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 1-40 01 01 Программное обеспечение информационных технологий (программирование интернет-приложений)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине «Объектно-ориентированные технологии программирования и стандарты проектирования»

Тема\_\_\_\_\_\_\_\_\_\_\_\_Программное средство для тестирования «CWTester»

Исполнитель

студентка 2 курса 4 группы Рудаковский Степан Дмитриевич

(Ф. И. О.)

Руководитель работы ассистент Пахолко А.С.

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

Курсовой проект защищен с оценкой

Председатель Пацей Н.В.

(подпись)

Минск 2022

**Содержание**

[**ВВЕДЕНИЕ** 3](#_Toc104154756)

[**1 Аналитический обзор литературы и постановка задачи** 4](#_Toc104154757)

[1.1 Анализ прототипов 4](#_Toc104154758)

[1.2 Постановка задачи 6](#_Toc104154759)

[**2 Анализ требований к программному средству и разработка функциональных требований** 7](#_Toc104154760)

[2.1 Определение требований к программному стредству 7](#_Toc104154761)

[2.2 Описание средств разработки 7](#_Toc104154762)

[2.3 Описание функциональности программного средства 8](#_Toc104154763)

[**3. Проектирование и создание программного средства** 9](#_Toc104154764)

[3.1 Архитектура системы 9](#_Toc104154765)

[3.2 Проектирование логической структуры базы данных 12](#_Toc104154766)

[3.3 Проектирование доступа к базе данных 14](#_Toc104154767)

[4 Реализация программного средства 15](#_Toc104154768)

[4.1 Реализация MVVM 15](#_Toc104154769)

[4.2 Реализация авторизации и регистрации пользователей 16](#_Toc104154770)

[4.3 Выбор теста для прохождения и его запуск 18](#_Toc104154771)

[4.4 Переход по вопросам и подсчет результата после ответа на последний вопрос 19](#_Toc104154772)

[4.5 Просмотр собственных результатов 20](#_Toc104154773)

[4.6 Создание теста 21](#_Toc104154774)

[4.7 Очистка истории прохождения тестов администратором 23](#_Toc104154775)

[**5 Тестирование** 24](#_Toc104154776)

[**6 Руководство по использованию** 28](#_Toc104154777)

[6.1 Регистрация и авторизация 28](#_Toc104154778)

[6.2 Использование приложения обычным пользователем 29](#_Toc104154779)

[6.3 Использование приложения администратором 30](#_Toc104154780)

[**ЗАКЛЮЧЕНИЕ** 32](#_Toc104154781)

[**СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ** 33](#_Toc104154782)

[Приложение А 34](#_Toc104154783)

[Приложение Б 35](#_Toc104154784)

[Приложение В 36](#_Toc104154785)

[Приложение Г 37](#_Toc104154786)

# **ВВЕДЕНИЕ**

Различные средства тестирования всегда были популярны, однако в связи с пандемией нужда в них возросла еще сильнее, потому что это сильно упрощает дистанционный контроль знаний студентов, позволяет осуществлять его быстро и с моментальным получением оценки знаний.

Также часто средства тестирования используются простыми обывателями, в целью проверки своих знаний в определенной теме или к примеру для соревнований с другими людьми в эрудиции.

Часто возможность тестирования встраивают в более масштабные программные продукты, что говорит о популярности подобных программных средств и в связи с этим я выбрал данную тему.

Данное программное средство создано для предоставления удобного интерфейса пользователям, которые хотят осуществлять тестирование и принимать в нем участие.

Для успешной реализации курсового проекта необходимо:

* провести анализ соответствующей литературы;
* ознакомиться с прототипами программных средств выбранной мной темы;
* определить функциональные требования;
* продумать структуру базы данных;
* продумать структуру проекта;
* реализовать программное средство;
* протестировать программное средство;
* написать руководство пользователя.

Содержание данной пояснительной записки отражает этапы выполнения курсового проекта.

# **1 Аналитический обзор литературы и постановка задачи**

Для того чтобы окончательно определиться с постановкой задачи курсового проекта, необходимо проанализировать прототипы программных средств выбранной темы.

## 1.1 Анализ прототипов

moeobrazovanie.ru ­– платформа, на которой предоставлен широкий выбор тестов, созданных другими пользователями(рисунок 1.1).

Достоинства:

* большой выбор тестов
* статистика по каждому вопросу

Недостатки:

