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

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

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

Тема «Приложение для прохождения тестов и решения математических задач»

Исполнитель

студент 2 курса 4 группы Костюкова Анна Олеговна

Руководитель работы ассистент Рауба А.А.

Курсовой проект защищен с оценкой

Председатель Рауба А.А.

Минск 2020

**Содержание**

[ВВЕДЕНИЕ 4](#_Toc41150864)

[1 Постановка задачи и обзор литературы 5](#_Toc41150865)

[1.1 Алгоритмы решения 5](#_Toc41150866)

[1.2 Обзор прототипов 6](#_Toc41150867)

[1.3 Актуальность задачи 9](#_Toc41150868)

[2 Проектирование архитектуры проекта 11](#_Toc41150869)

[2.1 Архитектура системы 11](#_Toc41150870)

[2.2 Проектирование логической структуры базы данных 11](#_Toc41150871)

[2.3 Проектирование структуры классов 12](#_Toc41150872)

[2.4 Проектирование структуры окон 15](#_Toc41150873)

[3 Разработка функциональной модели и модели данных ПС 16](#_Toc41150874)

[3.1 Функции, реализуемые программой 16](#_Toc41150875)

[3.2 Реализация классов 16](#_Toc41150876)

[3.3 Реализация работы с базой данных 18](#_Toc41150877)

[3.4 Реализация паттерна MVVM 19](#_Toc41150878)

[4 Тестирование 21](#_Toc41150879)

[4.1 Тест авторизации 21](#_Toc41150880)

[4.2 Тест регистрации 23](#_Toc41150881)

[4.3 Тест отправки сообщения 26](#_Toc41150882)

[4.4 Тест использования калькулятора 27](#_Toc41150883)

[ЗАКЛЮЧЕНИЕ 28](#_Toc41150884)

[Список используемых источников 29](#_Toc41150885)

[ПРИЛОЖЕНИЕ А 30](#_Toc41150886)

[ПРИЛОЖЕНИЕ Б 31](#_Toc41150887)

[ПРИЛОЖЕНИЕ В 34](#_Toc41150888)

[ПРИЛОЖЕНИЕ Г 35](#_Toc41150889)

[ПРИЛОЖЕНИЕ Д 36](#_Toc41150890)

[ПРИЛОЖЕНИЕ Е 39](#_Toc41150891)

[ПРИЛОЖЕНИЕ Ж 40](#_Toc41150892)

[ПРИЛОЖЕНИЕ З 41](#_Toc41150893)

[ПРИЛОЖЕНИЕ И 44](#_Toc41150894)

# **ВВЕДЕНИЕ**

На сегодняшний день цифровые технологии сильно изменили жизнь человека, в том числе и подход к образованию. Компьютер предоставляет большие возможности для организации учебного процесса.

Система дистанционного обучения – это совокупность программных средств для организации обучения и проверки знаний студентов посредством интернет-технологий. Данная система может применяться не только для обучения по конкретным предметам, но и факультетам в целом.

С помощью системы дистанционного образования удобно составлять учебный процесс, отслеживать и анализировать успеваемость студентов, так как все учебные материалы хранятся в одном месте, в том числе и история студента, которая позволяет просматривать курсы, посещаемые студентом, а также количество времени, потраченное на изучение материала.

Целью курсового проекта является создания приложения «Lova» – это программное средство, для организации дистанционного обучения по предмету «Высшая математика. С помощью этого приложения преподаватель может создавать обсуждения и тесты, для оценивания уровня знаний студентов и дальнейшего планирования учебного процесса.

В качестве интерфейса прикладного программирования 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. Язык разработки проекта – C#. При выполнении курсового проекта будут использованы принципы и приемы ООП.

# **1 Постановка задачи и обзор литературы**

Основной задачей курсового проекта является разработка программного средства, которое позволяет хранить информацию в базе данных и добавлять новую информацию в уже имеющуюся.

Функционально приложение поддерживает:

Функции студента:

* Выполнять регистрацию и авторизацию;
* Присылать ответы на форум;
* Выполнять тесты;
* Искать нужный тест по названию;
* Использовать встроенный калькулятор.

Функции преподавателя:

* Выполнять авторизацию;
* Создавать и удалять тесты;
* Создавать и удалять обсуждения;
* Отправлять сообщение на форум;
* Удалять сообщения с форума;
* Просматривать статистику выполнения студентами тестов.

## **1.1 Алгоритмы решения**

При разработке приложения были использованы следующие технологии:

* WPF;
* Entity Framework Core;
* Microsoft SQL Server.

Windows Presentation Foundation (WPF) – это современная графическая система отображения для Windows. Это радикальное изменение от технологий, которые были до него, с инновационными функциями, такими как встроенные аппаратное ускорение и независимость от разрешения, которые вы будете изучать в этой главе.

WPF – это лучший набор инструментов, который можно использовать, если вы хотите создать богатое настольное приложение, работающее на Windows 7, Windows 8 и Windows 10 в режиме рабочего стола, на основе современных пользовательских интерфейсов, которые могут включать в себя мультимедиа, графика, документы и различные данные [1].

Entity Framework Core (EF Core) – это объектно-ориентированная и легко расширяемая технология доступа к данным. EF Core представляет собой более высокий уровень абстракции по сравнению с ADO.NET и позволяет абстрагироваться от самой базы данных и ее таблиц, работая с данными независимо от типа хранилища.

Entity Framework Core поддерживает множество различных систем баз данных. По умолчанию Microsoft предоставляет ряд встроенных провайдеров: для работы с MS SQL Server, для SQLite, для MySQL.

Entity Framework Core многое унаследовал от Entity Framework 6, но EF Core – это не новая версия по отношению к EF 6, а совершенно иная технология, поэтому в её рамках используется своя система версий. Текущая версия – 3.0 была выпущена в сентябре 2019 года. Технология продолжает развиваться [2].

Microsoft SQL Server – это система управления реляционными базами данных, разработанная корпорацией Microsoft. Основной используемый язык запросов – Transact-SQL, который является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия.

Microsoft SQL Server в качестве языка запросов использует версию SQL являющуюся реализацией SQL-92 с множественными расширениями. T-SQL позволяет использовать дополнительный синтаксис для хранимых процедур и обеспечивает поддержку транзакций [3].

## **1.2 Обзор прототипов**

Отдел дистанционных образовательных технологий учебно-методического управления создан с целью внедрения в образовательный процесс БГТУ информационно-коммуникационных технологий.

В университете система дистанционного обучения применяется при проведении различных видов учебных занятий для студентов всех форм получения образования I и II ступени. Степень внедрения образовательных технологий с использованием СДО в процесс обучения определяет кафедра и преподаватель конкретной учебной дисциплины.

Для начала работы перейти по адресу [https://dist.belstu.by](https://dist.belstu.by/) и в поля Логин и Пароль(рисунок 1.1) ввести полный номер студенческого билета.

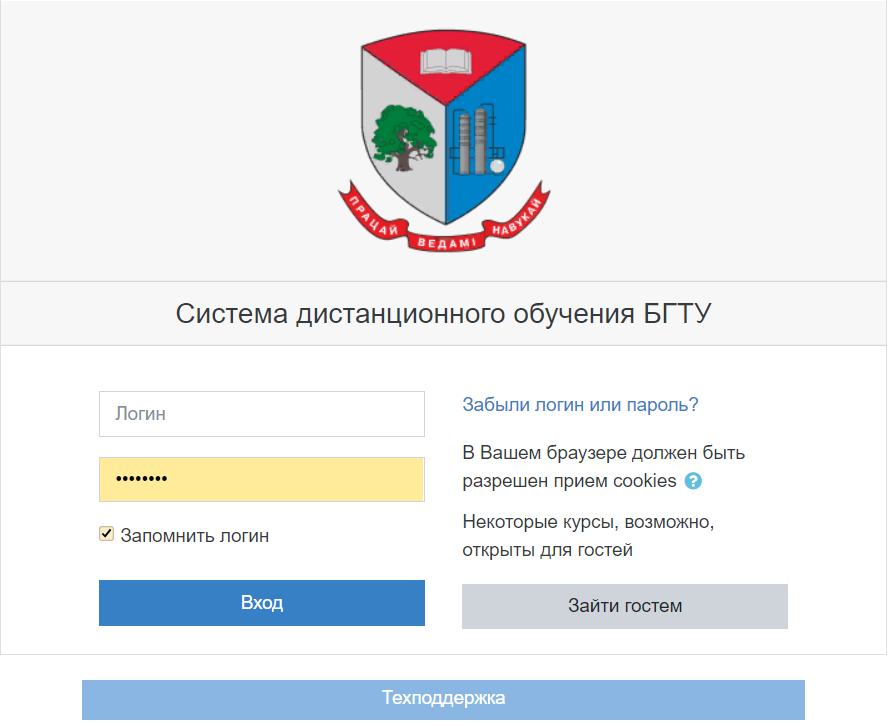


Рисунок 1.1 – Авторизация в систему дистанционного обучения БГТУ

После авторизации открывается личный раздел (рисунок 1.2), где в блоке Сводка по курсам или в меню слева выбирается из курсов, на которые Вы записаны, необходимая дисциплина.

Открывается страница, содержащая материалы соответствующей учебной дисциплины (лекции, презентации, тесты и др.).

