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

Направление специальности 1-40 01 01 10 Программное обеспечение информационных технологий (программирование интернет-изданий)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине «Объектно-ориентированные технологии и стандарты проектирования»

Тема «Студенты университета»

Исполнитель

студент 2 курса группы 5 Сураго Дмитрий Александрович

Руководитель работы ассистент Северинчик Н.А.

Курсовой проект защищён с оценкой

Председатель Пацей Н.В.

Минск 2020

**Оглавление**

[**ВВЕДЕНИЕ** 3](#_Toc41224721)

[**1.** **Обзор аналогов** 4](#_Toc41224722)

[**2.** **Анализ требований к программному средству и разработка функциональных требований** 8](#_Toc41224723)

[**2.1 Функциональные требования** 8](#_Toc41224724)

[**2.2 Спецификация функциональных требований** 8](#_Toc41224725)

[**3.** **Проектирование программного средства** 10](#_Toc41224726)

[**3.1 Модель базы данных** 10](#_Toc41224727)

[**3.2 UML-диаграммы** 12](#_Toc41224728)

[**3.3 Структура проекта** 12](#_Toc41224729)

[**4.** **Реализация программного средства** 13](#_Toc41224730)

[**5.** **Тестирование, проверка работоспособности и анализ полученных результатов** 18](#_Toc41224731)

[**6.** **Руководство пользователя по использованию** 23](#_Toc41224732)

[**ЗАКЛЮЧЕНИЕ** 26](#_Toc41224741)

[**СПИСОК ЛИТЕРАТУРЫ** 27](#_Toc41224742)

[**ПРИЛОЖЕНИЕ А** 28](#_Toc41224743)

[**ПРИЛОЖЕНИЕ Б** 28](#_Toc41224745)

[**ПРИЛОЖЕНИЕ В** 29](#_Toc41224746)

[**ПРИЛОЖЕНИЕ Г** 29](#_Toc41224748)

[**ПРИЛОЖЕНИЕ Д** 30](#_Toc41224750)

[**ПРИЛОЖЕНИЕ E** 31](#_Toc41224752)

[**ПРИЛОЖЕНИЕ Ж** 33](#_Toc41224756)

[**ПРИЛОЖЕНИЕ З** 33](#_Toc41224758)

[**ПРИЛОЖЕНИЕ И** 34](#_Toc41224759)

**ВВЕДЕНИЕ**

Для того, чтобы проверить усвоение материала студентами, уведомить их о важных новостях или дать им нужную литературу, студенты обязаны были ходить в университет и тратить свое время. Так же преподавателям приходилось каждый раз проверять каждый ответ студента и выставлять соответствующую оценку.

Поэтому для того, чтобы не тратить драгоценное время студентов и избавить преподавателей от хлопот, разрабатываются приложения, которые ускоряют процесс обучения, а так же делают его удобным.

Данные приложения позволяют студентам с легкостью проверять свои знания, читать нужную литературу, а преподаватели могут забыть о работе с бумажными носителями.

Целью данного курсового проекта является разработка ПО, осуществляющего авторизацию и аутентификацию пользователей, а так же поиска нужной информации по заданным предметам, прохождения тестов, просмотра важных новостей для студентов, добавления тестов и нужной литературы для своего предмета, отслеживания успеваемости студентов для преподавателей.

1. **Обзор аналогов**

В качестве первого аналога было выбрано программное средство «Учет в школе». У себя в приложении я использовал основные возможности аналога.

На рисунке 1.1. представлена возможность регистрирования новых учетных записей.

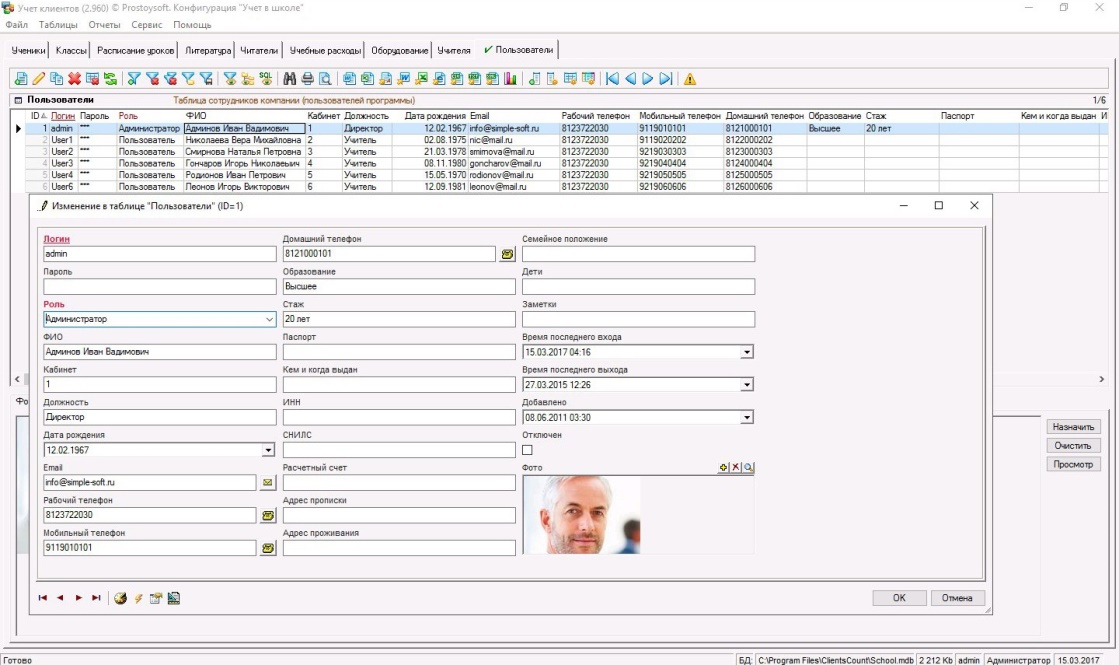


Рисунок 1.1 – Регистрирование пользователей

Также на рисунке 1.2 представлена возможность программного средства просматривать информацию об учащихся с возможностью их редактирования.

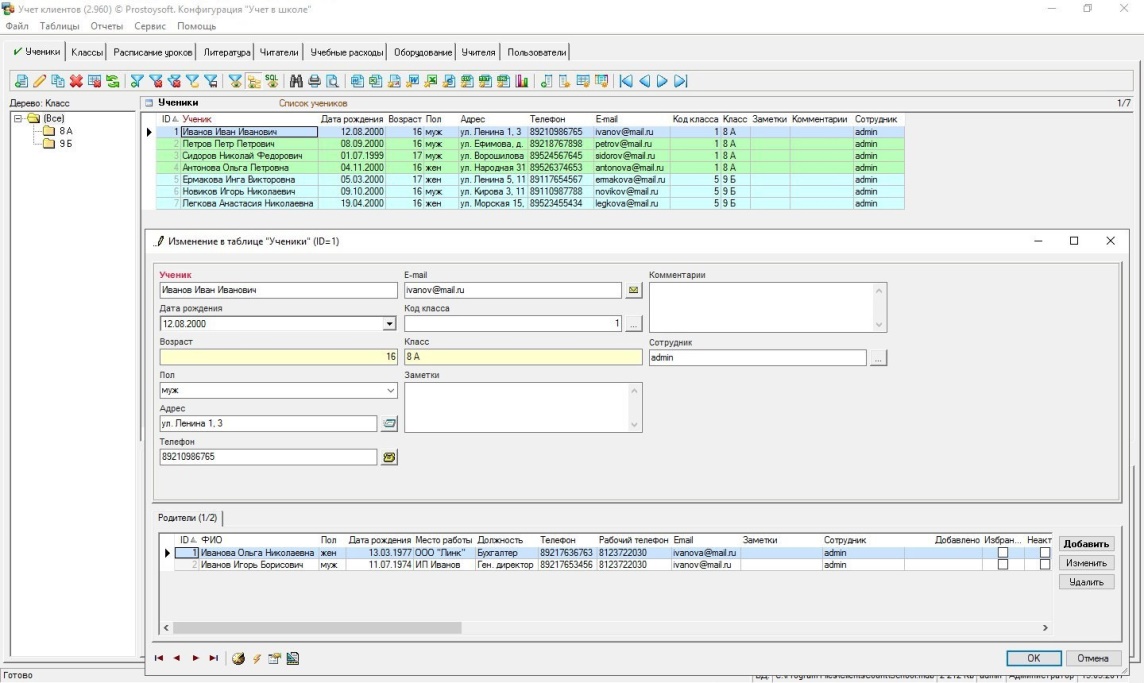


Рисунок 1.2 – Списки учащихся

Программное средство включает просмотр разной литературы для необходимого предмета. Это продемонстрировано на рисунке 1.3.