* для Windows существует только веб-версия

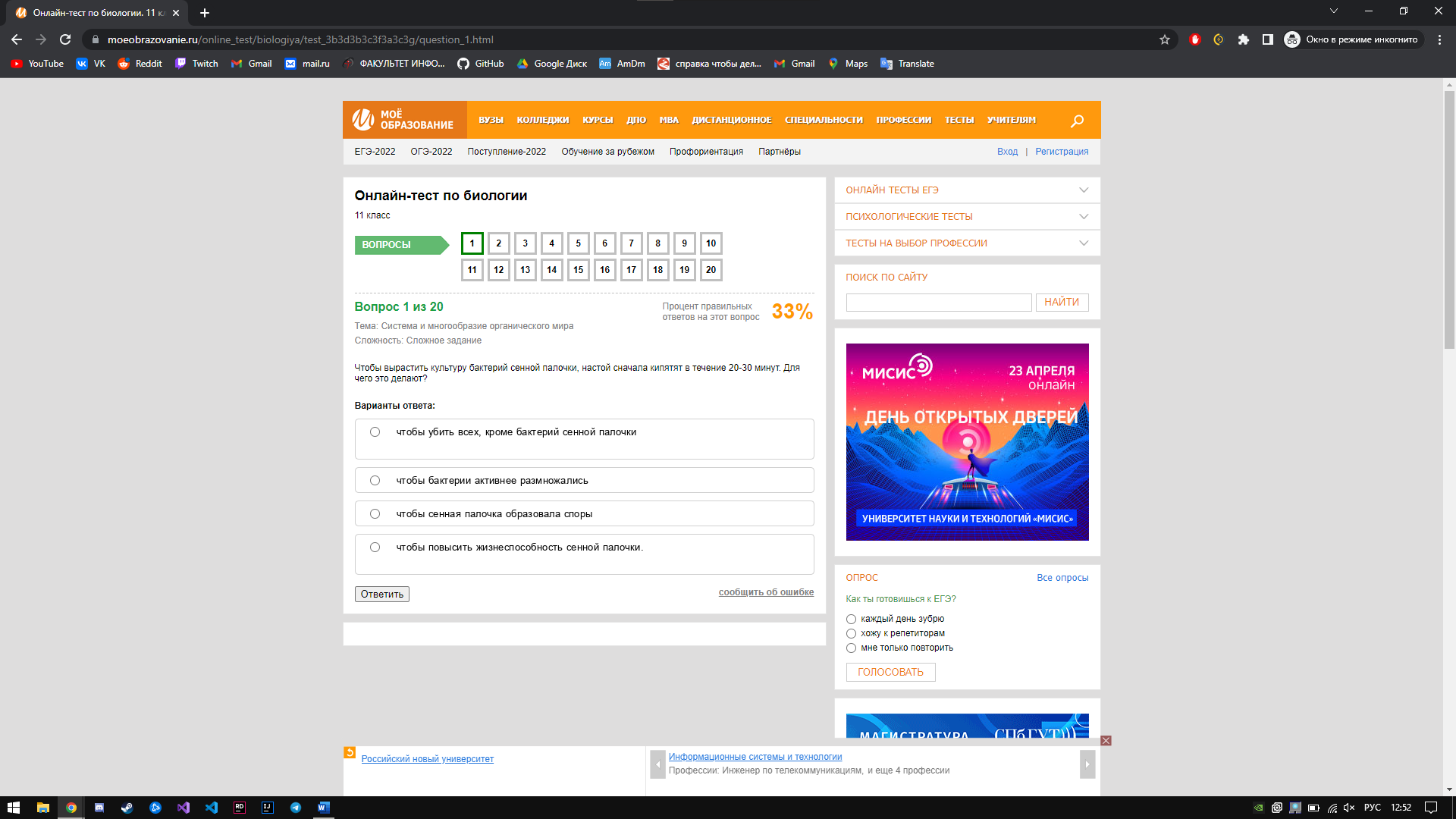


Рисунок 1.1 – Интерфейс веб-приложения «moeobrazovanie.ru»

ustaliy.ru – платформа для тестирования(рисунок 1.2).

Достоинства:

* широкий выбор тестов
* приятный интерфейс

Недостатки:

* для Windows существует только веб-приложение

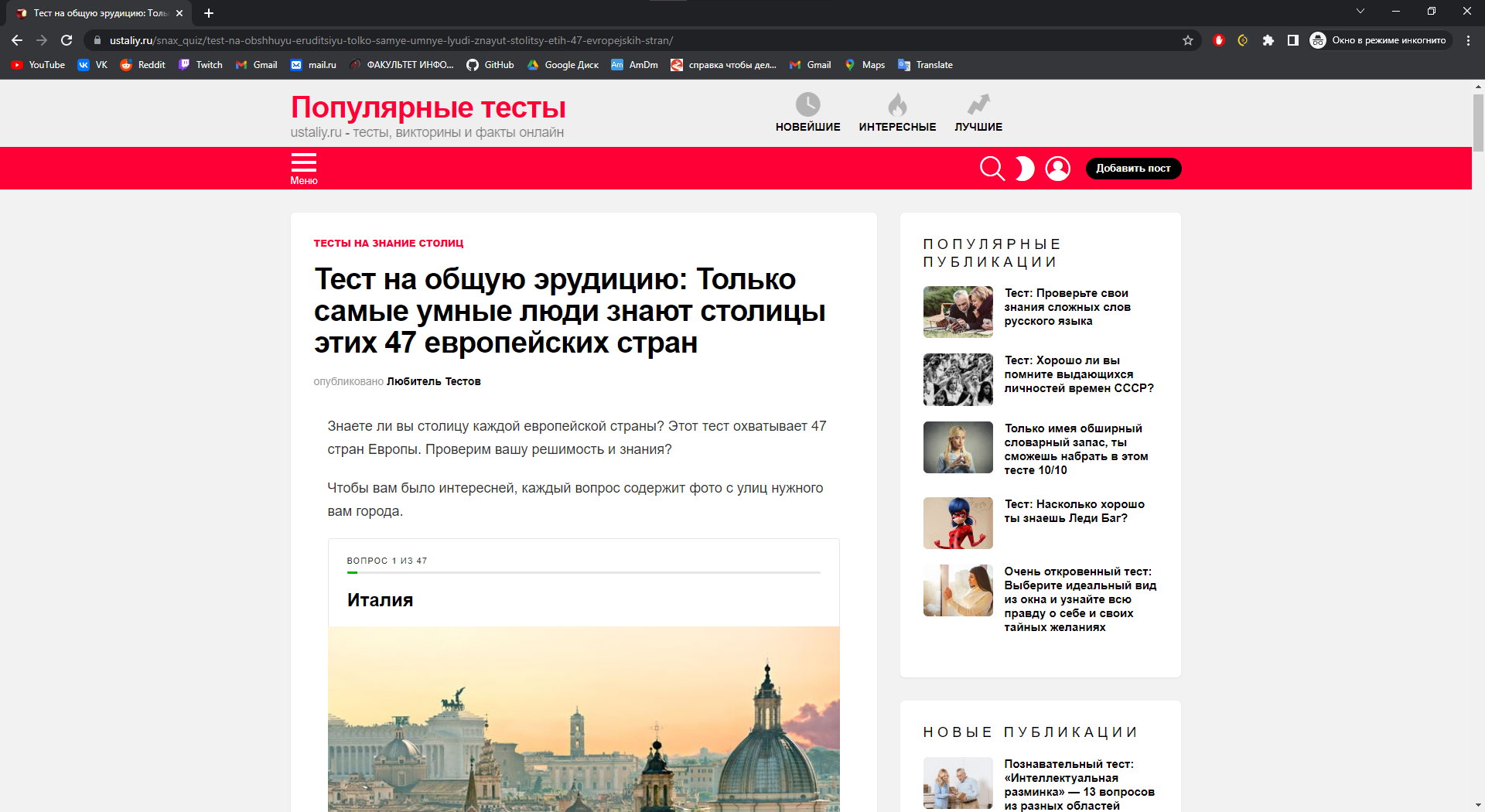


Рисунок 1.2 – Интерфейс веб-приложения «ustaliy.ru»

## 

## 1.2 Постановка задачи

Проанализировав прототипы, были выделены основная задача и функциональные требования проекта.

Основной задачей курсового проекта является разработка десктопного приложения, позволяющего выполнять авторизацию и регистрацию пользователей, пользователю в лице администратора добавлять продукты с информацией о них с возможностью удаления. Пользователю в лице клиента просматривать все продукты, добавлять их в корзину, удалять и оформлять заказ с последующей возможностью отмены.

Функциональные требования описаны в главе 2.

# **2 Анализ требований к программному средству и разработка функциональных требований**

## 2.1 Определение требований к программному стредству

Для того, что бы приступить к этапу разработки проекта, необходимо четко сформулировать функциональные требования к программному средству.

Функционально ПС должно выполнять следующие задачи:

* регистрация и авторизация пользователей;
* сохранение рабочей информации в централизованной базе данных;
* для администратора добавление тестов с возможностью их удаления;
* для клиента прохождение тестов с возможностью просмотра результатов;
* для администратора просмотр всех тестов и пользователей с возможностью удаления.

## 2.2 Описание средств разработки

* интегрированная среда разработки Microsoft Visual Studio 2019;
* программная платформа .NET Framework 4.8;
* язык программирования C#;
* расширяемый язык разметки XAML;
* технология WPF;
* технология Entity Framework 6.4.4;
* Microsoft SQL Server 2019;

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.

Чтобы осуществлять связь между базой данных и приложением на C# необходим посредник. И именно таким посредником является технология Entity Framework. Она предоставляет собой объектно-ориентированную технологию доступа к данным, является object-relational mapping (ORM) решением для платформы .NET Framework. Entity Framework предоставляет возможность взаимодействия с объектами посредством LINQ to Entities. Центральной концепцией Entity Framework является понятие сущности или entity. Сущность представляет набор данных, ассоциированных с определенным объектом. Поэтому данная технология предполагает работу не с таблицами, а с объектами и их наборами.

## 2.3 Описание функциональности программного средства

Описание функциональности программного средства представлено с помощью UML-диаграммы вариантов использования (Приложение А).

Спецификация функциональности программного средства представлена в следующем пункте.

# **3. Проектирование и создание программного средства**

Проектирование программного средства — процесс создания проекта программного обеспечения. Целью проектирования является определение внутренних свойств системы и детализации её внешних свойств на основе исходных условий задачи. Исходные условия задачи уже были сформулированы во втором разделе данной пояснительной записки. Этап проектирования подразумевает их анализ.

## 3.1 Архитектура системы

В данном приложении используется архитектурный паттерн MVVM (MODEL-VIEW-VIEWMODEL). Он позволяет отделить логику приложения от визуальной части. MVVM состоит из трех частей:

* Модель описывает используемые в приложении данные.
* Представление определяет визуальный интерфейс, через который пользователь взаимодействует с приложением.
* Модель представления связывает модель и представление через механизм привязки данных. ViewModel также содержит логику по получению данных из модели, которые потом передаются в представление.

На рисунке 3.1 представлена диаграмма, которая показывает общую структуру приложения в рамках шаблона MVVM.



Рисунок 3.1 – Структура шаблона MVVM

View или представление определяет визуальный интерфейс, через который пользователь взаимодействует с приложением. Так как пользовательский интерфейс и качество его реализации играет далеко не последнее место в конечном результате, разработка эффективного интерфейса, приятного и удобного для конечного пользователя, является важной задачей.

Поэтому для хорошего проектирования View необходимо понять, как пользователь будет взаимодействовать с приложением.

Для этого была составлена схема на рисунке 3.2, на которой представлен принцип работы приложения с точки зрения пользователя.