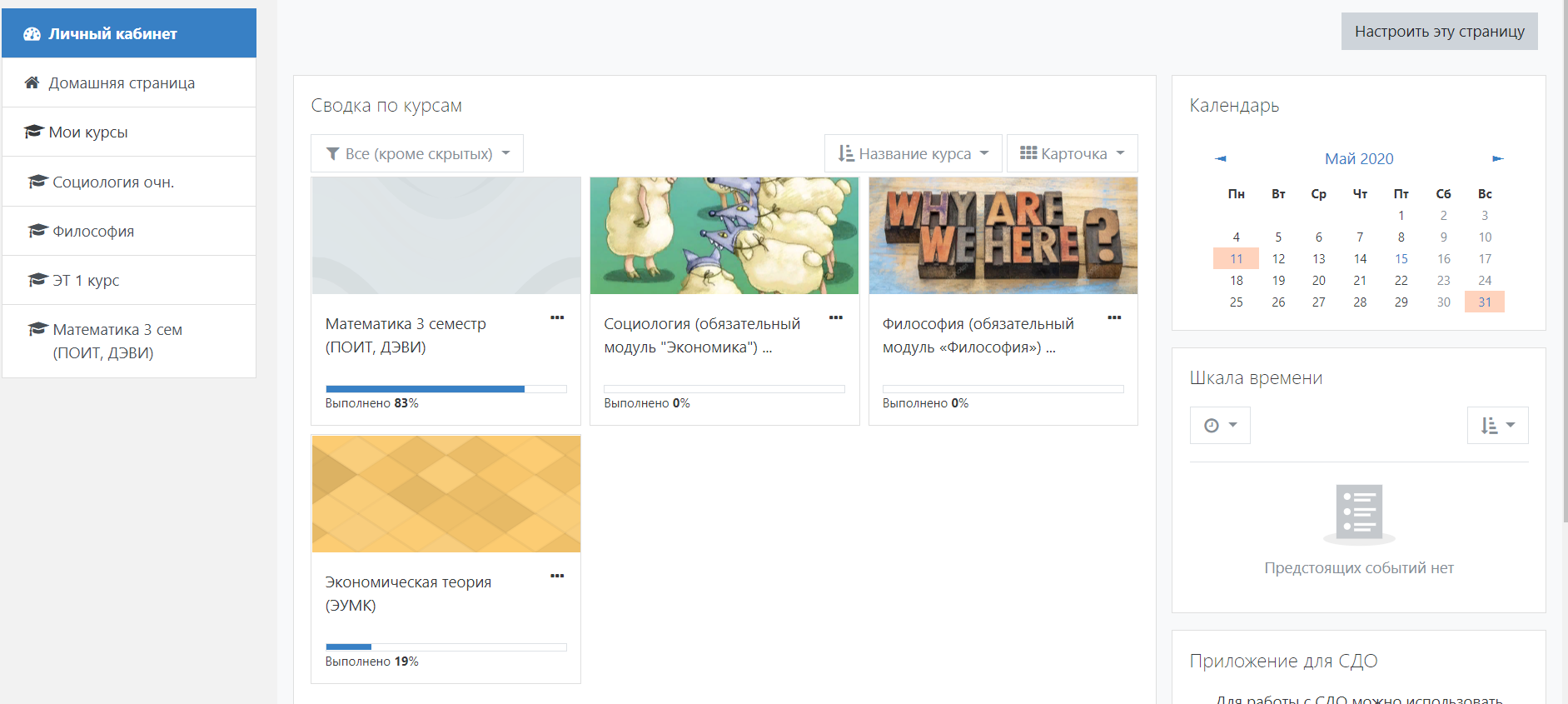


Рисунок 1.2 – Личный кабинет

Результат работы с курсомможно увидеть, выбрав в меню слева пункт Оценки(рисунок 1.3):

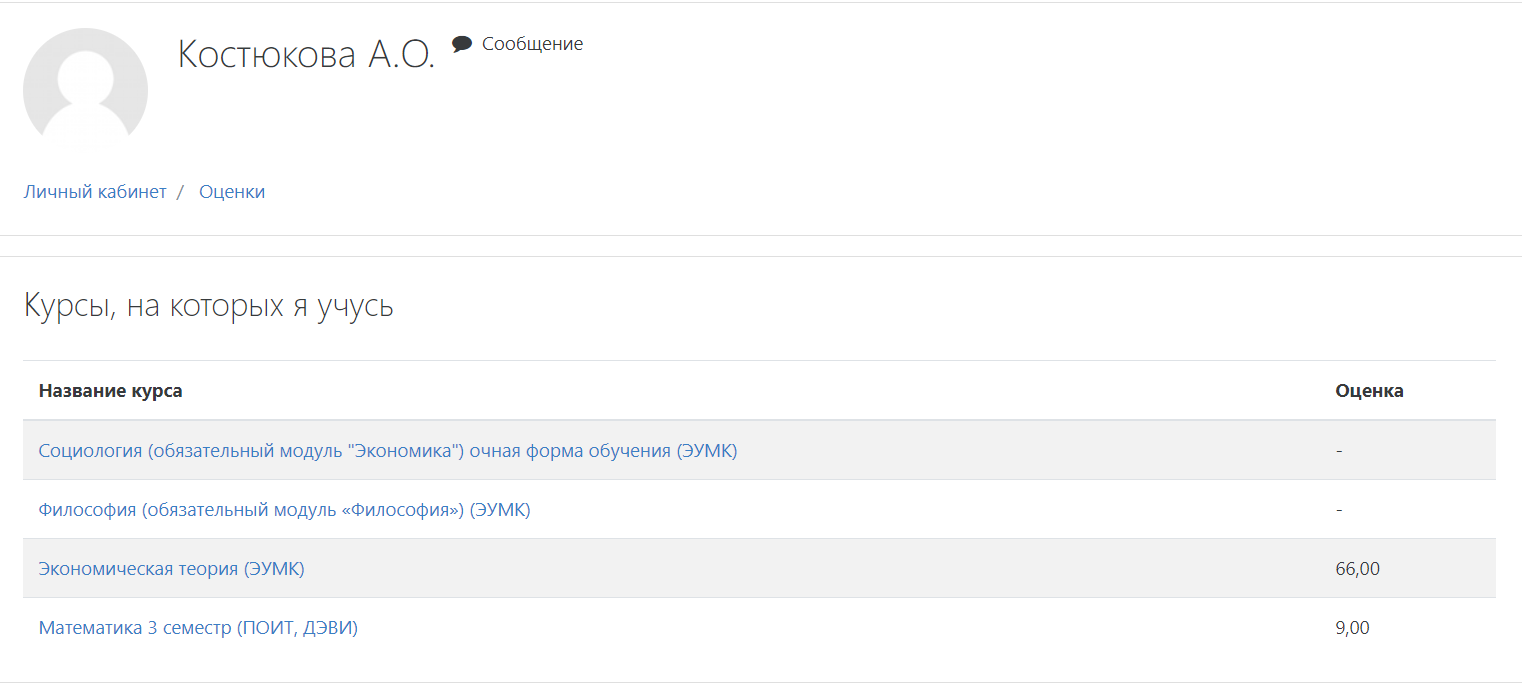


Рисунок 1.3 – Пункт Оценки

Основными целями применения системы дистанционного обучения в университете являются:

* создание возможностей получения качественных образовательных услуг на уровне современных требований национальных и международных стандартов с использованием современных информационно-коммуникационных технологий;
* эффективное вовлечение в образовательный процесс всего многообразия средств информатизации;
* развитие у обучающихся мотивации к получению знаний, непрерывному самообразованию посредством использования современных информационно-коммуникационных технологий;
* предоставление обучающимся возможности осваивать образовательные программы непосредственно по месту жительства или пребывания;
* обеспечение сетевого взаимодействия участников образовательного процесса;
* предоставление образовательных услуг в максимально удобной форме;
* создание единого образовательного пространства для обучающихся.

Приложение «Универсальный тест»

Универсальный тест – это программное средство для тестирования с возможностью добавлять новые тесты. Программа состоит из двух частей. Модуль под названием Tester предоставляет тесты для проверки знаний учащихся, студентов и персонала на самые разные темы, например, по русскому и английскому языку, информатике, математике, физике, и т.д. Эти тесты составлялись пользователями предыдущих версий программы. Во время тестирования выводится вопросы из базы в случайном порядке с вариантами ответов (рисунок 1.4).

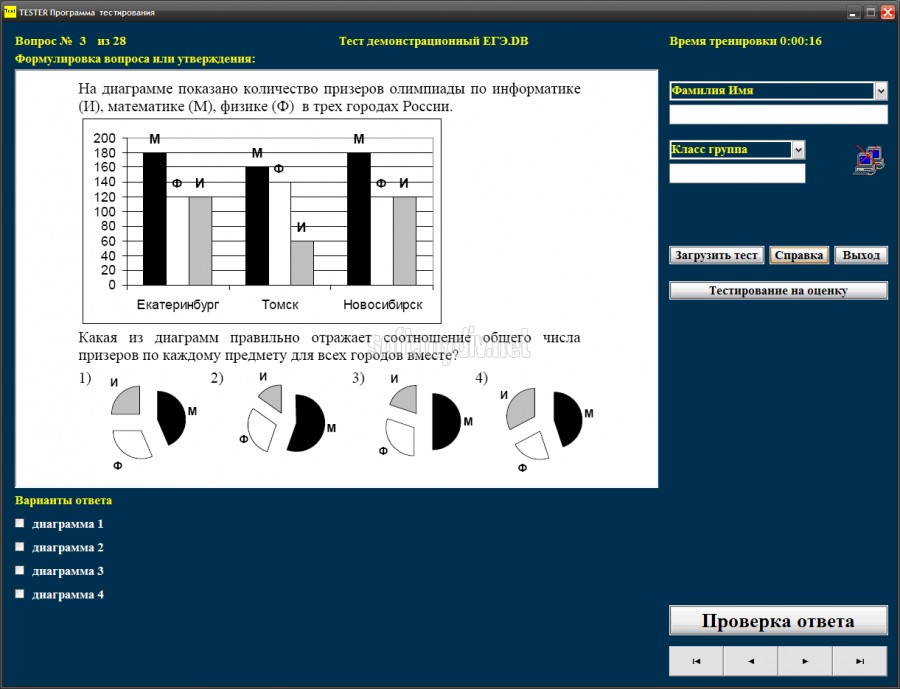


Рисунок 1.4 – Демонстрация теста

При помощи модуля TestMaker можно составлять собственные тесты. В них можно использовать графику и тексты с форматированием. Имеется возможность тестирования по локальной сети и сбора полученных результатов при помощи встроенного сервера.