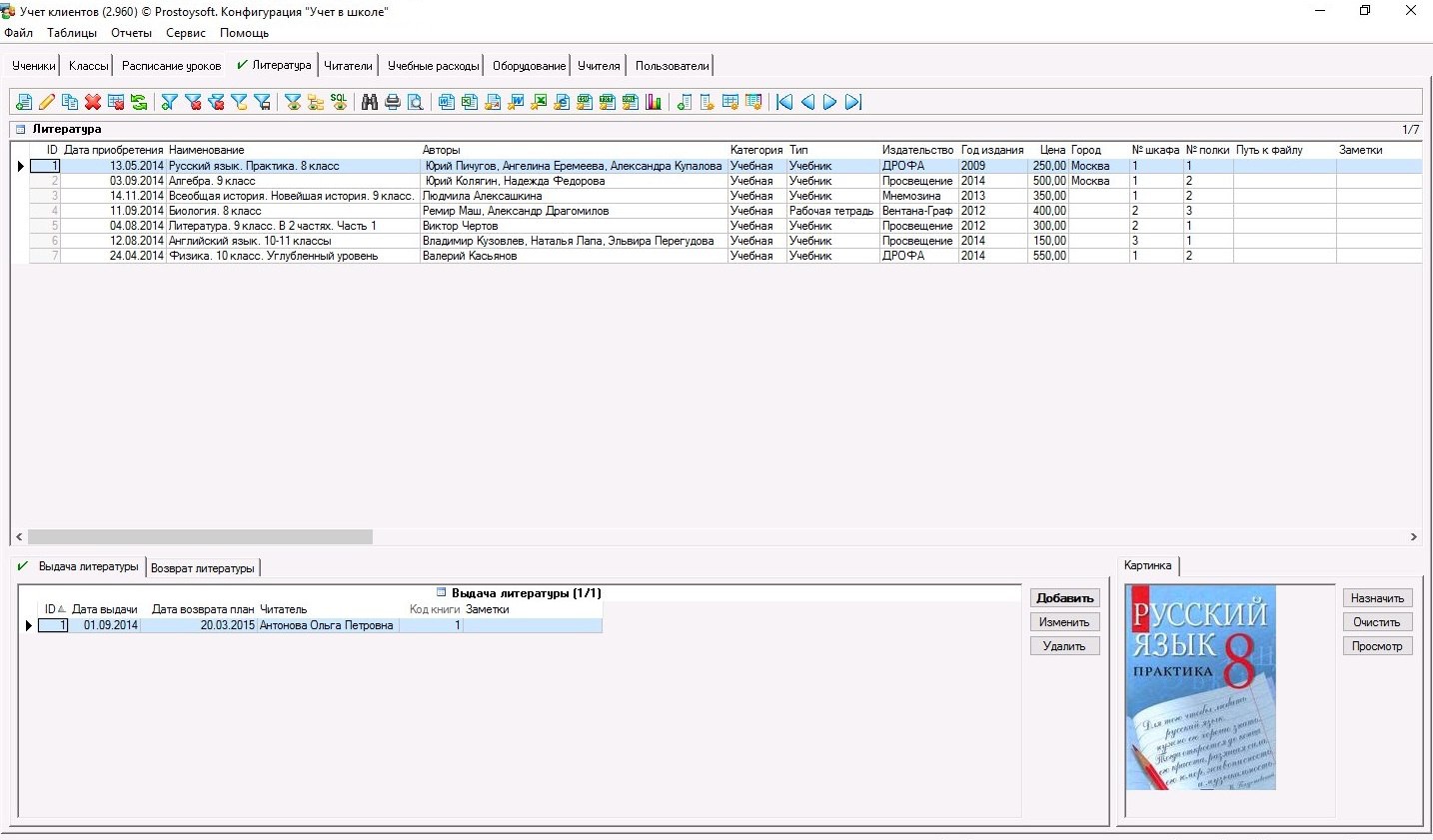


Рисунок 1.3 – Просмотр литературы

На рисунке 1.4 продемонстрирована возможность отправлять сообщения на E-mail или на мобильный телефон.

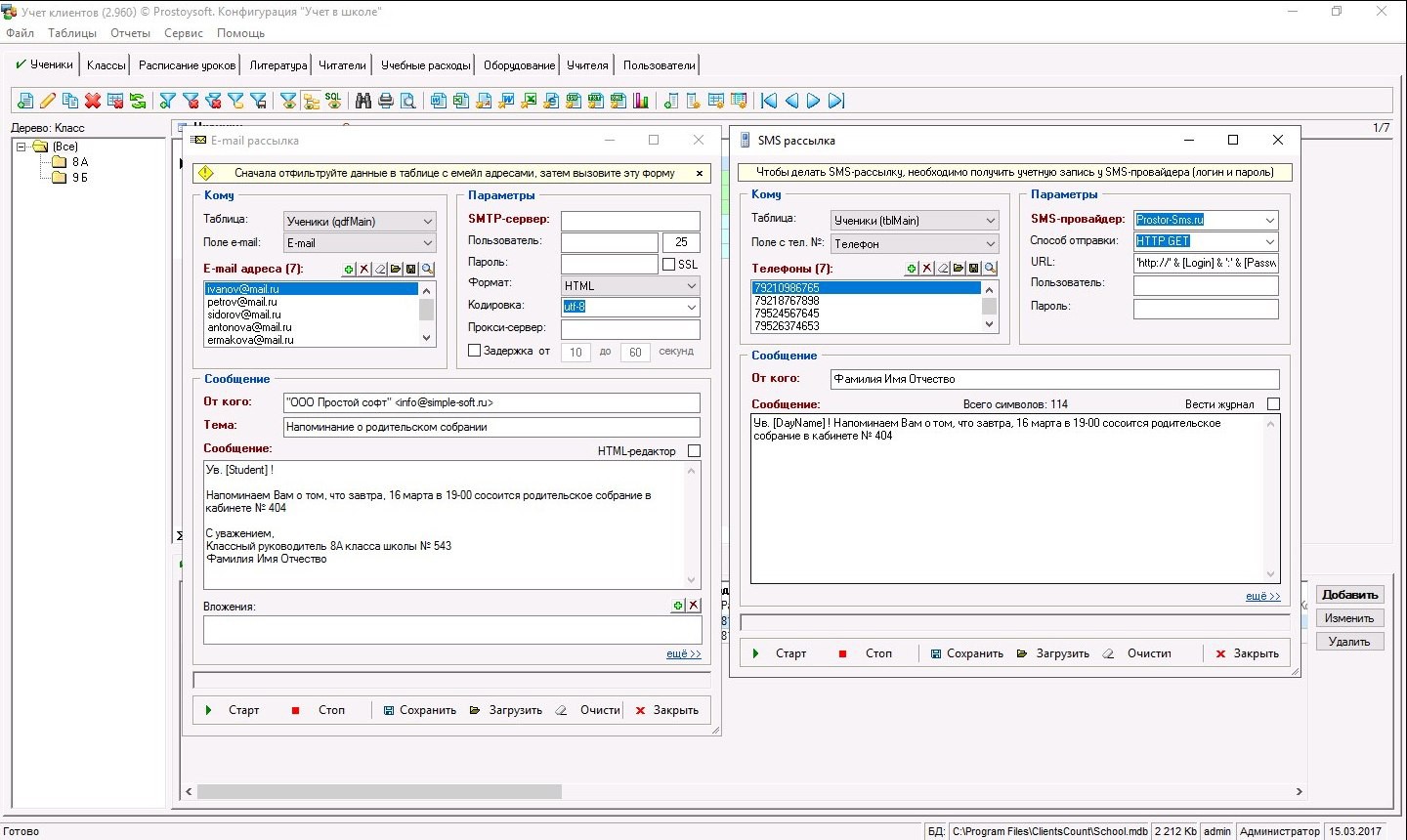


Рисунок 1.4 – E-mail и SMS рассылка

В качестве второго аналога было выбрано приложение «СДО БГТУ».

На рисунке 1.5 представлена возможность авторизации.