В программном средстве при запуске необходимо реализовать регистрацию и авторизацию пользователей для дальнейшего использования приложения. Для авторизации входными параметрами являются логин и пароль пользователя, которые содержатся в базе данных. Для того, чтобы зарегистрироваться необходимо ввести логин, пароль и подтвердить пароль. Введенные данные успешно прошедшие валидацию, заносятся в базу данных.



Рисунок 3.2 – Общая схема приложения для пользователя

При авторизации пользователя он видит приветственный текст после чего может перемещаться с помощью навигационному меню по страницам, также он может выйти из аккаунта и сменить язык приложения. На странице «*Tests*» пользователь может осуществлять поиск по названию и проходить один из отображаемых тестов. На странице «*Results*» пользователь может просматривать результаты предыдущих прохождений тестов с названием теста, временем и датой. На странице «*Profile*» пользователь может просмотреть под какой учетной записью он находится в системе тестирования.

Все данные о тестах, пользователях и результатах должны заноситься в базу данных в соответствующие таблицы.

Если вы зашли под ролью администратора, вам доступная схема приложения, которая представлена на рисунке 3.3



Рисунок 3.3 – Общая схема для администратора

При авторизации администратора на странице «*Users*» он может просматривать список всех пользователей, а также просматривать их роль. На странице «*Tests*» администратор может просматривать список всех тестов и имеет возможность их удаления. На странице «*Create New*» администратор может создать новый тест. На странице «*Results*» администратор может просматривать все результаты всех пользователей и очищать историю прохождений

Структура проекта описана в таблице 3.4.

Таблица 3.4 – Описание структурных пакетов проекта

|  |  |
| --- | --- |
| Имя пакета | Описание |
| Models | Описаны модели, на основе которых происходит генерация и работа с базой данных. |
| ViewModels | Содержит логику, которая позволяет получить данные при помощи Views, обработать их, используя при этом Models |
| Views | Содержит все представления, которые позволяют пользователю работать с приложением. Они описывают графическую составляющую приложения. |
| PasswordEncryptor | Содержит метод для хэширования пароля. |
| DataBase | Содержит класс контекста подключения к базе данных. |
| Resources | Файлы ресурсов приложения. |
| App.config | Файл конфигурации приложения. |
| App.xaml | Основная задача данного файла состоит в определении ресурсов, общих для приложения. |
| Command | Содержит класс, при помощи которого реализуется паттерн Command |
| SingletonView | Содержит классы, при помощи которых реализуется паттерн Singleton |

ViewModels или модели представления связывает модель и представление через механизм привязки данных. Она содержит Модель, преобразованную к Представлению, а также команды, которыми может пользоваться Представление, чтобы влиять на Модель.

Models или Модели представляет собой логику работы с данными и описание фундаментальных данных, необходимых для работы приложения.

## 3.2 Проектирование логической структуры базы данных

Диаграмма базы данных представлена на рисунке 3.5

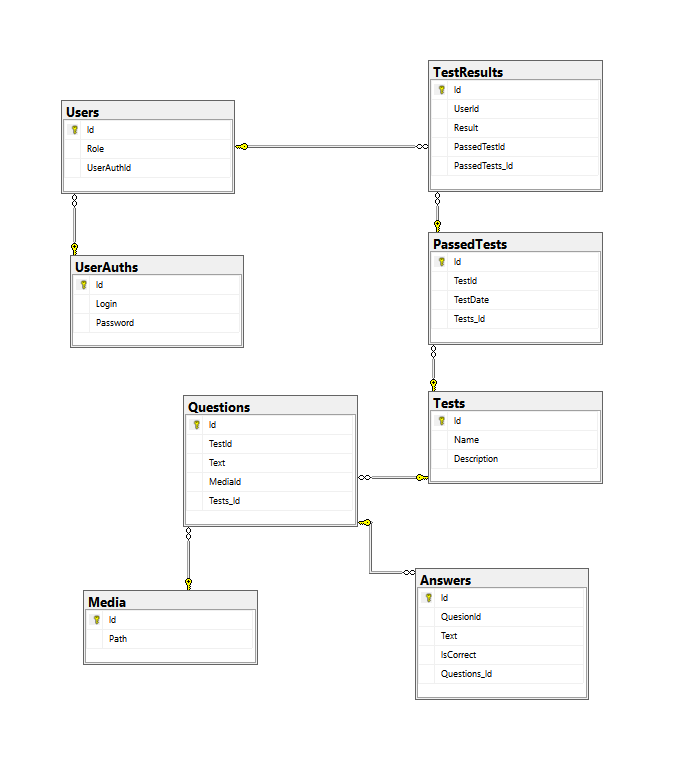


Рисунок 3.5 – Диаграмма базы данных

Таблица 3.5 – структура таблицы UserAuth

|  |  |
| --- | --- |
| Название столбца | Хранимые данные |
| id | Id авторизационных данных пользователя |
| Login | Имя пользователя |
| Password | Хешированный пароль |

Таблица 3.6 – структура таблицы User

|  |  |
| --- | --- |
| Название столбца | Хранимые данные |
| id | Id пользователя |
| Role | Роль пользователя |
| UserAuthId | Id авторизационных данных пользователя |

Таблица 3.7 – структура таблицы Tests

|  |  |
| --- | --- |
| Название столбца | Хранимые данные |
| id | Id теста |
| Name | Название теста |
| Description | Описание теста |

Таблица 3.8 – структура таблицы Questions

|  |  |
| --- | --- |
| Название столбца | Хранимые данные |
| id | Id вопроса |
| TestId | Id теста |
| Text | Текст вопроса |
| MediaId | Id медиаданных |
| Tests\_Id | Id теста |