Приложение *«AnsTester»*

AnsTester – это программное средство для создания тестов с системой начисления баллов. Помимо тестирования, программа позволяет проводить различные опросы. В пакет входят несколько модулей – один предназначен для проведения тестирования (рисунок 1.5), второй – для просмотра результатов тестов и третий – для создания тестов.

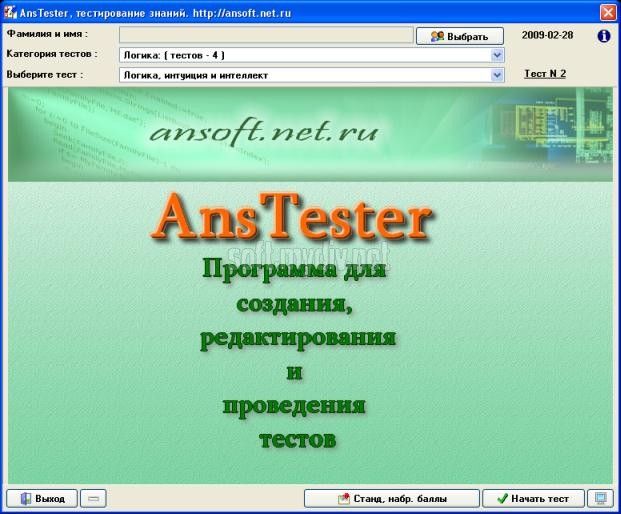


Рисунок 1.5 – Проведение тестирования

## **1.3 Актуальность задачи**

На сегодняшний день дистанционное обучение является важной сферой в образовании. С помощью системы дистанционного образования удобно составлять учебный процесс, отслеживать и анализировать успеваемость студентов, так как все учебные материалы хранятся в одном месте, в том числе и история студента, которая позволяет просматривать курсы, посещаемые студентом, а также количество времени, потраченное на изучение материала. Поэтому тема курсового проекта «Приложение для прохождения тестов и решения математических задач» выбрана не случайно.

Данное приложение позволяет решить проблему контроля за уровнем знаний, умений и навыков учащихся при помощи тестирования. Преподаватель сам составляет тесты по пройденному материалу и после может анализировать результаты студентов, выявляя наиболее часто встречающиеся ошибки.

Приложение как система прохождения тестов имеет ряд преимуществ:

* Студент имеет неограниченное время на прохождение теста, что позволяет в привычном ему темпе решать поставленный вопрос;
* Тест не содержит вариантов ответов, следовательно, нет возможности ответить наугад;
* Ответ можно менять неопределенное количество раз, на случай, если студент обнаружил неточность в своих вычислениях.

Помимо тестов в приложении присутствует возможность добавлять обсуждение, на котором все студенты могут принять участие в решении поставленной задачи. Возможность удалять сообщения есть только у преподавателя, чтобы студенты основательно подходили к отправленному ответу.

# **2 Проектирование архитектуры проекта**

## **2.1 Архитектура системы**

Для программного средства в курсовом проекте используется способ разработки программных модулей, основанный на клиентском приложении. Приложении состоит из 2 частей:

* Клиентской части, в которой происходит основное взаимодействие пользователя. Она имеет графический интерфейс.
* Базы данных. В данной БД хранятся тесты с правильными ответами, обсуждения, аккаунты студентов и результаты пройденными ими тестов.

## **2.2 Проектирование логической структуры базы данных**

При помощи подхода Code-First была создана база данных. Этот подход предполагает минимальное участие в проектировании сущностей базы данных программистом. Для её создания использовались система управления реляционными базами данных Microsoft SQL Server 2018.

База данных программной подсистемы состоит из 6 таблиц, взаимосвязанных друг с другом внешними ключами, ее схема изображена на рисунке 2.1.

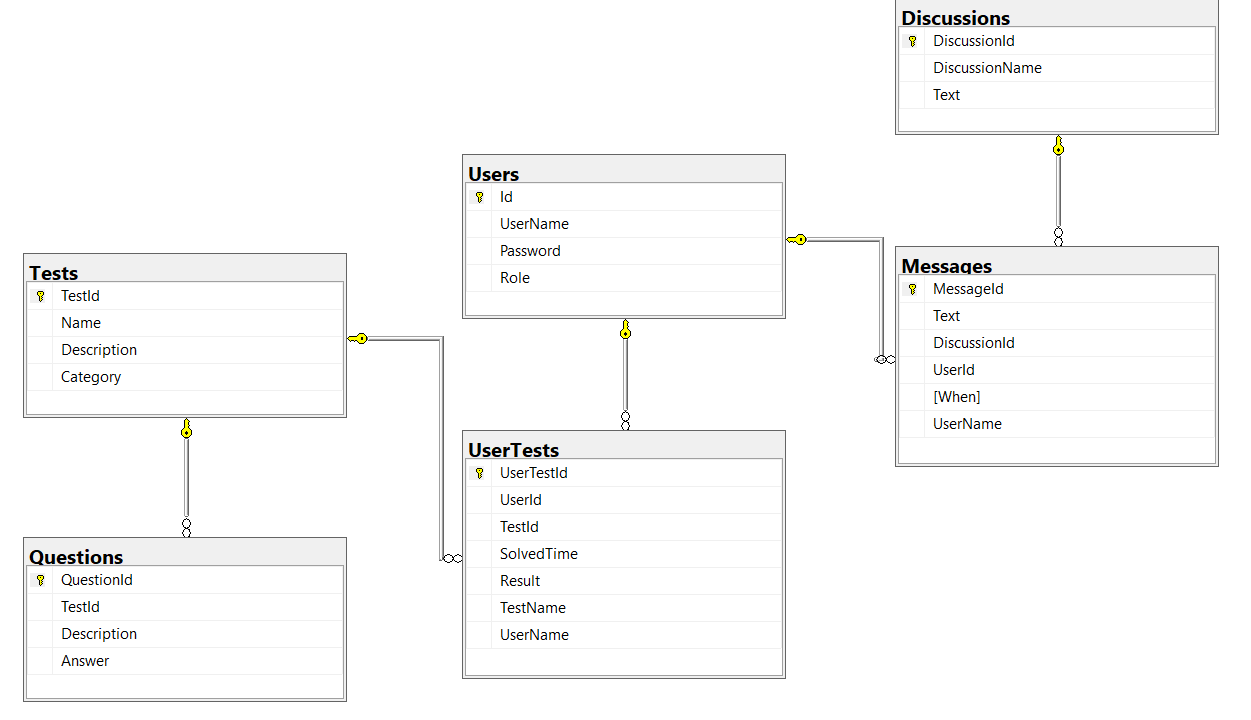


Рисунок 2.1 – Структура базы данных

Таблица Users хранит данные для входа. Включает 4 столбца: Id, UserName, Password, Role. Столбец Id является первичным ключом, инкрементируется при добавлении нового пользователя. Столбец UserName хранит имя пользователя. Столбец Password хранит пароль в зашифрованном виде. Столбец Role хранит роль пользователя: администратор или студент.

Таблица Messages хранит информацию о сообщениях, отправленные пользователями в обсуждениях. Включает 6 столбцов: Text, DiscussionId, UserId, UserName, When. Столбец MessageId является первичным ключом, инкрементируется при отправке нового сообщения. Столбец Text хранит сообщение пользователя, не допускается пустое сообщение. Столбец DiscussionId хранит идентификатор обсуждения, в которое отправляется сообщение. Столбец UserId хранит идентификатор пользователя, который отправил сообщение. Столбец UserName хранит имя пользователя, который отправил сообщение. Столбец When хранит дату и время отправки сообщения.

Таблица Discussions хранит информацию об обсуждениях. Включает 3 столбца: DiscussionId, DiscussionName, Text. Столбец DiscussionId является первичным ключом, инкрементируется при добавлении нового обсуждения. Столбец DiscussionName хранит название обсуждения. Столбец Text хранит текст обсуждения.

Таблица Tests хранит информацию о тестах. Включает 4 столбца: TestId, Name, Description, Category. Столбец TestId является первичным ключом, инкрементируется при добавлении нового теста. Столбец Name хранит название теста. Столбец Description хранит описание теста. Столбец Category хранит категорию теста.

Таблица UserTests хранит результаты пройденных тестов. Включает 7 столбца: UserTestId, UserId, TestId, SolvedTime, Result, TestName, UserName. Столбец UserTestId является первичным ключом, инкрементируется при пройденном тесте. Столбец UserId хранит идентификатор пользователя, который прошел тест. Столбец TestId хранит идентификатор пройденного теста. Столбец SolvedTime хранит время и дату, когда был пройден тест. Столбец Result хранит результат теста в формате строки «количество правильно отвеченных вопросов/общее количество вопросов». Столбец TestName хранит название пройденного теста. Столбец UserName хранит имя пользователя, который прошел тест.

Таблица Questions хранит информацию о вопросах к тестам. Включает 4 столбца: QuestionId, TestId, Description, Answer. Столбец QuestionId является первичным ключом, инкрементируется при добавлении нового вопроса к тесту. Столбец TestId хранит идентификатор теста. Столбец Description хранит описание вопроса. Столбец Answer хранит правильный ответ на вопрос.