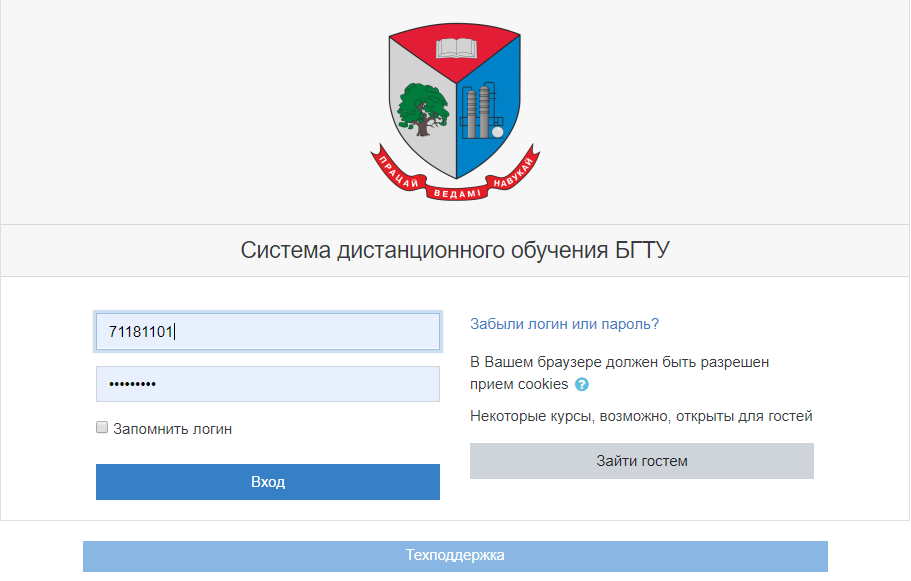


Рисунок 1.5 – Авторизация

Главное меню представлено на рисунке 1.6.

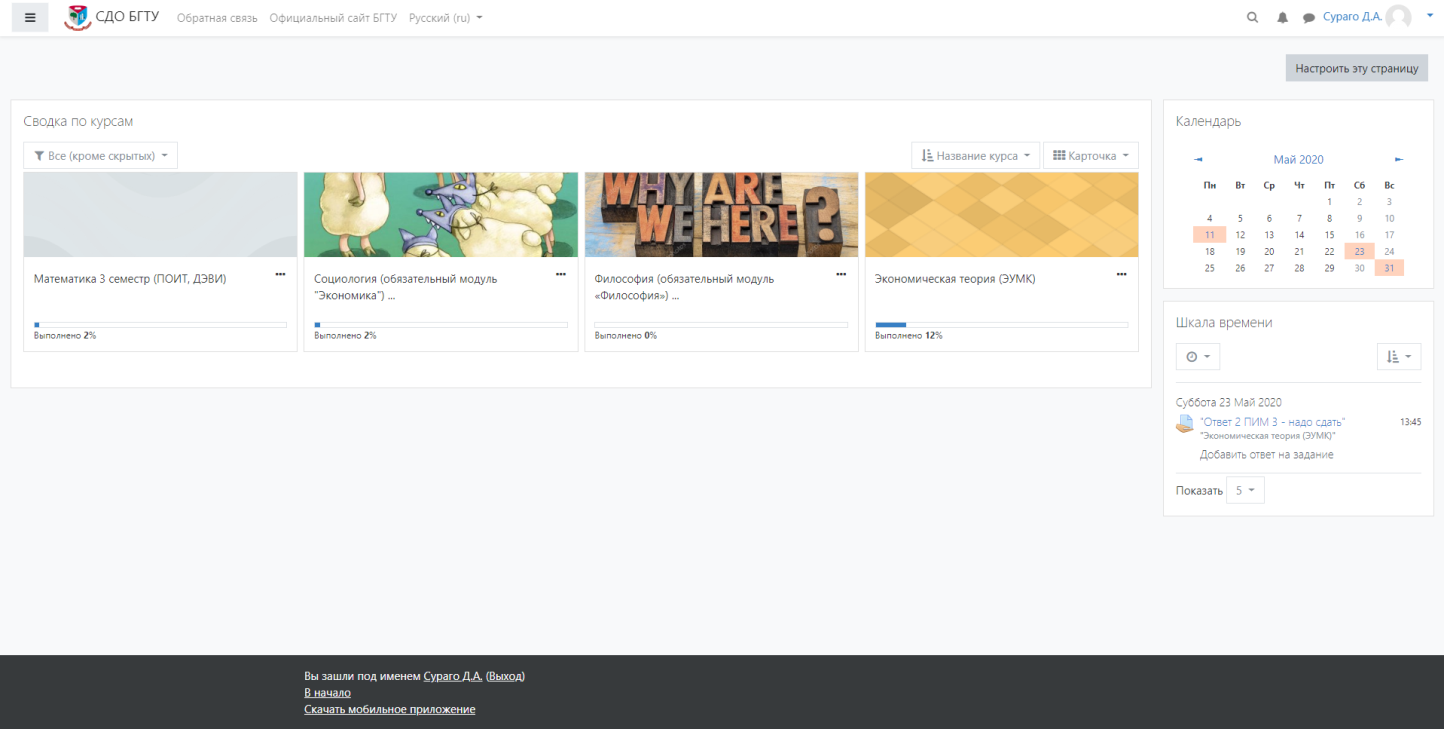


Рисунок 1.6 – Главное меню

Если кликнуть по нужной дисциплине, то мы попадем на страницу с необходимой литературой и заданиями. На рисунке 1.7 представлены задания для дисциплины «Математика».

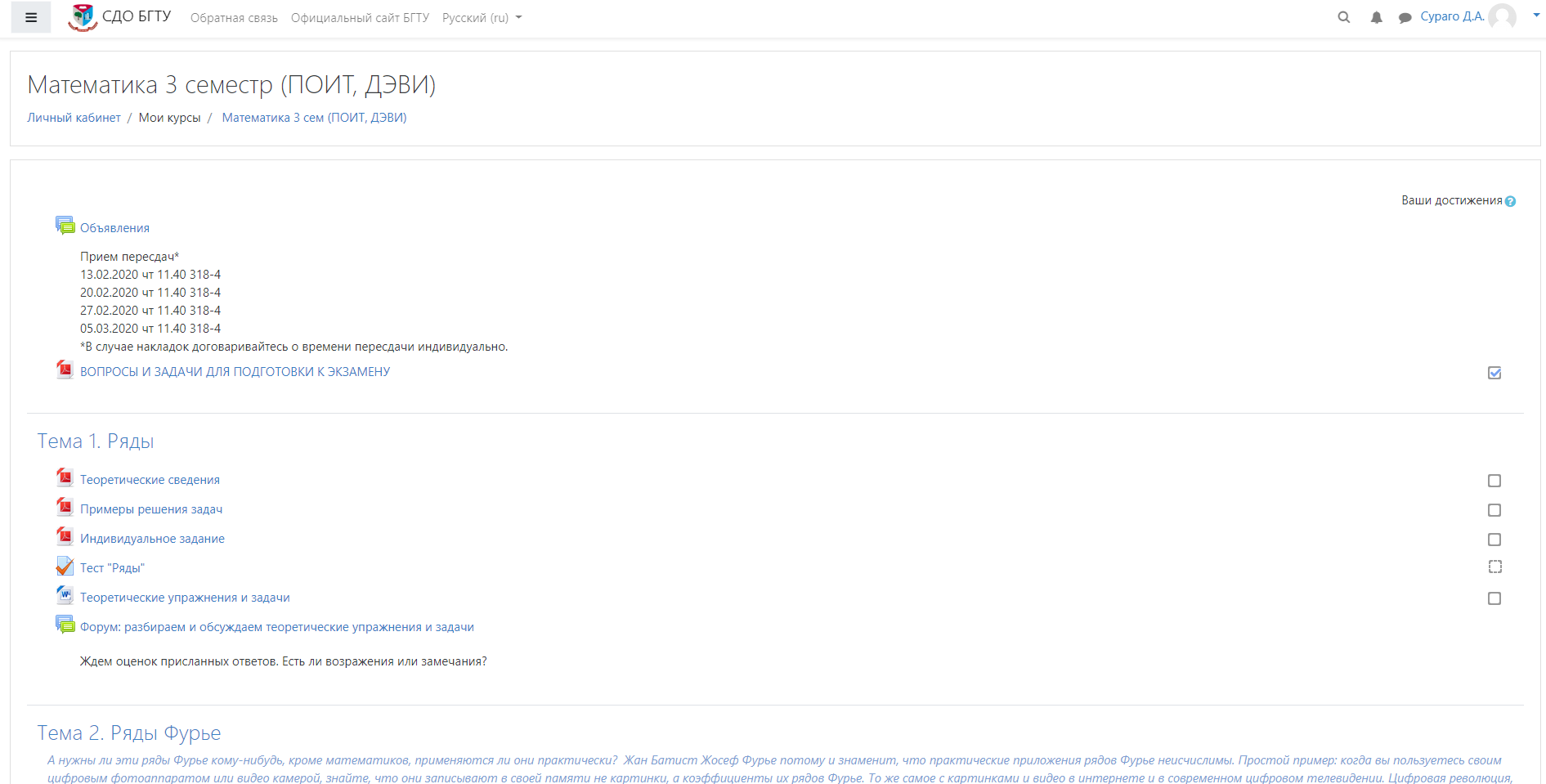


Рисунок 1.7 – раздел «Математика»

Проанализировав аналоги, я решил использовать следующий функционал:

Возможность регистрирования новых пользователей, так же их авторизация, просмотр списка студентов с возможностью их редактирования, поиск литературы по заданным предметам, E-mail рассылка при восстановлении пароля, тестирование студентов по разным предметам.