Таблица 3.9 – структура таблицы Answers

|  |  |
| --- | --- |
| Название столбца | Хранимые данные |
| id | Id ответа |
| QuesionId | Id вопроса |
| Text | Текст ответа |
| IsCorrect | Флаг правильности ответа |
| Question\_Id | Id вопроса |

Таблица 3.10 – структура таблицы TestResults

|  |  |
| --- | --- |
| Название столбца | Хранимые данные |
| id | Id результата |
| UserId | Id пользователя |
| Result | Результат |
| PassedTestId | Id данных о пройденном тесте |
| PassedTests\_Id | Id данных о пройденном тесте |

Таблица 3.11 – структура таблицы PassedTests

|  |  |
| --- | --- |
| Название столбца | Хранимые данные |
| id | Id данных о пройденном тесте |
| TestId | Id теста |
| TestDate | Дата и время прохождения теста |
| Tests\_Id | Id теста |

Таблица 3.12 – структура таблицы Media

|  |  |
| --- | --- |
| Название столбца | Хранимые данные |
| id | Id медиаданных |
| Path | Путь к медиаданным |

## 3.3 Проектирование доступа к базе данных

Для доступа к базе данных используется Entity Framework 6. Этот подход предоставляет ряд существенных преимуществ: нам не нужно беспокоиться о коде доступа к данным и не нужно знать деталей работы СУБД SQL Server и синтаксиса языка запросов T-SQL, вместо этого мы работаем с таблицами базы данных как с классами C#, с полями этих таблиц - как со свойствами классов, а синтаксис SQL-запросов заменен на [LINQ](https://professorweb.ru/my/LINQ/base/level1/info_linq.php). Entity Framework берет на себя обязанности по преобразованию кода C# в SQL-инструкции.

В своем проекте я использовал подход Code-First, с помощью которого создавалась база данных по написанной мной модели;

## 4 Реализация программного средства

## 4.1 Реализация MVVM

Паттерн MVVM реализуется через базовый класс BaseViewModel, который реализует интерфейс INotifyPropertyChanged и от которого наследуются все страницы. Для реализации паттерна файлы программы были распределены по соответствующим пространствам имен (см. 3.2) и реализованы следующие функции. На рисунке 4.1 видим, что класс WelcomeViewModel наследует класс BaseViewModel (рис. 4.2) .

|  |
| --- |
| class WelcomeViewModel : BaseViewModel  {  public void Close()  {  foreach (System.Windows.Window window in System.Windows.Application.Current.Windows)  {  if (window.DataContext == this)  {  window.Close();  }  }  } |

Листинг 4.1 – Пример использования BaseViewModel

|  |
| --- |
| public class BaseViewModel : INotifyPropertyChanged  {  public event PropertyChangedEventHandler PropertyChanged;  public void OnPropertyChanged([CallerMemberName] string prop = "")  {  if (PropertyChanged != null)  {  PropertyChanged(this, new PropertyChangedEventArgs(prop));  }  } |

Листинг 4.2 – Структура класса BaseViewModel

В приложении используется паттерн Command который позволяет инкапсулировать запрос на выполнение определенного действия в виде отдельного объекта. В WPF команды представлены интерфейсом ICommand. В приложении он представлен в виде собственной команды Command код которой приведен в приложении Е. Класс реализует два метода:

* CanExecute: определяет, может ли команда выполняться
* Execute: собственно, выполняет логику команды

## 4.2 Реализация авторизации и регистрации пользователей

Для того чтобы пользоваться приложением каждый пользователь должен войти в систему, предварительно зарегистрировавшись.

На рисунке 4.3 находится код, реализующий регистрацию.

****

Рисунок 4.3 – Команда, реализующая регистрацию

Для безопасности хранения информации используется хэширование, в данном случае применяется встроенный класс MD5. Код метода хэширования приведен на рисунке 4.4.

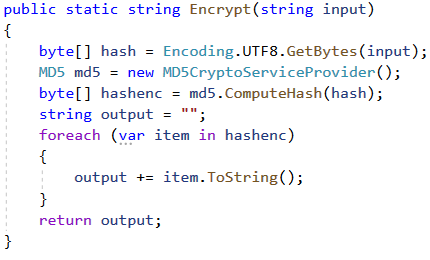


Рисунок 4.4 – Пример метода обеспечивающего хэширование пароля

При регистрации пользователю задается тип аккаунта User. Если все данные были введены верно и логин пользователя, почта еще не заняты, то осуществляется регистрация и переход на страницу авторизации.

Данные при регистрации проходят проверку по условиям максимальной и минимальной длины.

При авторизации проверяется существует ли пользователь с такими данными, а затем проверяется значения хэш-функций паролей. В случае неудовлетворения требований появляется надпись с ошибкой. В ином случае создается окно обычного пользователя или администратора в зависимости от значения в поле Role в базе данных. Пример команды, реализующей авторизацию приведен на рисунке 4.6.



Рисунок 4.6 – Команда, реализующая авторизацию

## 4.3 Выбор теста для прохождения и его запуск

Выбор теста и его запуск производится с помощью команды, привязанной на странице, где отображаются все существующие тесты. Пример команды, отвечающей за данное действие приведен на рисунке 4.7.



Рисунок 4.7 – Команда, реализующая выбор теста и его запуск

Удаление теста происходит аналогично по аналогичной команде, но на странице администратора. Пример команды, реализующей удаление приведен на рисунке 4.8.