## **2.3 Проектирование структуры классов**

Приложение курсового проекта состоит из 4 проектов: Calculator, Models, Presentation, XUnitTestProject (рисунок 2.2).

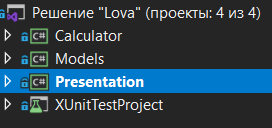


Рисунок 2.2 – Состав приложения

Проект Calculator представляет собой библиотеку и содержит два класса, диаграмма которых представлена на рисунке 2.3.

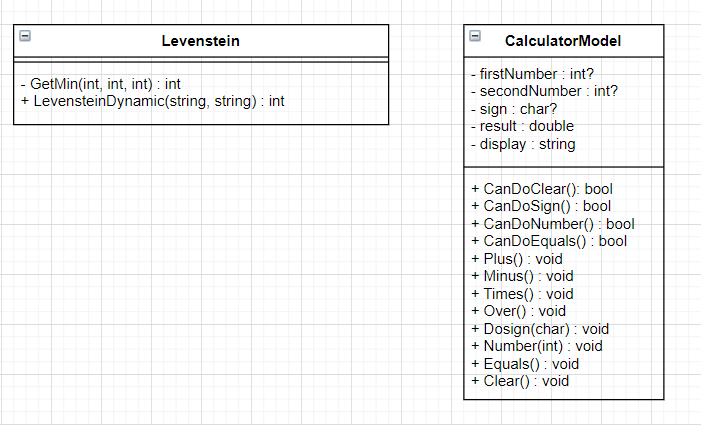


Рисунок 2.3 – Диаграмма классов проекта Calculator

Проект Models представляет собой библиотеку и включает в себя основные классы для работы с базой данных, модели, согласно которым генерируется база данных при помощи подхода Code-First, и миграции (рисунок 2.4).

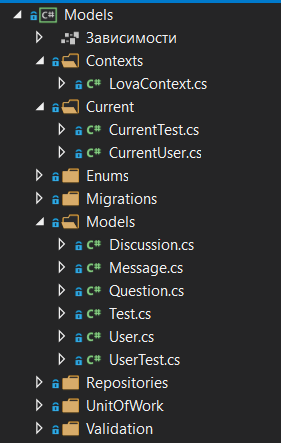


Рисунок 2.4 – Состав проекта Models

Проект Presentation является автозагружаемым проект и состоит из страниц и соответствующих им классам с суффиксом ViewModel (рисунок 2.5).

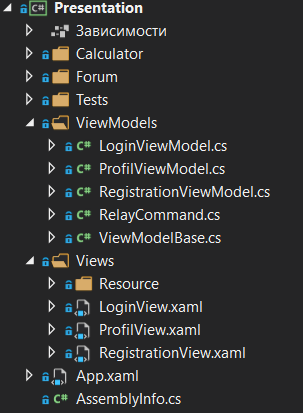


Рисунок 2.5 – Состав проекта Presentation

Проект XUnitTestProject является проектом для модульного тестирование на базе .Net Core. В нём содержатся два класса: TestCalculator и TestLevenstein (рисунок 2.6).

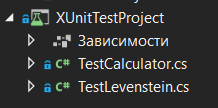


Рисунок 2.6 – Состав проекта XUnitTestProject

## **2.4 Проектирование структуры окон**

Приложение включает одно главное окно и 16 страниц. Связи между ними представлены на рисунке 2.7.

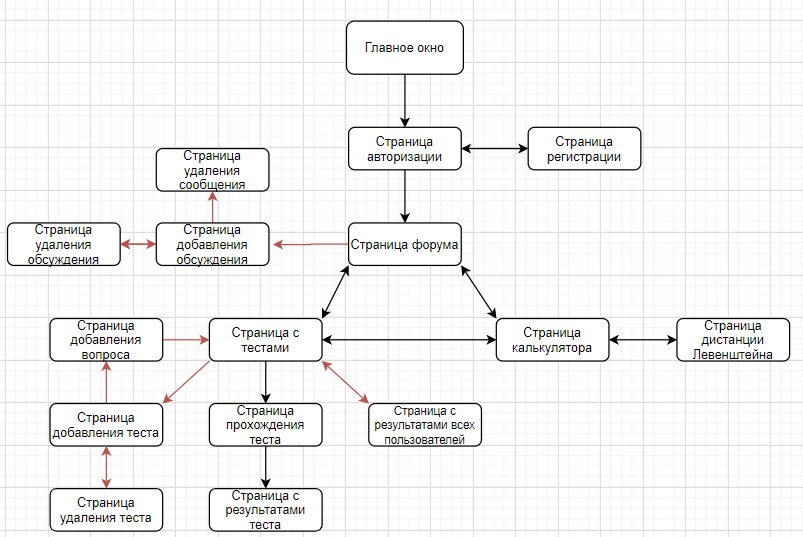


Рисунок 2.7 – Схема структуры окон

При запуске приложения появляется окно авторизации. Если пользователь не зарегистрирован, он может перейти на страницу регистрации. Если зарегистрированный пользователь по ошибке перешёл на страницу регистрации, есть возможность перейти на страницу авторизации.

При успешной авторизации пользователь попадает на страницу форума. Есть возможность перейти на страницу с тестами и калькулятором.

На схеме структуры окон красными стрелками показаны переходы на страницы, которые доступны только администратору.

# **3 Разработка функциональной модели и модели данных ПС**

## **3.1 Функции, реализуемые программой**

Функционально приложение поддерживает:

Функции студента:

* Выполнять регистрацию и авторизацию;
* Присылать ответы на форум;
* Выполнять тесты;
* Искать нужный тест по названию;
* Использовать встроенный калькулятор.

Функции преподавателя:

* Выполнять авторизацию;
* Создавать и удалять тесты;
* Создавать и удалять обсуждения;
* Отправлять сообщение на форум;
* Удалять сообщения с форума;
* Просматривать статистику выполнения студентами тестов.

Диаграмма вариантов использование представлена на рисунке 3.1.

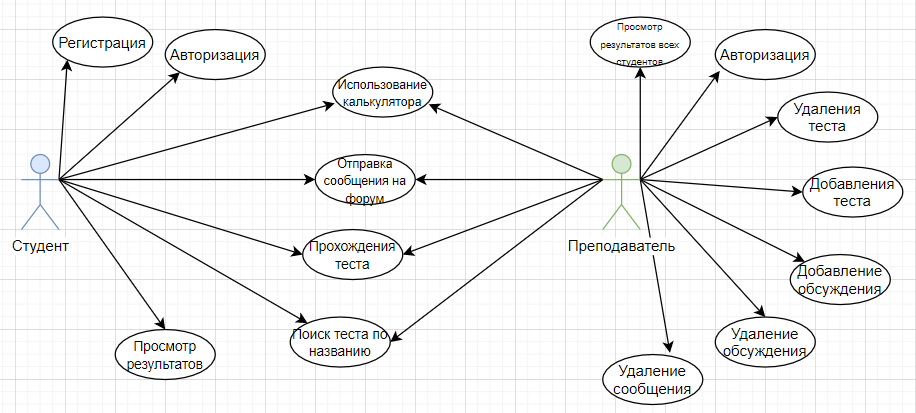


Рисунок 3.1 – Диаграмма вариантов использования

## **3.2 Реализация классов**

Данный раздел содержит описание и реализацию некоторых классов приложения.

Класс LovaContext, листинг которого представлен в приложении А, реализует интерфейс DbContext, необходимый для управления подключением к базе данных.

Класс CurrentUser используется для хранения идентификатора текущего пользователя (рисунок 3.2).

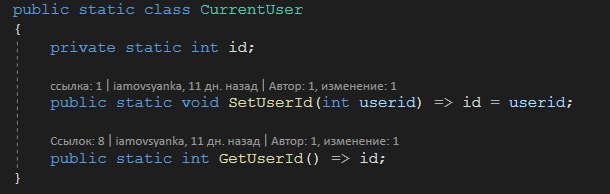


Рисунок 3.2 – Реализация класса CurrentUser

Класс CurrentTest используется для хранения идентификатора теста, который пользователь выбрал для прохождения (рисунок 3.3).

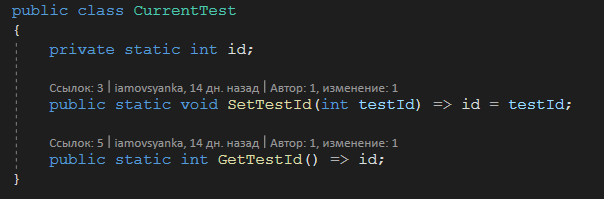


Рисунок 3.3 - Реализация класса CurrentTest

Класс Validation содержит 2 регулярных выражения, необходимых для валидации данных при регистрации и метод GetHashString, необходимый для шифрования пароля при получении и занесении в базу данных (рисунок 3.4).

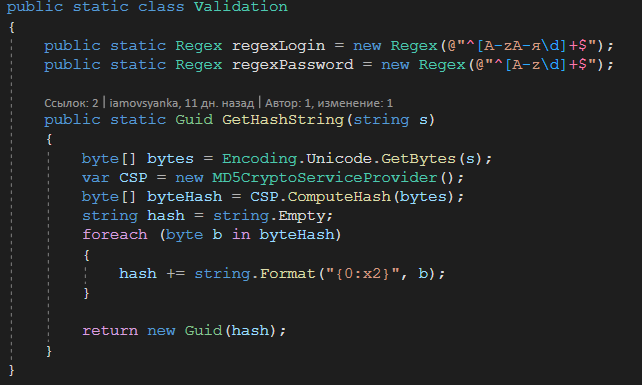


Рисунок 3.4 – Реализация класса Validation

## **3.3 Реализация работы с базой данных**

Для работы с базой данных были реализованы следующие паттерны: Repository и UnitOfWork.