1. **Анализ требований к программному средству и разработка функциональных требований**

**2.1 Функциональные требования**

Рассмотрев аналоги, были представлены следующие функциональные требования:

Администратор:

- регистрация учетных записей для студентов;

- регистрация учетных записей для преподавателей;

- редактирование информации о студентах;

- добавление новостей.

Преподаватель:

- просмотр списка студентов по своему предмету;

- добавление и удаление тестов;

- добавление необходимой литературы для своего предмета.

Студент:

- восстановление учетной записи;

- просмотр своей успеваемости;

- просмотр важных новостей;

- поиск необходимой литературы по выбранному предмету;

- прохождение теста по разным предметам;

- просмотр своего профиля (информации о себе);

- изменение пароля.

**2.2 Спецификация функциональных требований**

Задачей курсового проекта является разработка приложения, предназначенного для осуществления работы спортзала, а также ряда дополнительных операций: выдачи программы тренировок, поиска, получение информации о тренировках и оставления личных отметок. Абонементы и программы тренировок выдаются непосредственно с самого приложения администратором/модератором.

Программный продукт должен быть реализовать на объектно-ориентированном языке программирования C# с использованием технологии WPF. Windows Presentation Foundation (WPF) — система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем, графическая (презентационная) подсистема в составе .NET Framework (начиная с версии 3.0), использующая язык XAML.

В основе WPF лежит векторная система визуализации, не зависящая от разрешения устройства вывода и созданная с учётом возможностей современного графического оборудования. WPF предоставляет средства для создания визуального интерфейса, включая язык XAML (eXtensible Application Markup Language), элементы управления, привязку данных, макеты, двухмерную и трёхмерную графику, анимацию, стили, шаблоны, документы, текст, мультимедиа и оформление. База данных была разработана в Microsoft SQL Server.

SQL Server является одной из наиболее популярных систем управления базами данных (СУБД) в мире.

SQL Server характеризуется такими особенностями как:

* Производительность. SQL Server работает очень быстро.
* Надежность и безопасность. SQL Server предоставляет шифрование данных.
* Простота. С данной СУБД относительно легко работать и вести администрирование.

Для организации баз данных MS SQL Server использует реляционную модель, которая предполагает хранение данных в виде таблиц, каждая из которых состоит из строк и столбцов. Каждая строка хранит отдельный объект, а в столбцах размещаются атрибуты этого объекта.

Для взаимодействия с базой данных применяется язык SQL (Structured Query Language). Клиент (например, внешняя программа) отправляет запрос на языке SQL должным образом интерпретирует и выполняет запрос, а затем посылает клиенту результат выполнения.

Основной используемый язык запросов —Transact-SQL — реализован на структурированном языке запросов (SQL) с расширениями. Для работы с базой данных была взята СУБД Microsoft SQL Server и была связана с приложением при помощи технологии ADO.NET.

ADO.NET представляет 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.

1. **Проектирование программного средства**

**3.1 Модель базы данных**

Для реализации данного программного средства была создана база данных Desktop.

Структура базы данных представлена в Приложении А.

Таблица «ADMIN» хранит данные учетной записи администратора, она представлена следующими полями:

* LOGIN (логин)
* APASS (пароль)

Таблица «NEWS» хранит информацию о новостях, состоит их полей:

* NAME (заголовок)
* DESCRIPTION (описание)
* PICTURE (прикрепленная картинка)
* NAMEPICTURE (имя картинки)
* DATE (дата добавления новости)

Таблица «PROFESSION» хранит информацию о специальностях, представлена полями:

* PROFESSION (название специальности)

Таблица «GROUPS» хранит информацию о группах, она состоит из полей:

* IDGROUP (номер группы)
* PROFESSION (название специальности)

Таблица «STUDENT» содержит информацию о студентах, здесь имеются следующие поля:

* RECORD (номер зачетки)
* SPASS (пароль)
* NAME (имя и фамилия)
* PICTURE (картинка)
* IDGROUP (номер группы)
* COURSE (курс)
* NAMEPICTURE (название картинки)

Таблица «PROGRESS» хранит отметки студента за пройденные тесты, она состоит из полей:

* SUBJECT (предмет)
* IDSTUDENT (номер зачетки)
* NOTE (отметка)

Таблица «TESTS» содержит вопросы и ответы к тестам по разным предметам, она содержит следующие поля:

* SUBJECT (предмет)
* QUESTION (вопрос)
* ANSWER (ответ)
* NOANSWER1 (неверный ответ 1)
* NOANSWER2 (неверный ответ 2)

Таблица «TEACHER» хранит информацию о преподавателях, состоит из:

* TEACHER (логин)
* TPASS (пароль)
* SUBJECT (предмет)
* TEACHERNAME (имя и фамилия)

Таблица «COURSE» содержит все курсы, содержит поля:

* COURSE (курс)

Таблица «SUBJECT» содержит все предметы, включает поля:

* SUBJECT (предмет)
* SUBJECT\_NAME (название предмета)
* COURSE (курс)

Таблица «LITERATURE» содержит всю литературу, содержит поля:

* TITLE (заголовок)
* AUTHORS (авторы)
* SUBJECT (предмет)
* PICTURE (прикрепленная картинка)

**3.2 UML-диаграммы**

Диаграмма с функционалом пользователей представлена в Приложении Б.

В Приложении В представлена диаграмма деятельности для всех возможностей преподавателя.

Диаграмма, представленная в Приложении Г, – диаграмма последовательности для добавления новости администратором.

**3.3 Структура проекта**

Структурное дерево проекта представлено в Приложении Д. В таблице 3.1 описана структура проекта. В этой таблице приведены основные логические составляющие, которые используются в работе приложения.

Таблица 3.1 – Описание структурного дерева проекта

|  |  |
| --- | --- |
| Имя файла (папки) | Описание |
| MainWindow.xaml  AdminWindow.xaml  TeacherWindow.xaml  StudentWindow.xaml | Главные окна приложения (View):  - окно авторизации;  - окно администратора;  - окно преподавателя;  - окно студента. |
| DataBaseConnection | Папка, содержащая класс Connection.cs, который реализует подключение к базе данных. |
| RelayCommand | Папка, содержащая класс Command.cs, который служит для создания команд. Реализован на основе паттерна Command. |
| Pictures | Папка, в которую помещены картинки. |
| Login  Admin  Teacher  Student | Папки соответствующих окон, в которых содержатся Model и ViewModel для окон, а так же папку Pages (кроме Login). |
| Pages | Папка, в которой содержатся папки страниц, а в каждой из таких папок находятся View, Model и ViewModel для каждой из страниц окна. |

Так же для описания структуры проекта в Приложении E находится диаграмма классов.

1. **Реализация программного средства**

Начнем с того, что для хранения всей информации в приложении необходима База Данных. Для работы с базой данных была взята СУБД Microsoft SQL Server и была связана с приложением при помощи технологии ADO.NET.

Первым делом для работы с базой данных необходимо к ней подключится. На рисунке 4.1 показан файл Connection.cs, в котором находится строка подключения к базе данных. Сам класс Connection.cs реализован на основе паттерна Singleton и показан в приложении Ж, т.к. в ходе работы приложения мы создаем подключение одно подключение.

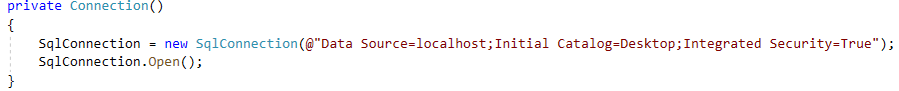


Рисунок 4.1 – Подключение к базе данных

Когда подключение создано, мы можем посылать запросы. Вид одного из запросов показан на рисунке 4.2. На этом рисунке показан select запрос, который выводит нам все номера групп.



Рисунок 4.2 – Select запрос к базе данных

При создании приложения использовался паттерн проектирование MVVM. Он заключается разделении представления от бизнес логики. Это достигается за счёт ввода новой логической конструкции ViewModel. ViewModel связывает представление (View) и бизнес логику (Model) приложения.

Для взаимодействия пользователя и приложения в MVVM используются команды. Это не значит, что вовсе не можем использовать события и событийную модель, однако везде, где возможно, следует использовать команды. Все команды, обычно, находятся во ViewModel. Класс Command.cs, который представлен в Приложении З, дает нам такую возможность. На рисунке 4.3 реализована сама команда, в которой вызывается метод ShowPage().

