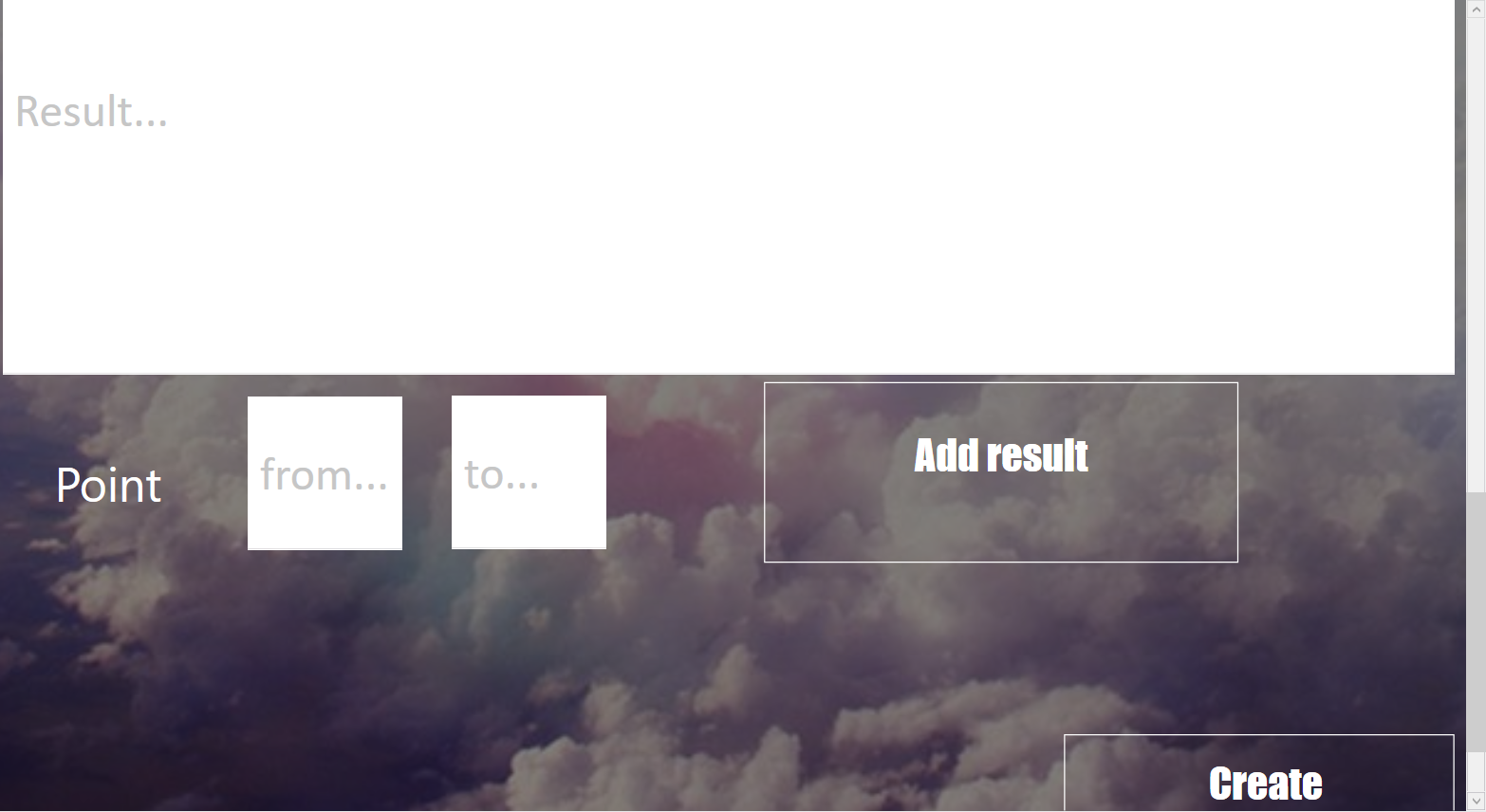


Рисунок 3.10 – Создание и добавление теста

Если пользователь забудет заполнить одно из полей и нажмёт кнопку «Add result» или «Create», система выдаст предупреждение, представленное на рисунке 3.11.