Repository — это паттерн проектирования при работе с базами данных, который позволяет отделить конкретные подключения к источникам данных и остальной программой [4].

Диаграмма классов, реализующих паттерн Repository, представлена в приложении А.

Интерфейс репозитория IRepository<T> включает функционал:

* IQueryable<T> Get();
* Task AddAsync(T item);
* Task UpdateAsync(int itemId, T newItem);
* Task RemoveAsync(int itemId);

Эти методы представляют собой реализацию CRUD и предоставляет асинхронную работу с базой данной.

Для каждой модели сущности был создан класс, реализующий необходимые методы (рисунок 3.5).

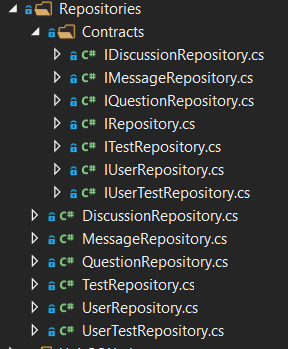


Рисунок 3.5 – Классы и интерфейсы Repository

UnitOfWork – это паттерн, который содержит набор репозиториев и ряд некоторых общих для них функций [4].

Реализация паттерна и диаграмма компонентов представлены в приложениях Б и В соответственно.

## **3.4 Реализация паттерна MVVM**

В курсовом проект реализован паттерн MVVM (Model-View-ViewModel), который позволяет отделить логику приложения от визуальной части. Данный паттерн является архитектурным, то есть он задает общую архитектуру приложения [5].

На рисунке 3.6 представлено размещение компонентов.

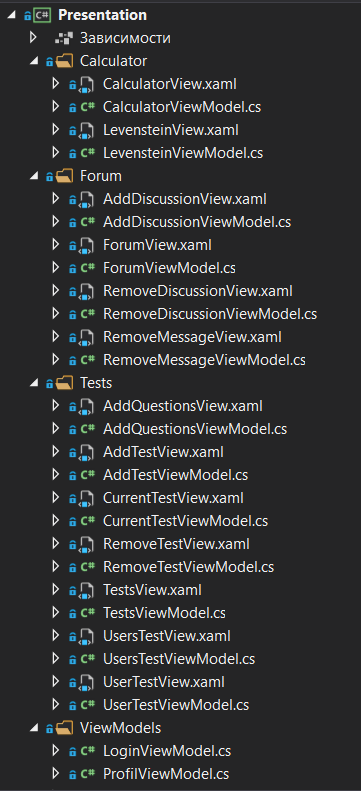


Рисунок 3.6 – Размещение компонентов по функциям

Все классы наследуют базовый класс ViewModelBase (рисунок 3.7), который реализует интерфейс INotifyPropertyChanged, уведомляющего клиента об изменении значения свойства.

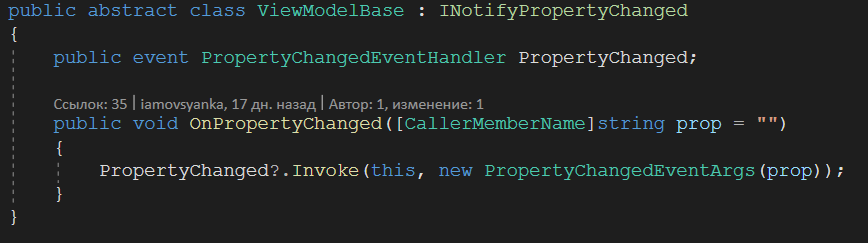


Рисунок 3.7 – Реализация класса ViewModelBase

Для взаимодействия пользователя и приложения в MVVM используются команды. С этой целью создан класс RelayCommand, реализующий интерфейс ICommand. Листинг находится в приложении Г.

Для каждой странички создан свой класс с суффиксом ViewModel. В данном разделе рассмотрен класс LoginViewModel, листинг которого представлен в приложении Д.

Класс содержит 2 команды:

* LoginCommand – команда для входа;
* GoToRegistrationCommand – команда для перехода на страницу регистрации.

Метод Login используется для авторизации пользователя. Если все необходимые данные введены, логин сверяется с базой данной. В случае обнаружения пользователя, сверяются пароли, иначе выводится сообщение о некорректности логина. Если пароль введён верно, открывается окно форума, в противном случае выводится сообщение о неправильном пароле. Диаграмма последовательности представлена в приложении Ж.

Метод GoToRegistration предназначен для перехода на страницу регистрации, если у пользователя ещё нет аккаунта.

Для взаимодействия со страницей используется Binding (рисунок 3.8).

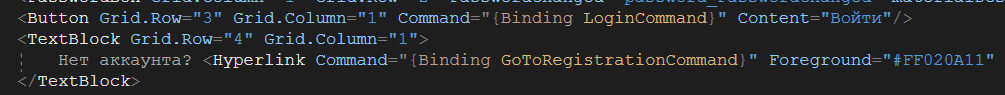


Рисунок 3.8 – XAML страницы авторизации

# **4 Тестирование**

При использовании программного средства могут возникнуть такие ситуации, когда введенные пользователем данные приведут к ошибке приложения.

В программе предусмотрена обработка возможных исключений, связанных с работой пользователя. В данном разделе продемонстрированы следующие тесты:

* авторизация и регистрация;
* отправка сообщения;
* использование калькулятора.

## **4.1 Тест авторизации**

При авторизации может возникнуть ситуация, когда пользователь не введёт необходимые данные. В таком случае выводится сообщение (рисунок 4.1, 4.2):

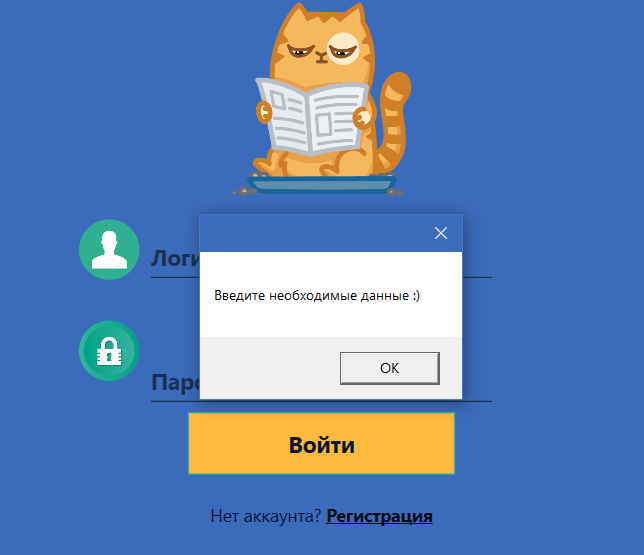


Рисунок 4.1 – Тест авторизации при не введённых данных

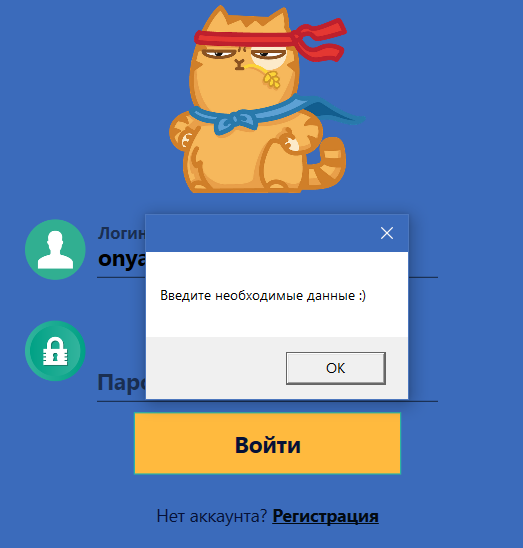


Рисунок 4.2 – Тест авторизации при не введённом пароле

При некорректном логине пользователю выводится следующее сообщение (рисунок 4.3):

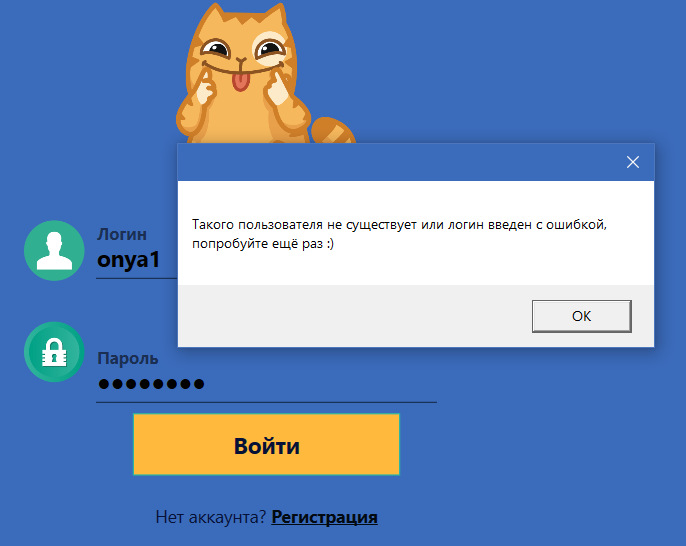


Рисунок 4.3 – Тест авторизации при некорректном логине

При некорректном пароле пользователю выводится сообщение (рисунок 4.4):

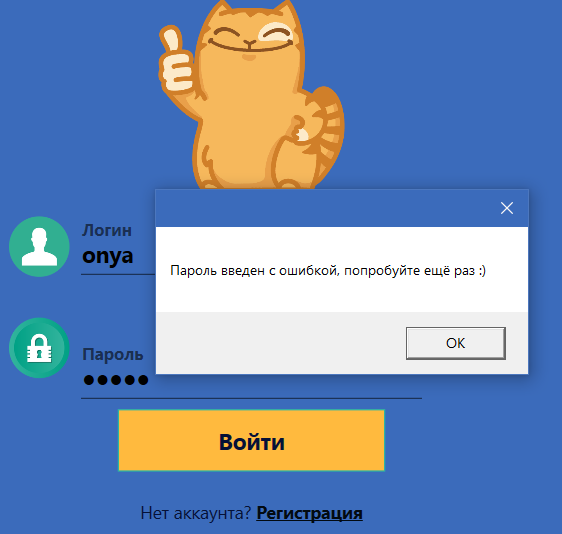


Рисунок 4.4 – Тест авторизации при некорректном пароле

При успешной авторизации пользователь попадает на страницу форума.