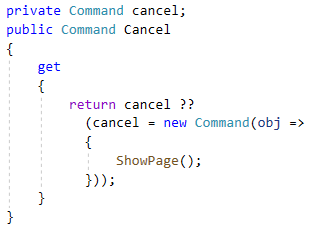


Рисунок 4.3 – Реализация команды

Для привязки команды к UI используется свойство Command. На рисунке 4.4 продемонстрирована привязка команды «Cancel» к кнопке.

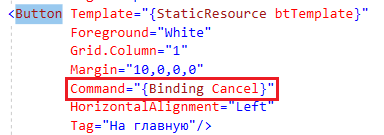


Рисунок 4.4 – Привязка команды к кнопке

Далее в рамках паттерна MVVM каждая часть приложения состоит из трёх частей, как уже говорилось выше – это View, Model и ViewModel.

Вся работа приложения сводится к привязке данных к нашей модели.

Для того, чтобы была возможность привязывать данные, необходимо к классе, который отвечает за «логику» окна, установить контекст, т.е. куда надо ссылаться для получения информации. В данном проекте контекст устанавливается на соответствующую ViewModel окна (или страницы). На рисунке 4.5 это видно.

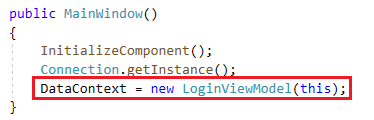


Рисунок 4.5 – Установка контекста для окна

Когда контекст установлен, мы спокойно можем привязывать данные через атрибут «Binding». Рисунок 4.6 показывает нам этот процесс.

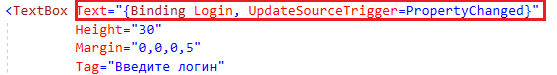


Рисунок 4.6 – Привязка данных

Как видно из рисунка 4.6, установлено свойство «UpdateSourceTrigger», которое принимает одно из значений, описанных ниже:

*- PropertyChanged* – источник обновляется немедленно, когда изменяется целевое свойство;

*- LostFocus* – источник обновляется немедленно, когда изменяется целевое свойство и цель теряет фокус;

*- Explicit* – источник не обновляется, пока не будет вызван метод BindingExpression.UpdateSource();

*- Default* – поведение обновления определяется метаданными целевого свойства.

Для того, чтобы отследить изменения этих свойств, необходимо использовать интерфейс INotifyPropertyChanged. Этот интерфейс содержит метод OnPropertyChanged, который необходимо реализовать наследуемому классу, что и показано на рисунке 4.7.

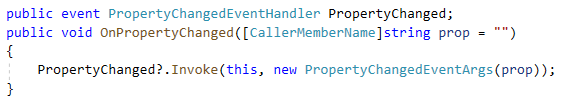


Рисунок 4.7 – Реализация метода OnPropertyChanged

Когда метод реализован, его необходимо поместить в отслеживаемое свойство. Рисунок 4.8 демонстрирует отслеживание изменения свойства.

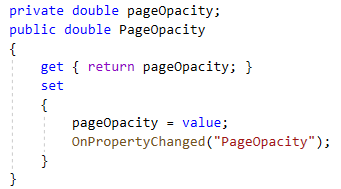


Рисунок 4.8 – Отслеживание изменения свойства

Для дизайна в проекте использованы стили, триггеры (рисунок 4.9) и шаблоны (рисунок 4.10).



Рисунок 4.9 – Стиль, содержащий триггеры



Рисунок 4.10 - Шаблон

На рисунках 4.11 и 4.12 показано их подключение соответственно, а на рисунке 4.13 – готовый результат.

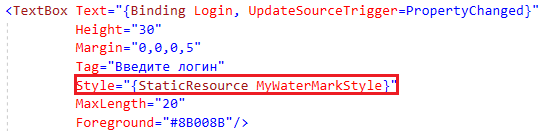


Рисунок 4.11 – Применение стиля к TextBox

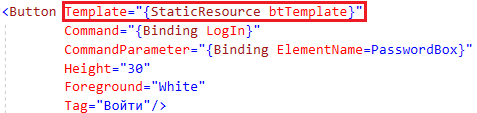


Рисунок 4.12 – Применение шаблона к Button

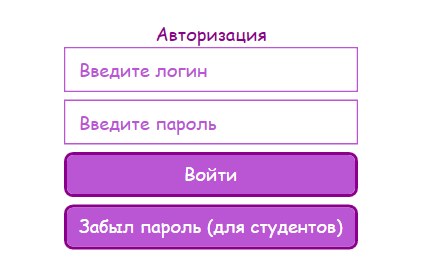


Рисунок 4.13 – Результат применения стилей и шаблонов

Для наглядности данных процессов в Приложении И показана реализация авторизации пользователей.

1. **Тестирование, проверка работоспособности и анализ полученных результатов**

Для корректной работы программы необходимо обеспечить защиту работы пользователя от ошибок и сбоев. Для этого лучше использовать конструкцию «if…else» для обработки ситуации неверного ввода информации пользователем.

В любой программе есть допустимые значения на ввод информации. Кроме того, пользователь может допустить ошибку при выборе поля и значений для него. Для всех этих ошибок нужно написать исключения для дальнейшей возможности работы приложения.

Далее будут представлено с тестирование приложения, показанные на рисунках 5.1-5.8:

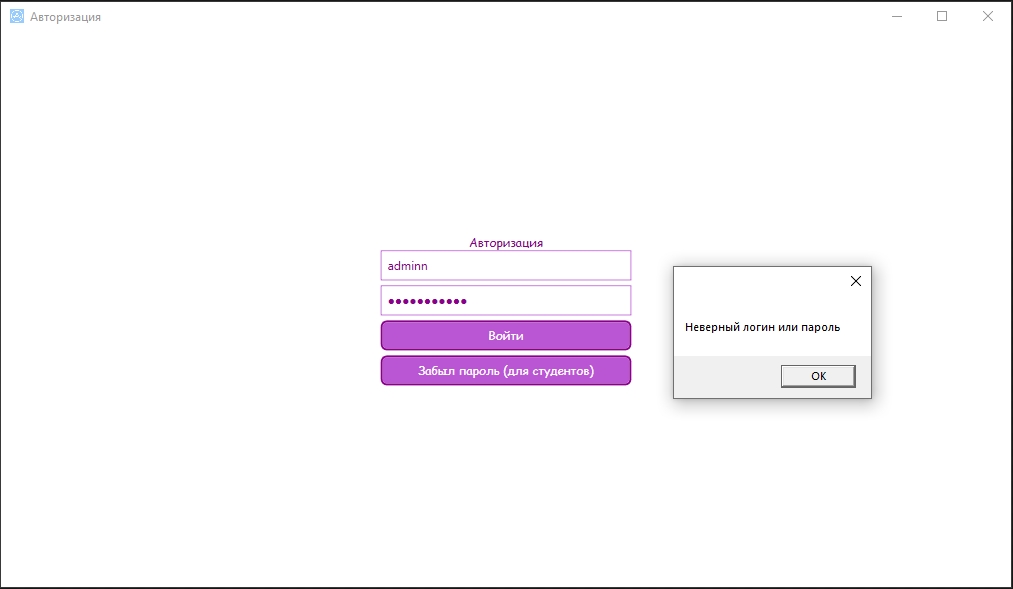


Рисунок 5.1 – Ввод неверных данных

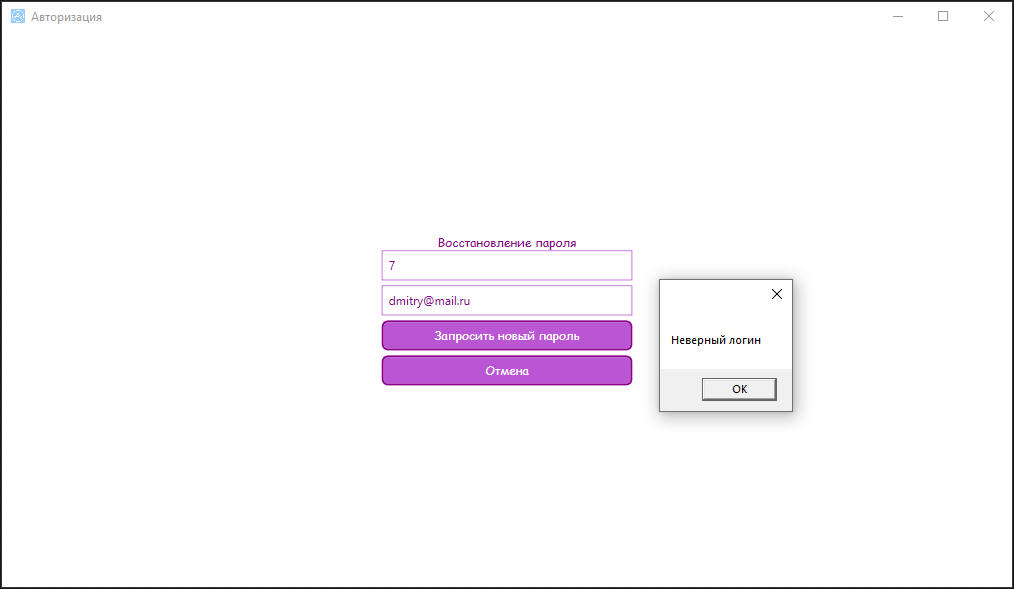


Рисунок 5.2 – Ввод несуществующего логина при восстановлении пароля

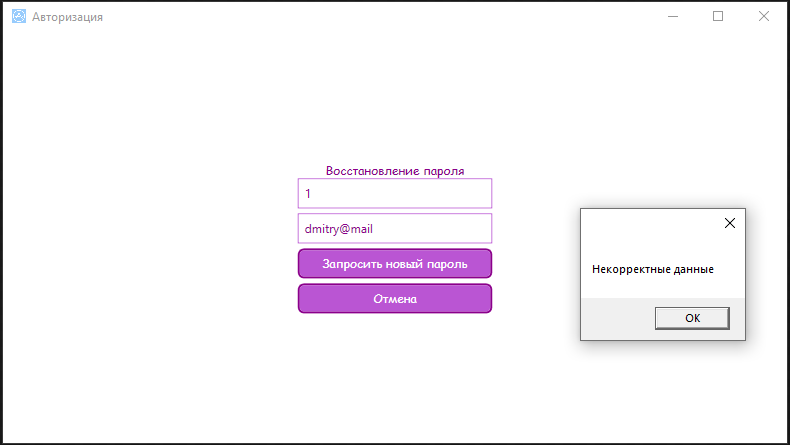


Рисунок 5.3 – Ввод некорректной почты при восстановлении пароля

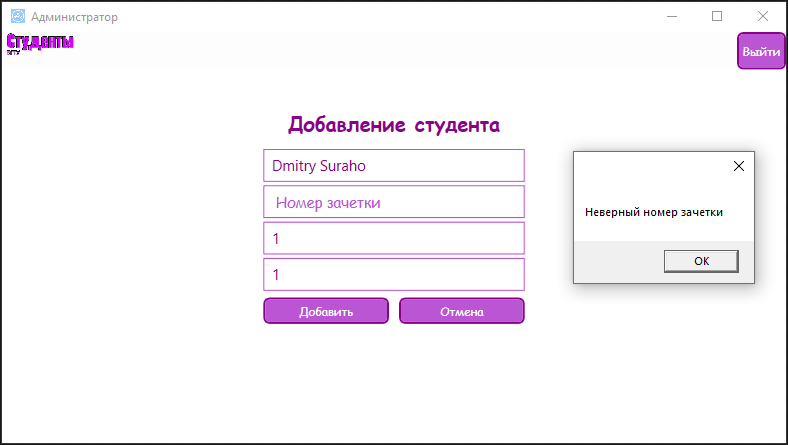


Рисунок 5.4 – Попытка оставить пустым поле для ввода

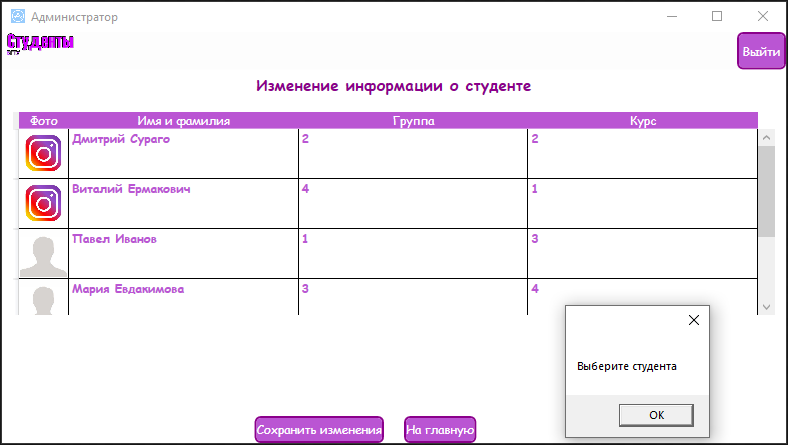


Рисунок 5.5 – Попытка сохранения изменений без выбора студента

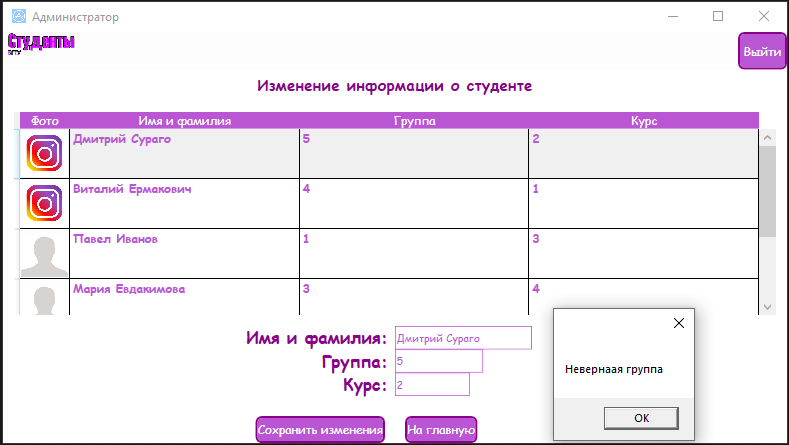


Рисунок 5.6 – Ввод несуществующей группы

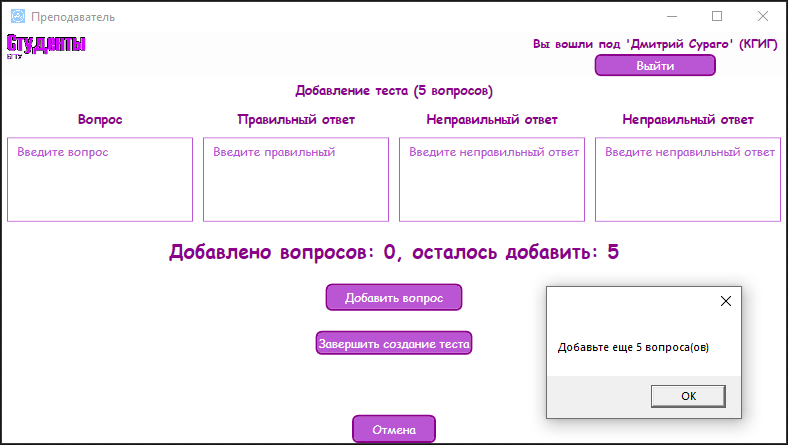


Рисунок 5.7 – Попытка завершить тест при не всех добавленных вопросах

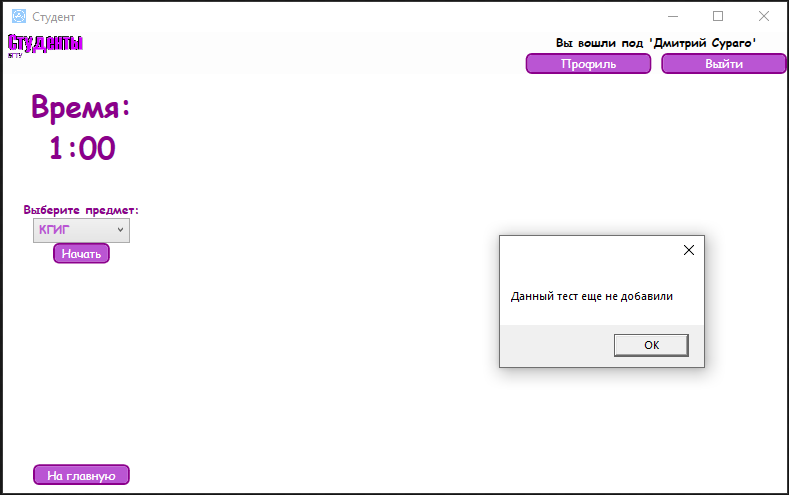


Рисунок 5.8 – Попытка начать тест, которого нет в базе данных

Были затронуты почти все возможные ошибки пользователя, для лучшего теста программы следует использовать динамическое тестирование интерфейса, которые проводится во время выполнения программы.

1. **Руководство пользователя по использованию**

Правила заполнения полей при регистрации пользователей (для администратора):

- логин должен быть уникальным;

- данные должны быть корректными.

Правила для авторизации:

- логин должен быть верным;

- пароль должен быть верным.

Правила заполнения полей при восстановлении пароля:

- логин должен быть верным;

- проверка пользователя осуществляется с помощью кода, высланного на почту;

- почта должна быть корректной.