Рисунок 4.8 – Команда, реализующая удаление теста

При удалении теста, результаты, связанные с удаленным тестом удаляются из базы данных.

## 4.4 Переход по вопросам и подсчет результата после ответа на последний вопрос

Метод отвечает за переход по вопросам текущего теста, которые получаются из базы данных, после ответа на которые подсчитывается результат и записывается в соответствующие таблиц. Пример команды приведен на рисунке 4.9.



Рисунок 4.9 – Команда, реализующая запись результатов прохождения теста

## 4.5 Просмотр собственных результатов

Для просмотра собственных результатов приложение запрашивает данные о пройденных тестах конкретным пользователем, после чего с помощью привязки размещает их в DataGrid на соответствующей странице.

Пример конструктора отвечающего за это на рисунке 4.10.

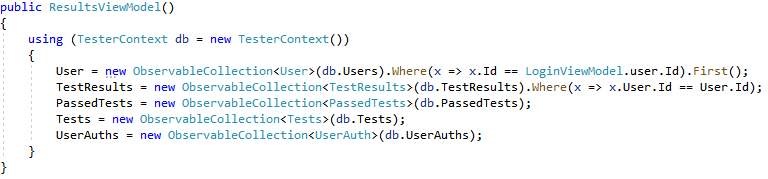


Рисунок 4.10 – Конструктор, который при вызове получает результаты текущего пользователя

## 4.6 Создание теста

Создание теста происходит администратором на странице CreateTestView, где происходит заполнение информации о тесте (название и описание), после чего нас перенаправляет к конструктору вопросов. В вопросах имеется 4 варианта ответов, так же может быть прикреплена картинка.

Пример команд, реализующих создание теста, вопросов и прикрепление к вопросу картинки приведены на рисунках 4.11, 4.12 и 4.13.

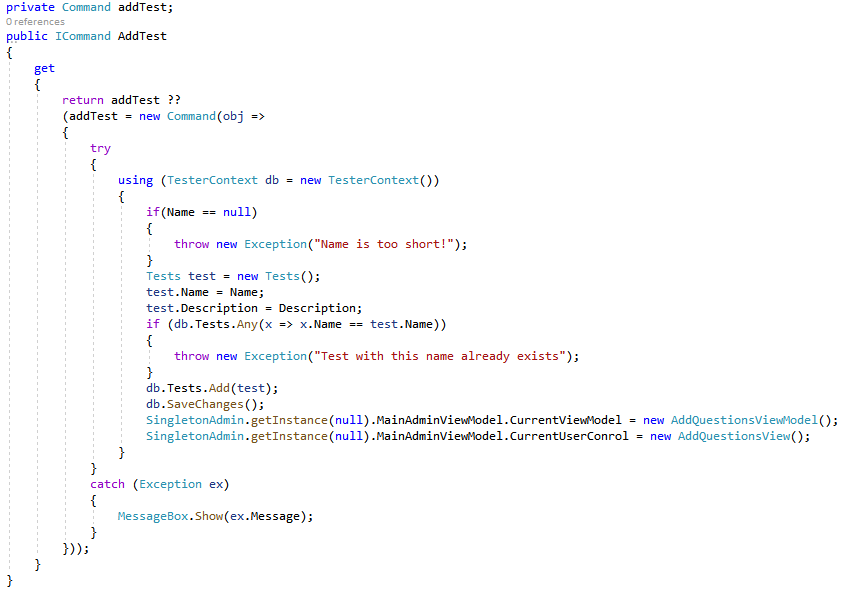


Рисунок 4.11– Команда, реализующая создание теста



Рисунок 4.12– Команда, реализующая добавление вопроса



Рисунок 4.13 – Команда, реализующая добавление картинки к вопросу

## 4.7 Очистка истории прохождения тестов администратором

Администратор имеет возможность очистить историю результатов по всем пользователям. Команда, предназначенная для очистки истории, приведена на рисунке 4.14.



Рисунок 4.13 – Команда, реализующая очистку истории прохождения тестов

# **5 Тестирование**

В курсовом проекте задействуется обработка ошибок, таким образом, что пользователь будет уведомлен о неудачном выполнении запроса к базе данных, , или недоступности данных в формах. Присутствуют различные элементы, всплывающие окна уведомляющие пользователя.