## **4.2 Тест регистрации**

При регистрации может возникнуть ситуация, когда пользователь не введёт необходимые данные, в этом случае, как и при авторизации, выводится сообщение (рисунок 4.5, 4.6):

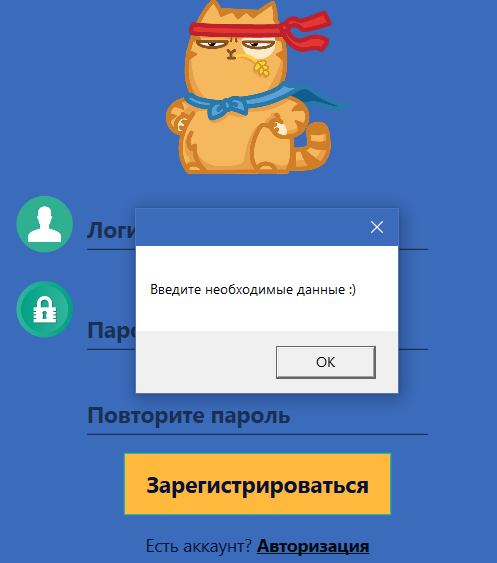


Рисунок 4.5 – Тест регистрации при не введённых данных

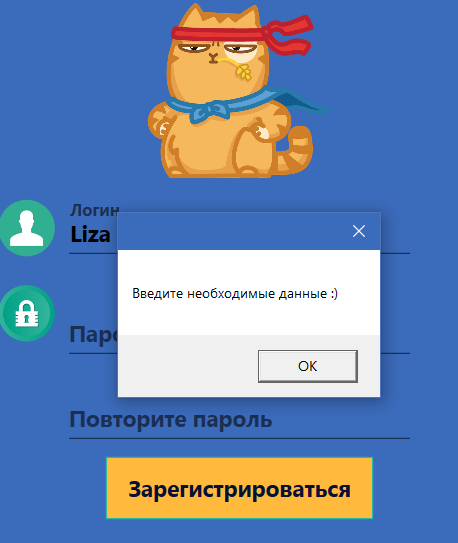


Рисунок 4.6 – Тест регистрации при не введённом пароле

Логин и пароль должны соответствовать заданным шаблонам: логин состоит из 4 и более символов латинского и русского алфавита и цифр, пароль состоит из 6 и более символов латинского алфавита и цифр (рисунок 4.7).



Рисунок 4.7 – Регулярные выражения для логина и пароля

При несоответствии введённых данных заданному шаблону выводится сообщения (рисунок 4.8 – 4.11):

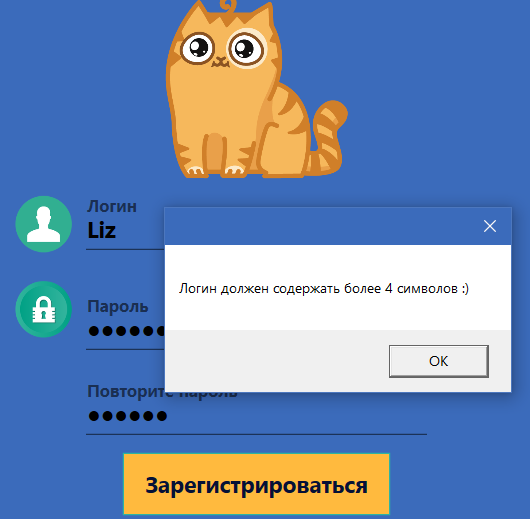


Рисунок 4.8 – Тест регистрации при некорректном логине

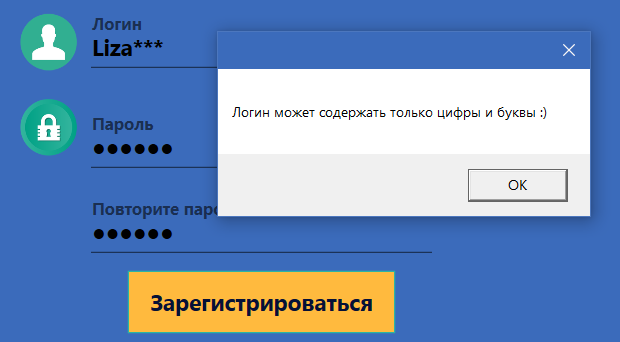


Рисунок 4.9 – Тест регистрации при некорректном логине

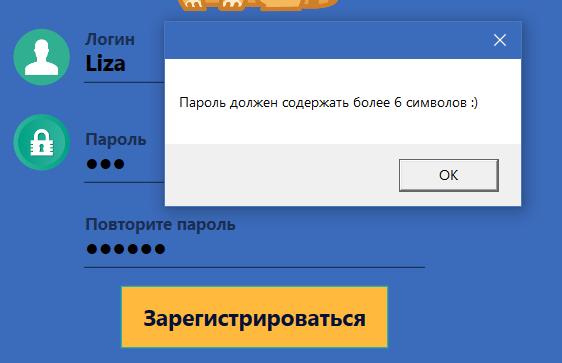


Рисунок 4.10 – Тест регистрации при некорректном пароле

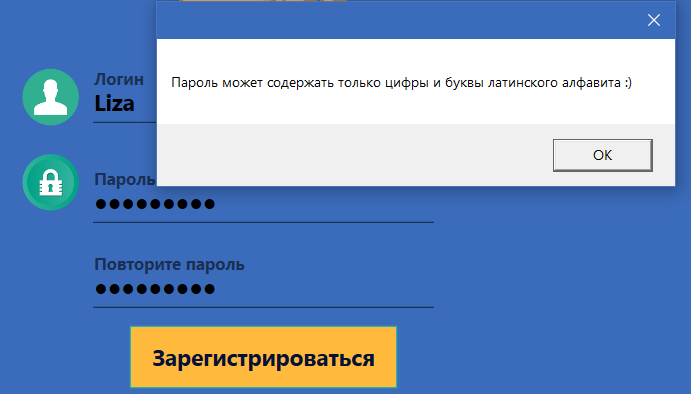


Рисунок 4.11 – Тест регистрации при некорректном пароле

Для регистрации необходимо подтверждение пароля и при несовпадении выводится следующее сообщение (рисунок 4.12):

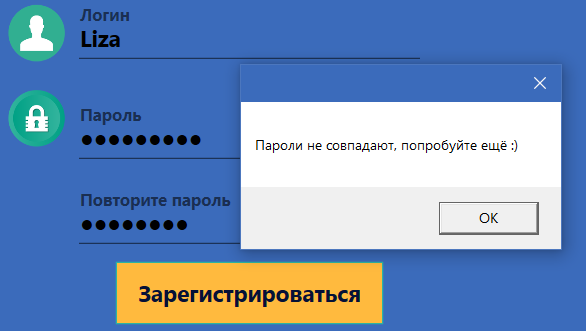


Рисунок 4.12 – Тест регистрации при несовпадении паролей

При корректных данных выводится сообщение об успешной регистрации, и пользователь попадает на страницу авторизации.

## **4.3 Тест отправки сообщения**

При работе с форумом возможны две ситуации: пользователь пытается отправить сообщение, не выбрав обсуждение, или сообщение пустое. В таких случаях выводятся сообщения (рисунок 4.13, 4.14):