Правила добавления данных в базу данных:

- необходимо учитывать связь между таблицами;

- некоторые таблицы требуют ввода существующих данных;

- данные должны быть корректными.

На следующих рисунках показаны интерфейсы для каждой из ролей.

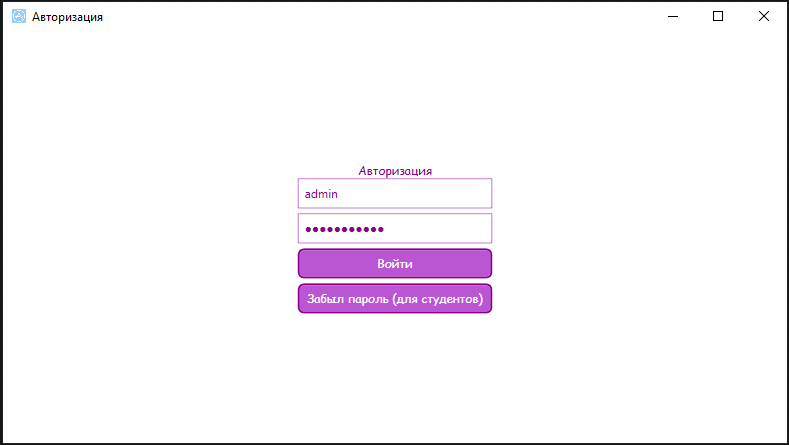


Рисунок 6.1 – Окно авторизации

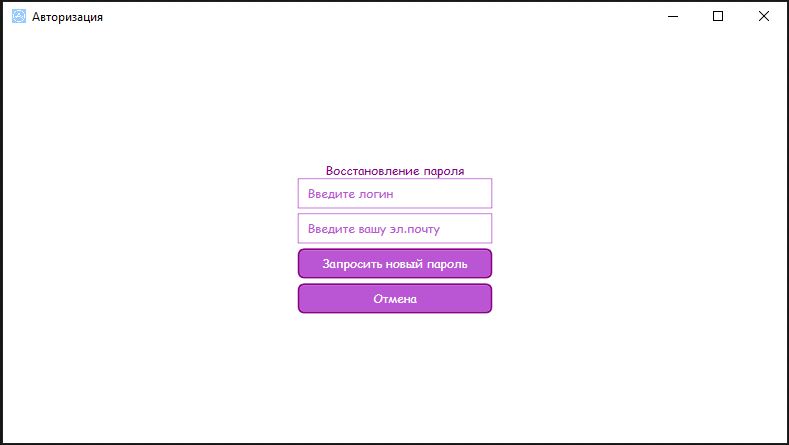
****

Рисунок 6.2 – Окно восстановления пароля

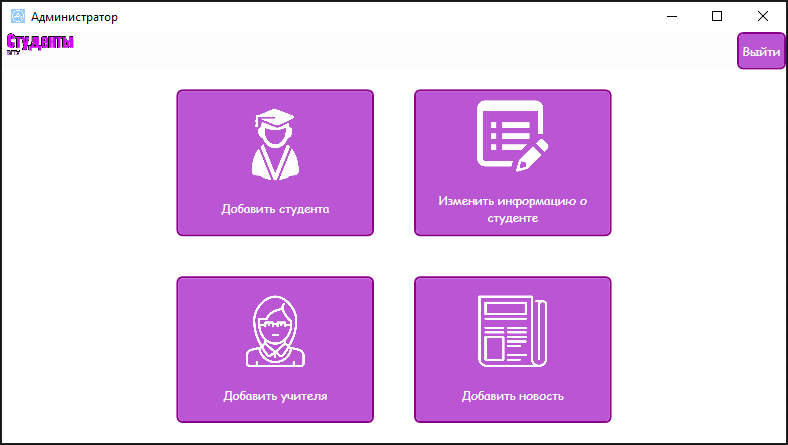
****

Рисунок 6.3 – Интерфейс администратора

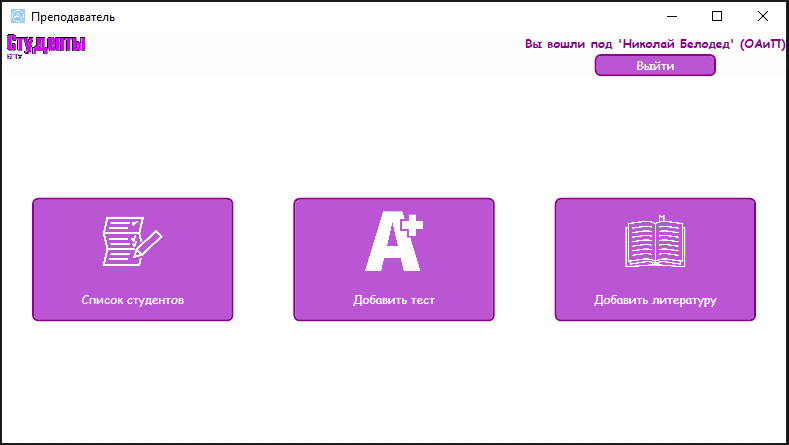
****

Рисунок 6.4 – Интерфейс преподавателя

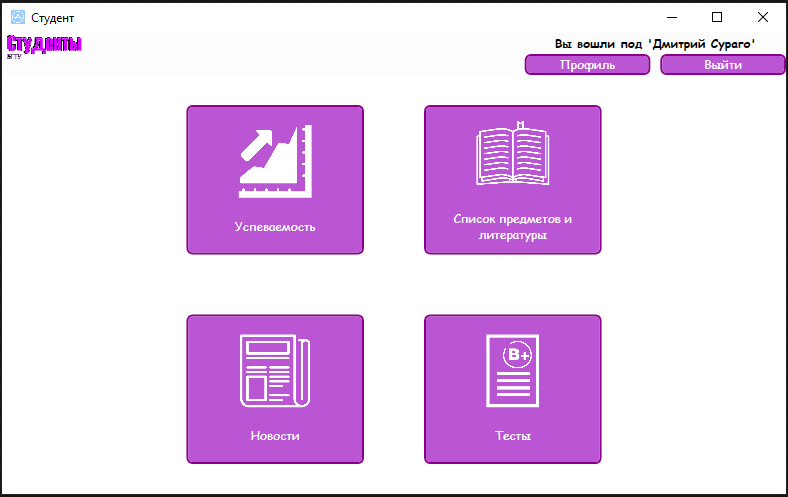
****

Рисунок 6.5 – Интерфейс студента

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения работы были достигнуты следующие результаты:

- изучены основы объектно-ориентированного программирования и применены в практической разработке программного средства;

- исследована сфера управления пользователями приложения;

- сформулирована теория, что внедрение паттерна MVVM способствует простому внедрению дополнительного функционала, простому расширению проекта, также повышает эффективность работы нескольких человек над проектом, возможность разрабатывать параллельно различные части проекта, и упрощает тестирование программного средства;

- показано и испытанно взаимодействие между уровнями проектирования во время выполнения программы;

- подготовлены материалы для дальнейшей разработки приложения и изучения данной сферы.

**СПИСОК ЛИТЕРАТУРЫ**

[1] Wikipedia [Свободная энциклопедия]. – Электронные данные. – Режим доступа: <https://en.wikipedia.org/wiki/Main_Page>.

[2] draw.io [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://www.draw.io/>.

[3] Metanit [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://metanit.com/>.

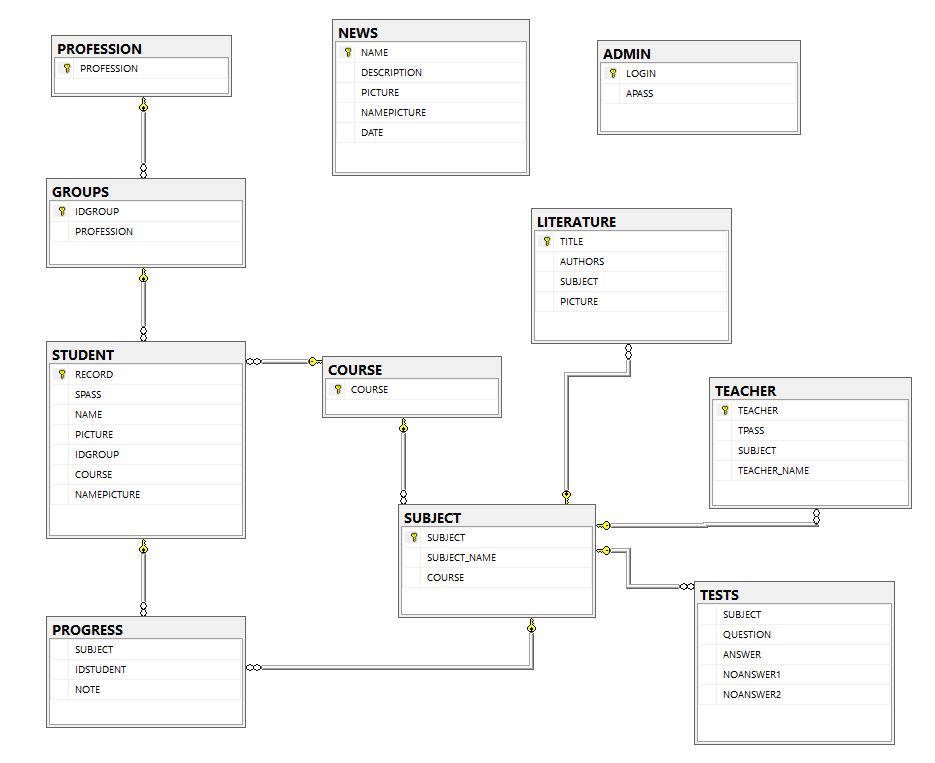
[4] Habr [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://habr.com/ru/>.