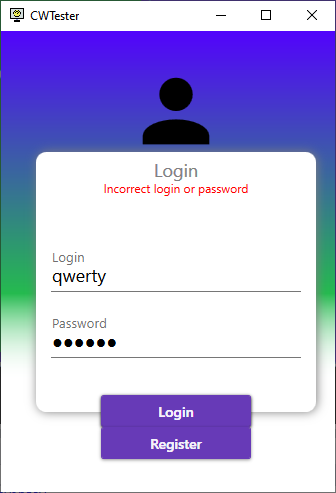


Рисунок 5.1 – Отсутствие введенного логина

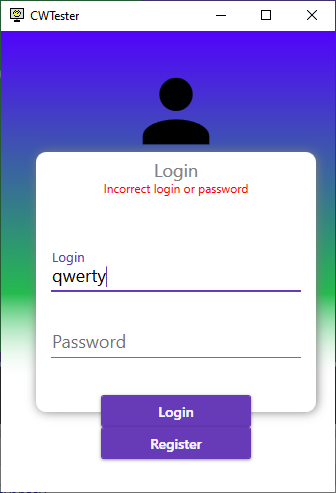


Рисунок 5.2 – Отсутствие введенного пароля

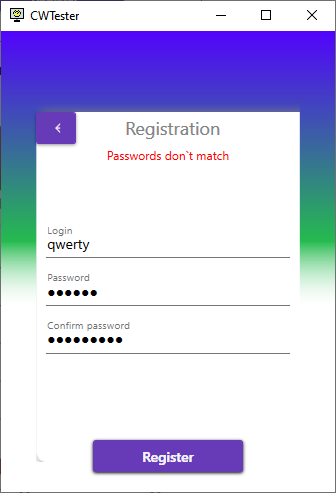


Рисунок 5.3 – Несовпадение паролей

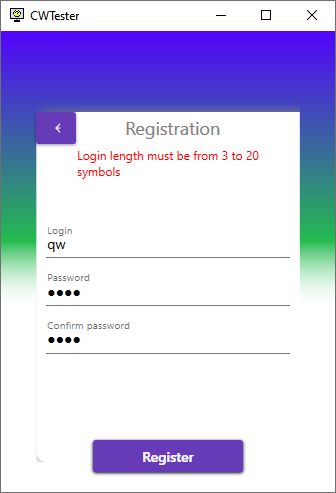


Рисунок 5.4 – Слишком короткое имя пользователя

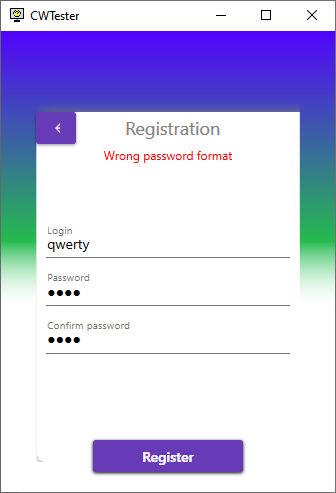


Рисунок 5.5 – Слишком короткий пароль

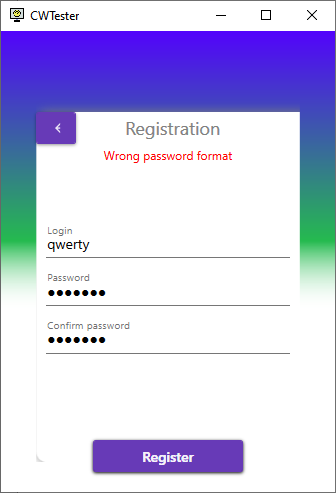


Рисунок 5.7 – Пароль начинается с пробела

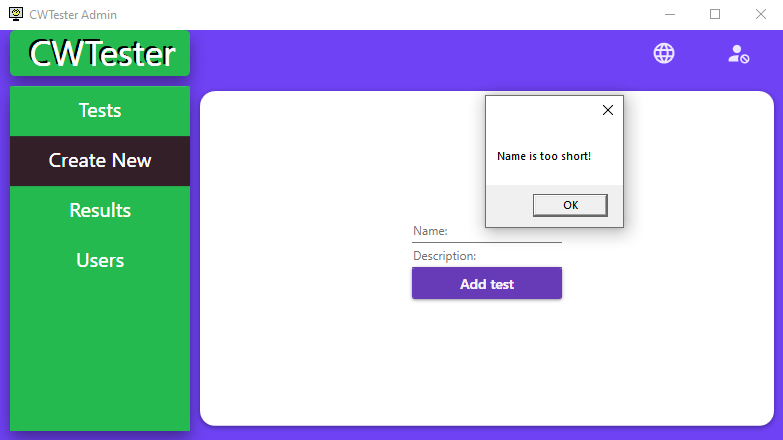


Рисунок 5.8 – Попытка создать тест с пустым именем

# **6 Руководство по использованию**

## 6.1 Регистрация и авторизация

После запуска приложения пользователь видит главное окно приложения, на котором он может авторизоваться и получить возможность приобретать товары. Окно представлено на рисунке 6.1.

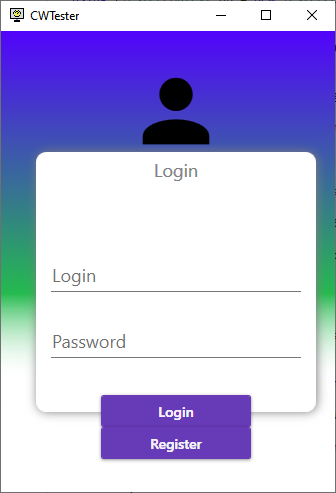


Рисунок 6.1 – Окно входа в аккаунт

Если пользователь ещё не имеет своего аккаунта, то по нажатию на соответствующую кнопку можно перейти к окну регистрации. Пользователь должен придумать логин и пароль, которые будет использовать для авторизации в приложении. Окно регистрации показано на рисунке 6.2.

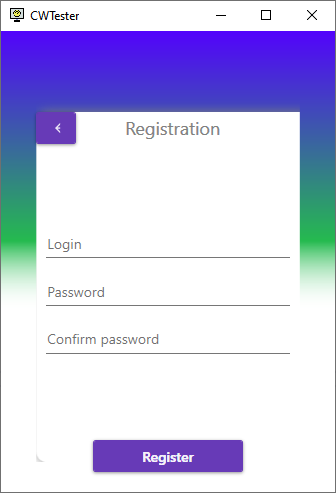


Рисунок 6.2 – Окно регистрации

## 6.2 Использование приложения обычным пользователем

После успешной авторизации пользователь попадает на приветственный экран, где выражается благодарность за использование приложения. В навигационном меню есть 2 кнопки для смены языка с английского на русский, и кнопка для выхода из аккаунта. Так же на начальном экране имеется 3 кнопки навигации «Tests», «Results» и «Profile», предназначенные для перемещения на соответствующие страницы.