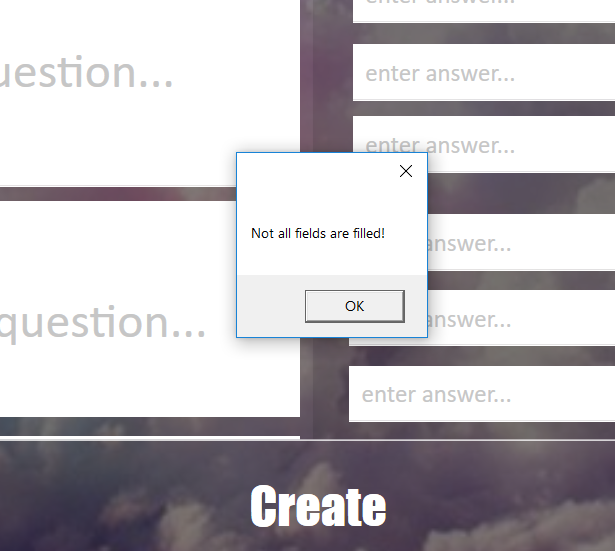


Рисунок 3.11 – Проверка на пустые поля

При выборе пункта «About» главного меню откроется окно, представленное на рисунке 3.12.



Рисунок 3.12 – О программе

На этом обзор пользовательского меню завершён.

При входе в меню администратора добавляется одна новая вкладка «Delete user». При нажатии на неё откроется окно со списком пользователей и кнопкой удалить(«Delete») (Рис. 3.13).

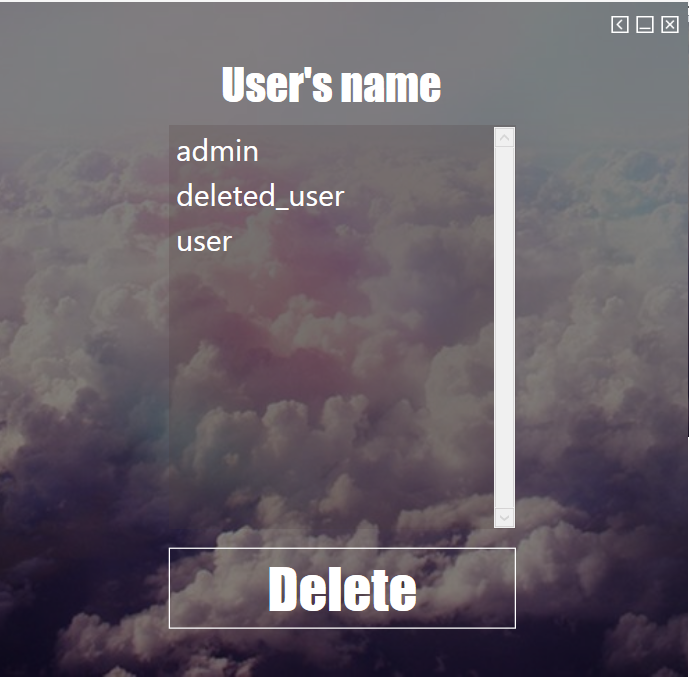


Рисунок 3.13 – Удаление пользователей

Зарезервированных пользователей, таких как администратор, удалить нельзя (Рис. 3.14).

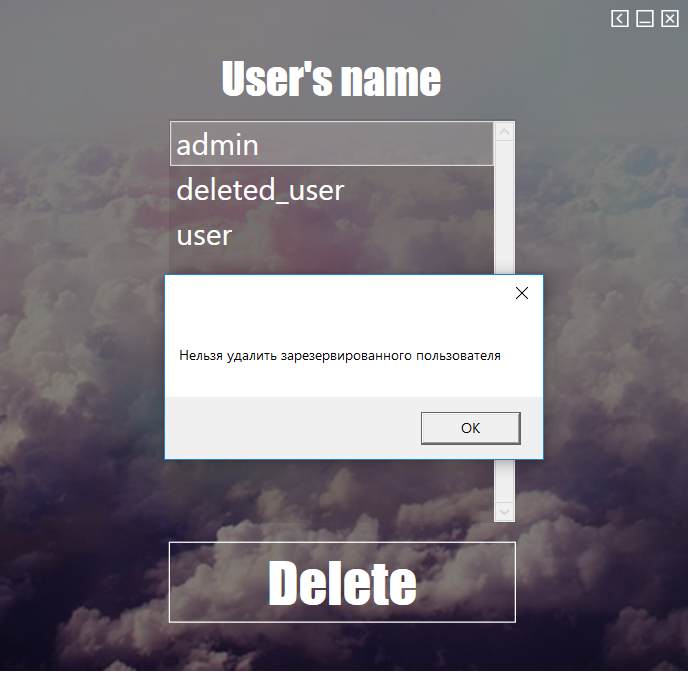


Рисунок 3.14 – Удаление зарезервированных пользователей

При удалении пользователя, вместе с ним автоматически удаляются и тесты, созданные этим пользователем.

# 4. Тестирование

При тестировании данного приложения были проведены следующие тесты на ситуации, которые могли бы привести к ошибке:

* тесты авторизации и регистрации;
* тест ввода корректных данных;
* тест изменения данных;

В момент регистрации возможна ситуация, когда пользователь вводит недостаточной безопасности пароль. Обработка данного исключения продемонстрирована на рисунке 4.1.