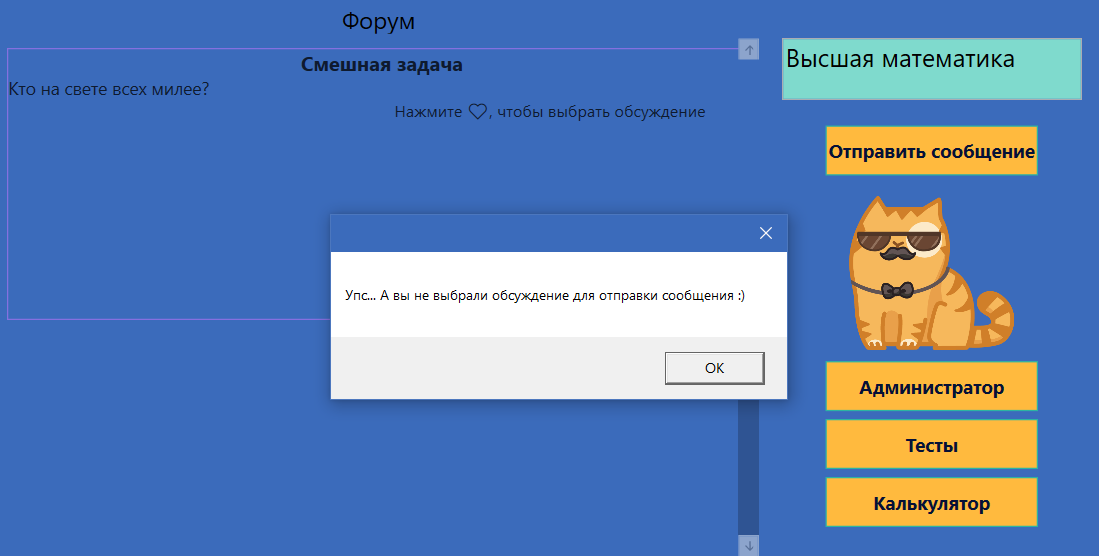
****

Рисунок 4.13 – Тест отправки сообщения при не выбранном обсуждении

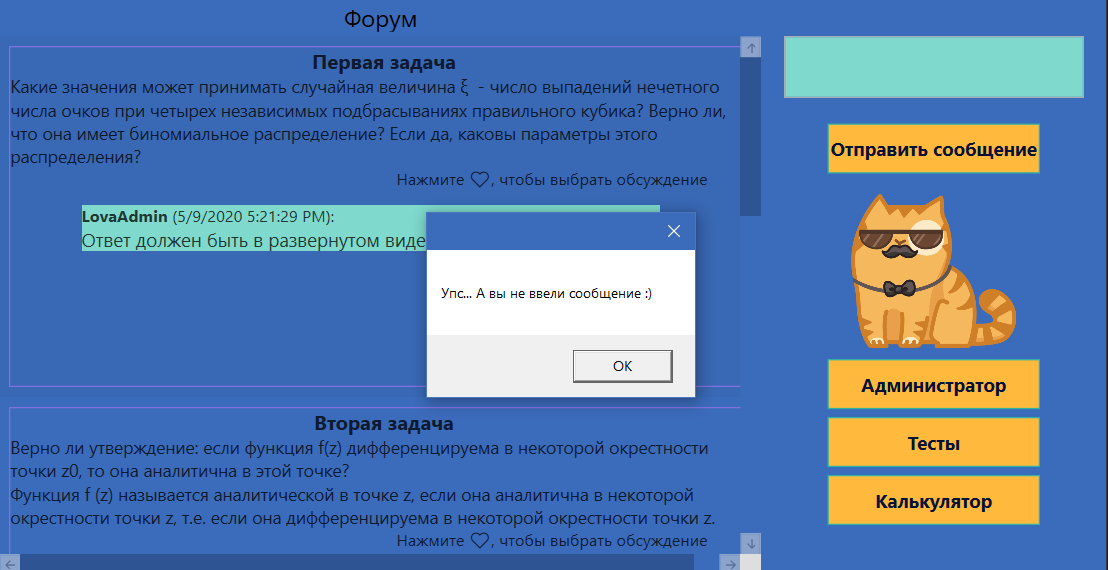
****

Рисунок 4.14 – Тест отправки сообщения при пустом ответе

## **4.4 Тест использования калькулятора**

Для тестирования калькулятора был создан проект XUnitTestProject для модульного тестирование на базе .Net Core. В нём содержатся два класса: TestCalculator и TestLevenstein. Листинги кода находятся в приложениях З, И. соответственно.

На рисунке 4.15 продемонстрированы успешно пройденные тесты.

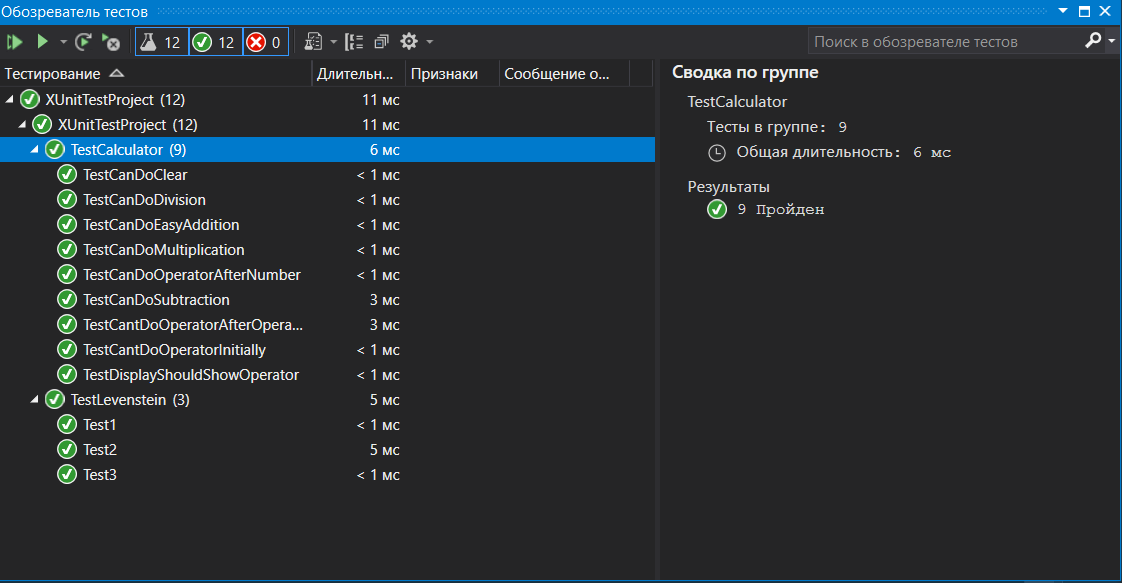


Рисунок 4.15 – Тесты калькулятора

# **ЗАКЛЮЧЕНИЕ**

В ходе выполнения курсового проекта был изучен теоретический материал по теме проекта, рассмотрены аналоги, выявлены их преимущества и недостатки.

Для реализации программного средства были изучены и применены технологии такие, как WPF, Entity Framework Core, Microsoft SQL Server. Само приложение было реализовано на языке C# в среде Visual Studio. Для структуры проекта применён архитектурный паттерн MVVM, для работы с базой данной – Repository и UnitOfWork.

Цель курсового проекта – создание приложения «Lova» для организации дистанционного обучения по предмету «Высшая математика, выполнена в полном объёме. С помощью этого приложения преподаватель может создавать обсуждения и тесты, для оценивания уровня знаний студентов и дальнейшего планирования учебного процесса.

Главной задачей являлось создание максимально удобного и понятного приложения, для этого была спроектирована структура окон для комфортного использования.

Функционально проект поддерживает:

* Регистрацию и авторизацию;
* Прохождение тестов;
* Отправку сообщений в обсуждение поставленной задачи;
* Поиск теста по названию;
* Получение результатов тестов;
* Использование встроенного калькулятора
* Создание и удаление тестов;
* Создание и удаление обсуждений.

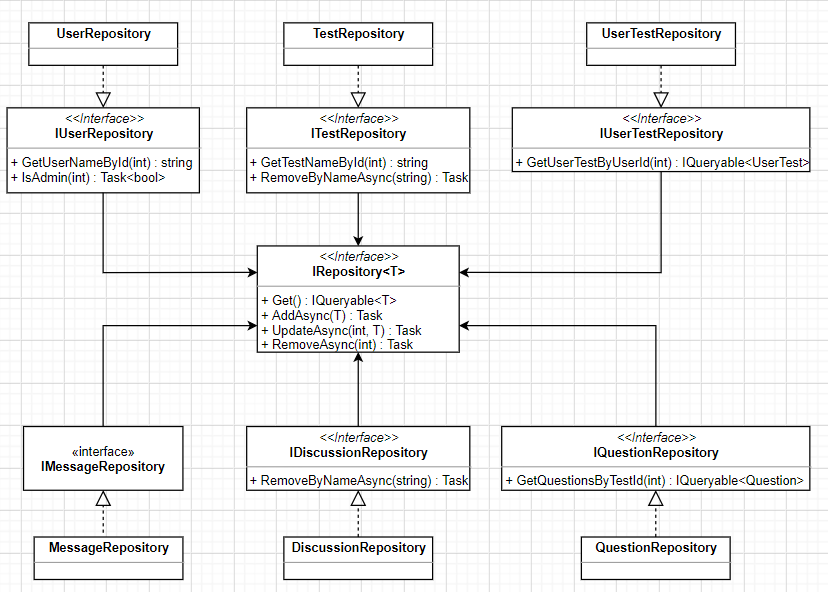
Все функции протестированы и выполняются корректно.

# **Список используемых источников**

1. Натан А. WPF 4. Подробное руководство. - Пер. с англ. - СПб.: Символ-Плюс, 2011. - 880 с.
2. Сайт о программировании Metanit: Введение в Entity Framework Core [Электронный ресурс] / metanit.com – Режим доступа: https://metanit.com/sharp/entityframeworkcore/1.1.php
3. Сайт о программировании Metanit: Введение в MS SQL Server [Электронный ресурс] / metanit.com – Режим доступа: <https://metanit.com/sql/sqlserver/1.1.php>
4. Пацей, Н.В. Курс лекций по языку программирования C# / Н.В. Пацей. – Минск: БГТУ, 2016. – 175 с.
5. Сайт о программировании Metanit: Определение паттерна MVVM [Электронный ресурс] / metanit.com – Режим доступа: <https://metanit.com/sharp/wpf/22.1.php>
6. Мэтью Макдональд – WPF: Windows Presentation Foundation в .NET 4.0 с примерами на C# / М. Макдональд. – Вильямс, 2011. – 1024 с.

# **ПРИЛОЖЕНИЕ А**

Диаграмма классов, реализующих паттерн Repository.