Стартовое окно пользователя приведено на рисунке 6.3.

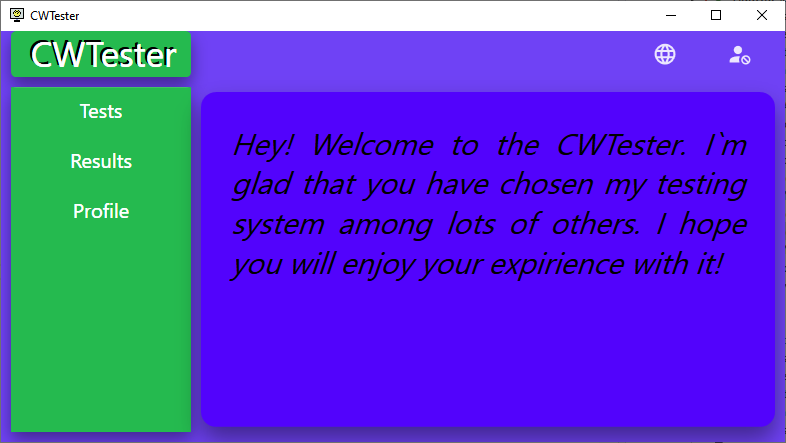


Рисунок 6.3 – Главное окно пользователя

## 6.3 Использование приложения администратором

После успешной авторизации администратор попадает в главное окно администратора, где видит 10 для навигации по страницам и выхода их приложения и приветственный текст с предложением создать новый тест.

Пример главного окна администратора приведен на рисунке 6.4.

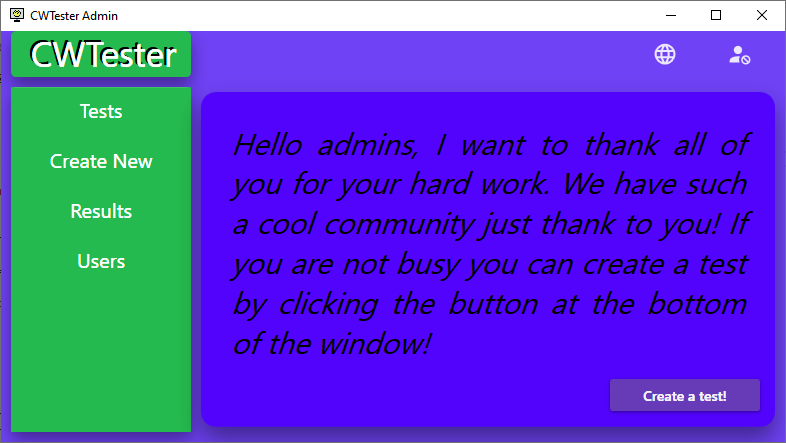


Рисунок 6.4 – Главное окно администратора

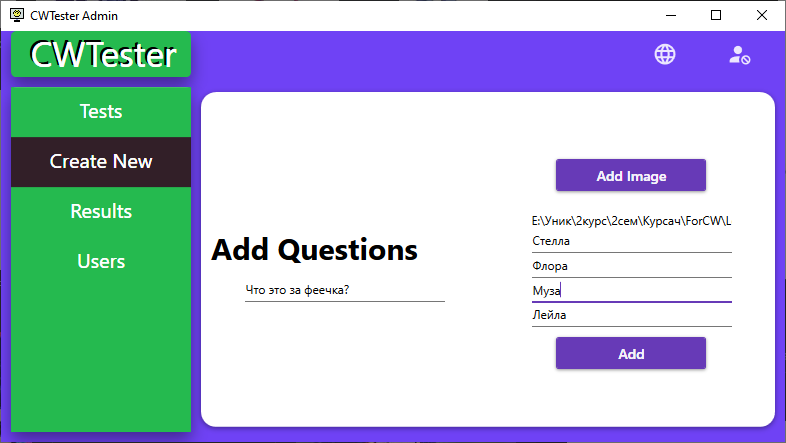


Рисунок 6.4 – Страница добавления вопросов в тесты

# **ЗАКЛЮЧЕНИЕ**

В итоге выполнения курсового проекта было разработано программное средство «CWTester», предоставляющее возможность создавать тесты, проходить их, отслеживать результаты своих прохождений а так же со стороны администратора вести контроль результатов всех пользователей.

Также в процессе выполнения данного курсового проекта были закреплены навыки в программировании на языке C#, создании приложений на WPF, использование Entity Framework 6, работа с паттерном MVVM, проектирование базы данных и реализация их в СУБД MS SQL Server 2019.

Был получен опыт взаимодействия со сторонними библиотеками, изучения их и внедрения в свой проект.

**СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

1) Пацей, Н.В. Курс лекций по языку программирования С# / Н.В. Пацей. – Минск: БГТУ, 2021. – 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/

4) METANIT.COM Сайт о программировании [Электронный ресурс] / Режим доступа: https://metanit.com

5) ProfessorWeb .NET & Web Programming [Электронный ресурс] / Режим доступа: https://professorweb.ru

6) StackOverflow [Электронный ресурс] / Режим доступа: <https://stackoverflow.com>

7) Хабр [Электронный ресурс] / Режим доступа: <https://habr.com>.

## Приложение А

## Приложение Б

## Приложение В

## Приложение Г