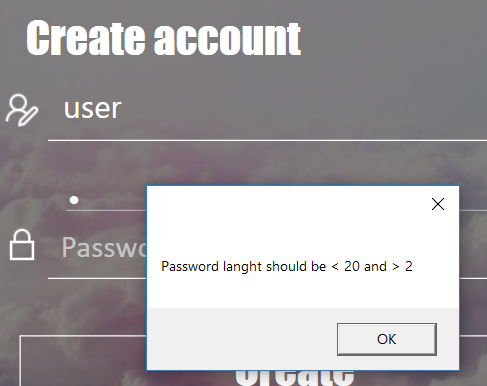


Рисунок 4.1 – Тестирование окна регистрации

В момент авторизации возможна ситуация, когда пользователь вводит неверные данные пользователя. Обработка данного исключения продемонстрирована на рисунке 4.2.

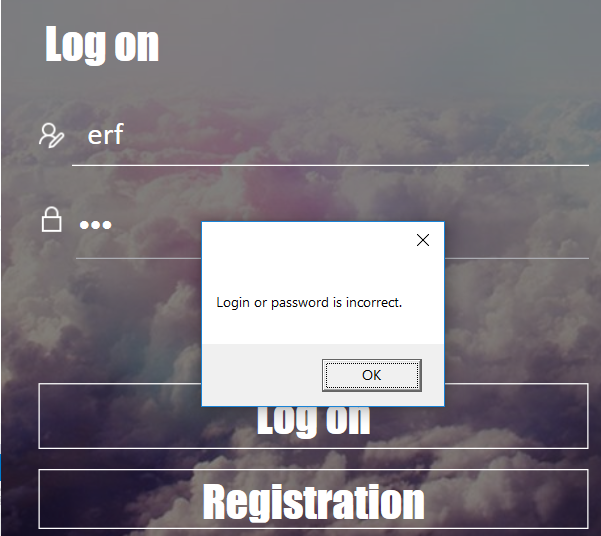


Рисунок 4.2 – Тестирование окна регистрации

Если пользователь при добавлении нового теста забыл заполнить хотя бы одно поле, то система сообщит ему об этом (рис. 4.3).

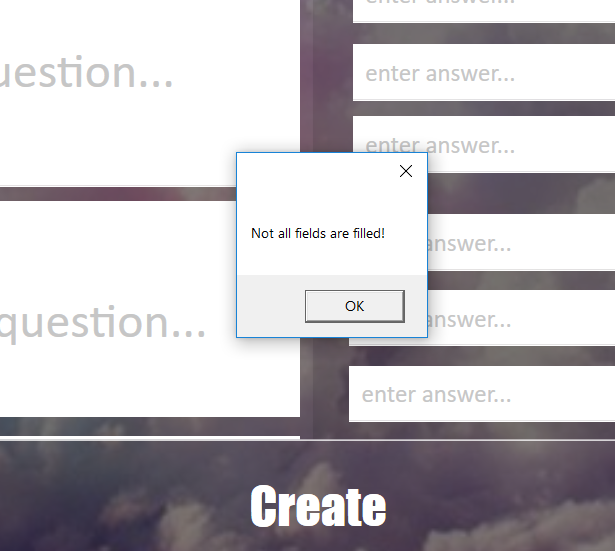


Рисунок 4.3 – Тестирование окна создания теста

Если пользователь забыл заполнить хотя бы одно поле при прохождении теста, то система выдаст ему предупреждение, представленное на рисунке 4.4.

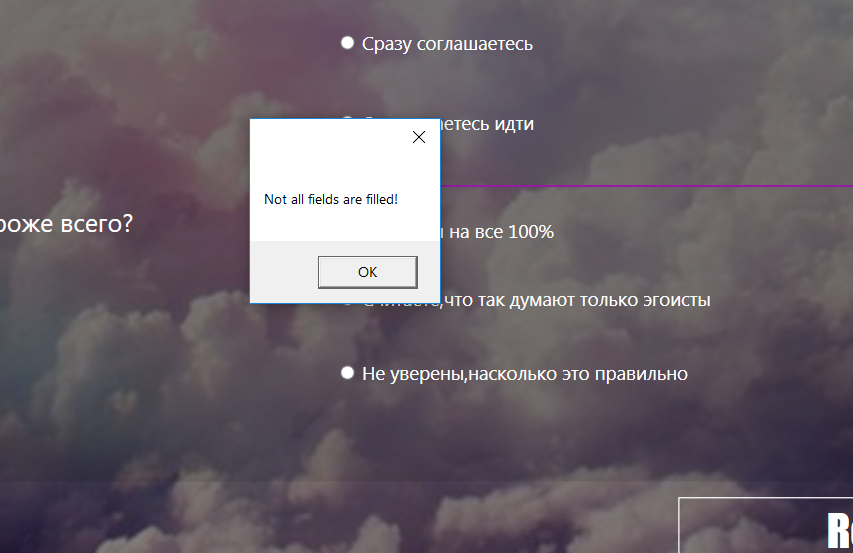


Рисунок 4.4 – Тестирование окна прохождения теста

# Заключение

В ходе выполнения курсовой работы было разработано приложение «Test», являющееся помощником в составлении и прохождении тестов. Во время выполнения данной курсовой работы было изучено немало теоретического материала, а также просмотрено и разобрано большое количество уже готовых решений тех или иных задач.