# **ПРИЛОЖЕНИЕ Б**

Листинг 1. Реализация класса UnitOfWork.

using Models.Contexts;

using Models.Repositories;

using System;

namespace Models.UnitOfWork

{

public class UnitOfWork : IDisposable

{

private LovaContext lovaContext = new LovaContext();

private DiscussionRepository discussionRepository;

private MessageRepository messageRepository;

private QuestionRepository questionRepository;

private TestRepository testRepository;

private UserRepository userRepository;

private UserTestRepository userTestRepository;

public DiscussionRepository DiscussionRepository

{

get

{

if (discussionRepository == null)

{

discussionRepository= new DiscussionRepository(lovaContext);

}

return discussionRepository;

}

}

public MessageRepository MessageRepository

{

get

{

if (messageRepository == null)

{

messageRepository = new MessageRepository(lovaContext);

}

return messageRepository;

}

}

public QuestionRepository QuestionRepository

{

get

{

if (questionRepository == null)

{

questionRepository = new QuestionRepository(lovaContext);

}

return questionRepository;

}

}

public TestRepository TestRepository

{

get

{

if (testRepository == null)

{

testRepository = new TestRepository(lovaContext);

}

return testRepository;

}

}

public UserRepository UserRepository

{

get

{

if (userRepository == null)

{

userRepository = new UserRepository(lovaContext);

}

return userRepository;

}

}

public UserTestRepository UserTestRepository

{

get

{

if (userTestRepository == null)

{

userTestRepository = new UserTestRepository(lovaContext);

}

return userTestRepository;

}

}

public void Save() => lovaContext.SaveChanges();

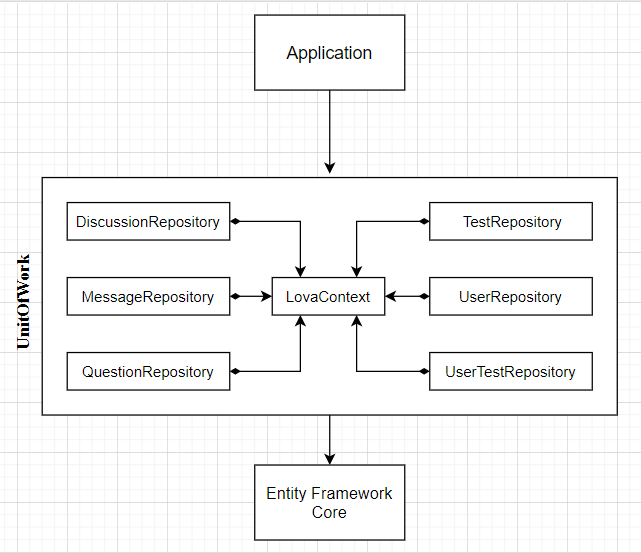
public void Dispose() => lovaContext.Dispose();

}

}

# **ПРИЛОЖЕНИЕ В**

Диаграмма компонентов с использованием паттерна Repository в связке с UnitOfWork паттерном.



# **ПРИЛОЖЕНИЕ Г**

Листинг 2. Реализация класса RelayCommand.

using System;

using System.Windows.Input;

namespace Models.Commands

{

public class RelayCommand : ICommand

{

private Action<object> execute;

private Func<object, bool> canExecute;

public event EventHandler CanExecuteChanged

{

add { CommandManager.RequerySuggested += value; }

remove { CommandManager.RequerySuggested -= value; }

}

public RelayCommand(Action<object> execute, Func<object, bool> canExecute = null)

{

this.execute = execute;

this.canExecute = canExecute;

}

public bool CanExecute(object parameter) => canExecute == null || canExecute(parameter);

public void Execute(object parameter) => execute(parameter);

}

}

# **ПРИЛОЖЕНИЕ Д**

Листинг 3. Реализация класса LoginViewModel.

using Lova;

using Models.Commands;

using Models.Current;

using Models.UnitOfWork;

using Models.Validation;

using Presentation.Views;

using System;

using System.Linq;

using System.Windows;

using System.Windows.Input;

namespace Presentation.ViewModels

{

public class LoginViewModel : ViewModelBase

{

private readonly UnitOfWork unitOfWork;

private string userName;

private string password;

public ICommand LoginCommand => new RelayCommand(obj => Login());

public ICommand GoToRegistrationCommand => new RelayCommand(obj => GoToRegistration());

public LoginViewModel()

{

unitOfWork = new UnitOfWork();

}

public string UserName

{

get => userName;

set

{

userName = value;

OnPropertyChanged("UserName");

}

}

public string Password

{

private get => password;

set

{

password = value;

OnPropertyChanged("Password");

}

}

public string Img

{

get

{

var random = new Random();

switch (random.Next(6))

{

case 0:

return "..\\Views\\Resource\\Image\\User0.png";

case 1:

return "..\\Views\\Resource\\Image\\User1.png";

case 2:

return "..\\Views\\Resource\\Image\\User2.png";

case 3:

return "..\\Views\\Resource\\Image\\User3.png";

case 4:

return "..\\Views\\Resource\\Image\\User4.png";

case 5:

return "..\\Views\\Resource\\Image\\User5.png";

default:

return "..\\Views\\Resource\\Image\\User0.png";

}

}

}

private void Login()

{

if (string.IsNullOrEmpty(UserName) || string.IsNullOrEmpty(Password))

{

MessageBox.Show("Введите необходимые данные :)");

}

else

{

var currentUser = unitOfWork.UserRepository.Get().FirstOrDefault(user => user.UserName == UserName);

if (currentUser != null)

{

if (Validation.GetHashString(Password) == currentUser.Password)

{

CurrentUser.SetUserId(currentUser.Id);

App.ForumPage = new ForumView();

App.ProfilViewModel.CurrentPage = App.ForumPage;

}

else

{

MessageBox.Show("Пароль введен с ошибкой, попробуйте ещё раз :)");

}

}

else

{

MessageBox.Show("Такого пользователя не существует или логин введен с ошибкой, попробуйте ещё раз :)");

}

}

}

private void GoToRegistration()

{

App.RegistrationPage = new RegistrationView();

App.ProfilViewModel.CurrentPage = App.RegistrationPage;

}

}

}

# **ПРИЛОЖЕНИЕ Е**

Листинг 4. Реализация класса LovaContext.

public class LovaContext : DbContext

{

public DbSet<User> Users { get; set; }

public DbSet<Test> Tests { get; set; }

public DbSet<Discussion> Discussions { get; set; }

public DbSet<Message> Messages { get; set; }

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

public DbSet<UserTest> UserTests { get; set; }

public LovaContext()

{

}

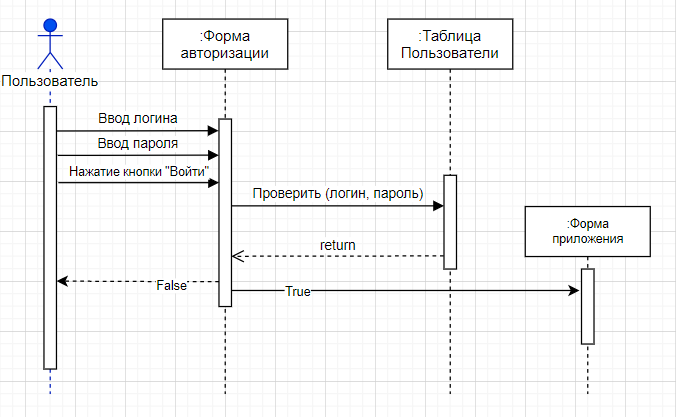
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder) => optionsBuilder.UseSqlServer("Server=LAPTOP-ADTTMHK8\\SQLEXPRESS;Database=LovaApp;Trusted\_Connection=True;");

protected override void OnModelCreating(ModelBuilder modelBuilder) => base.OnModelCreating(modelBuilder);

}

# **ПРИЛОЖЕНИЕ Ж**

Диаграмма последовательности.



# **ПРИЛОЖЕНИЕ З**

Листинг 5. Код тестирования TestCalculator.

using Calculator;

using Xunit;

namespace XUnitTestProject

{

public class TestCalculator

{

private CalculatorModel calculator = new CalculatorModel();

[Fact]

public void TestCantDoOperatorInitially()

{

Assert.False(calculator.CanDoSign());

}

[Fact]

public void TestCanDoOperatorAfterNumber()

{

calculator.Number(5);

Assert.True(calculator.CanDoSign());

}

[Fact]

public void TestCantDoOperatorAfterOperator()

{

calculator.Number(5);

calculator.Plus();

Assert.False(calculator.CanDoSign());

}

[Fact]

public void TestCanDoClear()

{

Assert.True(calculator.CanDoClear());

}

[Fact]

public void TestDisplayShouldShowOperator()

{

calculator.Number(1);

calculator.Plus();

Assert.Equal("1 + ", calculator.Display);

}

[Fact]

public void TestCanDoSubtraction()

{

calculator.Number(1);

calculator.Number(2);

calculator.Minus();

calculator.Number(1);

calculator.Number(3);

calculator.Equals();

Assert.Equal(-1, calculator.Result);

}

[Fact]

public void TestCanDoEasyAddition()

{

calculator.Number(4);

calculator.Plus();

calculator.Number(3);

calculator.Equals();

Assert.Equal(7, calculator.Result);

}

[Fact]

public void TestCanDoMultiplication()

{

calculator.Number(2);

calculator.Number(0);

calculator.Times();

calculator.Number(3);

calculator.Number(0);

calculator.Equals();

Assert.Equal(600, calculator.Result);

}

[Fact]

public void TestCanDoDivision()

{

calculator.Number(4);

calculator.Over();

calculator.Number(1);

calculator.Number(0);

calculator.Equals();

Assert.Equal(0.4, calculator.Result);

}

}

}

# **ПРИЛОЖЕНИЕ И**

Листинг 6. Код тестирования TestLevenstein.

using Calculator;

using Xunit;

namespace XUnitTestProject

{

public class TestLevenstein

{

[Fact]

public void Test1()

{

Assert.Equal(3, Levenstein.LevensteinDynamic("Вар", "Баран"));

}

[Fact]

public void Test2()

{

Assert.Equal(4, Levenstein.LevensteinDynamic("test", ""));

}

[Fact]

public void Test3()

{

Assert.Equal(4, Levenstein.LevensteinDynamic("", "test"));

}

}

}