В ходе выполнения курсовой работы было выполнено следующее:

- разработка базы данных для хранения информации;

- разработка архитектуры приложения;

- разработка функциональных части приложения;

- разработка пользовательского интерфейса;

- написание исходного кода приложения;

- тестирование приложения.

Разработка приложения­ – трудоемкое занятие, выполнение которого требует всестороннего знания той предметной области, к которой относится тема разрабатываемого приложения. Программная система имеет простой, интуитивно понятный интерфейс пользователя.

В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает верно, а требования технического задания выполнены в полном объеме.

# Список использованных источников

1. Пацей, Н.В. Курс лекций по языку программирования C++ / Н.В. Пацей. – Минск: БГТУ, 2016. – 175 с.

2. Пацей, Н.В. Технология разработки программного обеспечения / Н.В. Пацей. – Минск: БГТУ, 2016. – 129 с.

3. MSDN сеть разработчиков в Microsoft [Электронный ресурс] / Режим доступа: <http://msdn.microsoft.com/library/rus/> – Дата доступа: 31.05.2017.

4. METANIT.COM Сайт о программировании[Электронный ресурс] / Режим доступа: <https://metanit.com> – Дата доступа: 31.05.2017.

5. ProfessorWeb .NET & Web Programming [Электронный ресурс] / Режим доступа: <https://professorweb.ru> – Дата доступа: 31.05.2017.

# Приложение А

|  |
| --- |
| <Application.Resources>  <ResourceDictionary>  <ResourceDictionary.MergedDictionaries>  <ResourceDictionary Source="Style/Texts.xaml"/>  <ResourceDictionary Source="Style/Butt.xaml"/>  </ResourceDictionary.MergedDictionaries>  </ResourceDictionary>  </Application.Resources> |

Листинг 1 — Добавление словаря логических ресурсов

|  |
| --- |
| <Style x:Key="zzz" TargetType="{x:Type Button}" >  <Setter Property="Template">  <Setter.Value>  <ControlTemplate TargetType="Button" >  <Grid>  <Label Content="Create" Foreground="White" HorizontalAlignment="Center" VerticalAlignment="Center" FontFamily="Impact" Width="126"/>  <Rectangle Stroke="White" />  <Rectangle Stroke="{x:Static SystemColors.HighlightBrush }"  Fill="#841284" >  <Rectangle.OpacityMask>  <LinearGradientBrush x:Name="HighlightGradient"  StartPoint="0,0" EndPoint="1,1" SpreadMethod="Reflect" Opacity="0.4" >  <GradientStop Color="White" Offset="0"/>  <GradientStop Color="Transparent" Offset="0"/>  </LinearGradientBrush>  </Rectangle.OpacityMask>  </Rectangle>  <Path Stroke="White"/>  </Grid>  <ControlTemplate.Triggers>  <Trigger Property="IsMouseOver" Value="True">  <Trigger.EnterActions>  <BeginStoryboard>  <Storyboard>  <DoubleAnimation  Storyboard.TargetName="HighlightGradient"  Storyboard.TargetProperty="GradientStops[0].Offset"  To="1" Duration="0:0:0.4">  </DoubleAnimation>  <DoubleAnimation  Storyboard.TargetName="HighlightGradient"  Storyboard.TargetProperty="GradientStops[1].Offset"  To="1" Duration="0:0:0.3"/>  </Storyboard>  </BeginStoryboard>  </Trigger.EnterActions>  <Trigger.ExitActions>  <BeginStoryboard>  <Storyboard>  <DoubleAnimation  Storyboard.TargetName="HighlightGradient"  Storyboard.TargetProperty="GradientStops[0].Offset"  To="0" Duration="0:0:0.2">  </DoubleAnimation>  <DoubleAnimation  Storyboard.TargetName="HighlightGradient"  Storyboard.TargetProperty="GradientStops[1].Offset"  To="0" Duration="0:0:0.2"/>  </Storyboard>  </BeginStoryboard>  </Trigger.ExitActions>  </Trigger>  </ControlTemplate.Triggers>  </ControlTemplate>  </Setter.Value>  </Setter>  </Style> |

Листинг 2 — Стиль Butt для кнопок