[5] StackOverflow [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://stackoverflow.com/>.

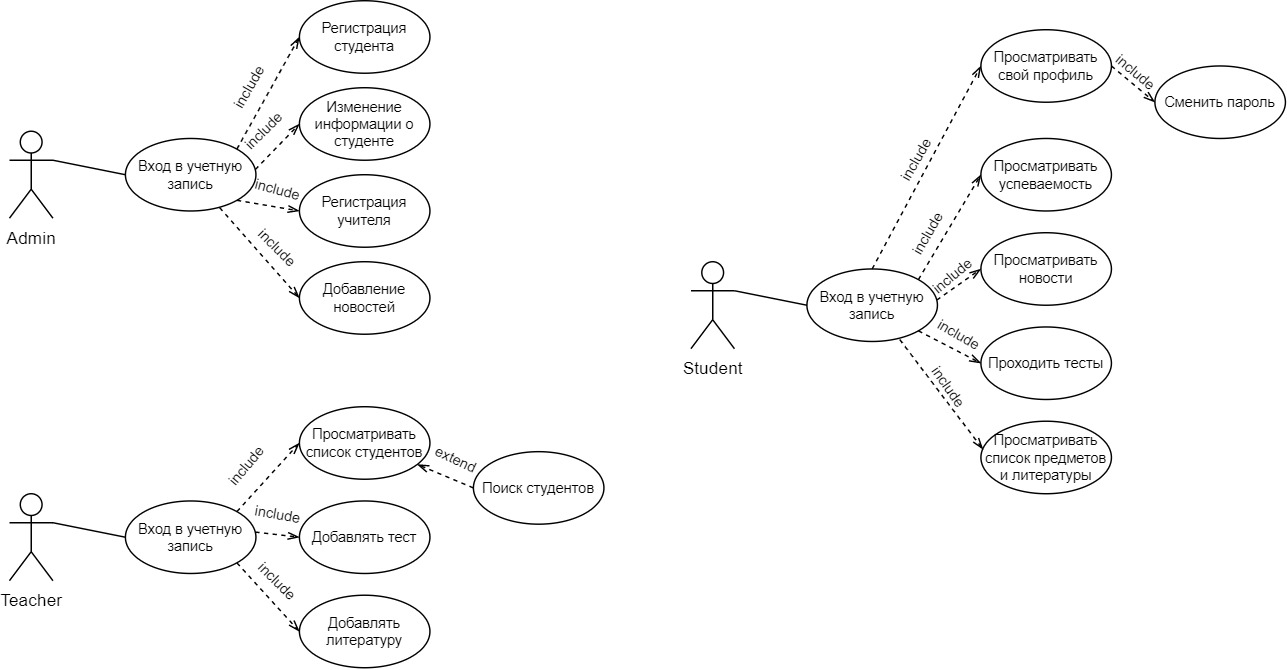
[6] Пацей, Н.В. Курс лекций по языку программирования C# / Н.В. Пацей. Минск: БГТУ, 2016. – 175 с.

[7] Microsoft Docs [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://docs.microsoft.com/en-us/>.

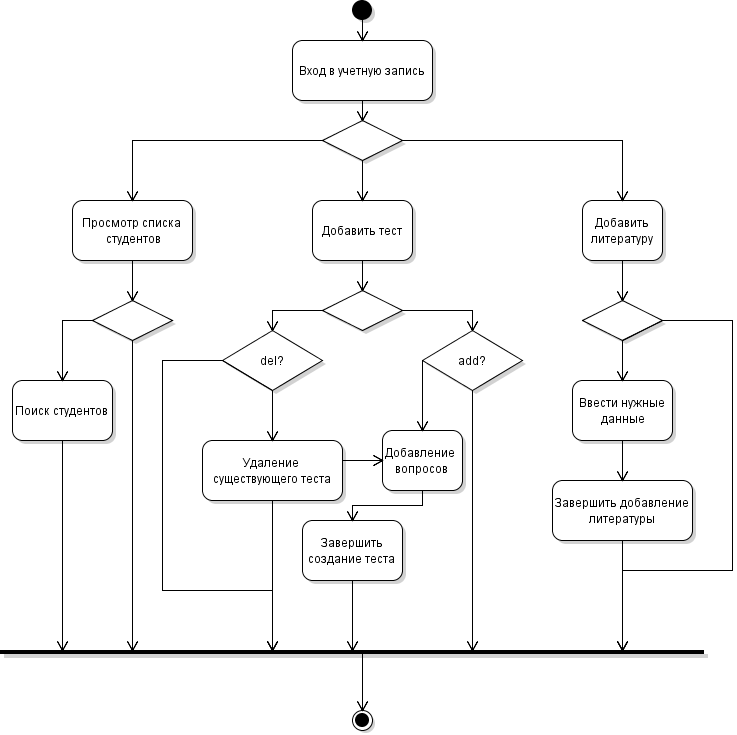
**ПРИЛОЖЕНИЕ А**



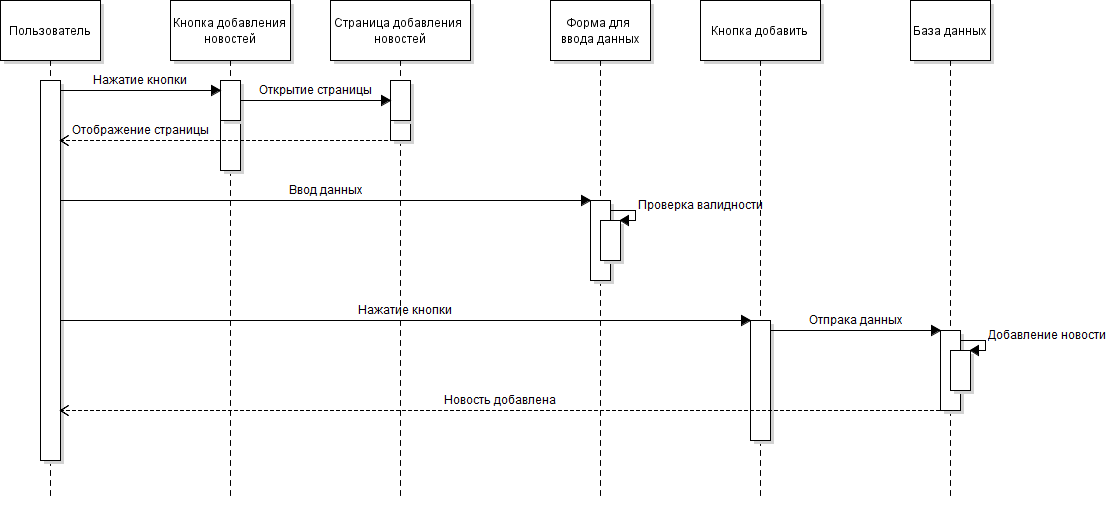
**ПРИЛОЖЕНИЕ Б**



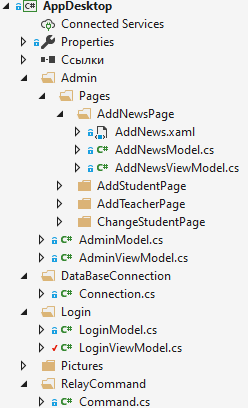
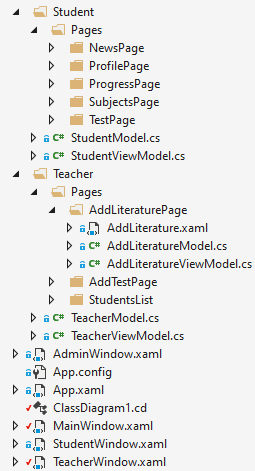
**ПРИЛОЖЕНИЕ В**



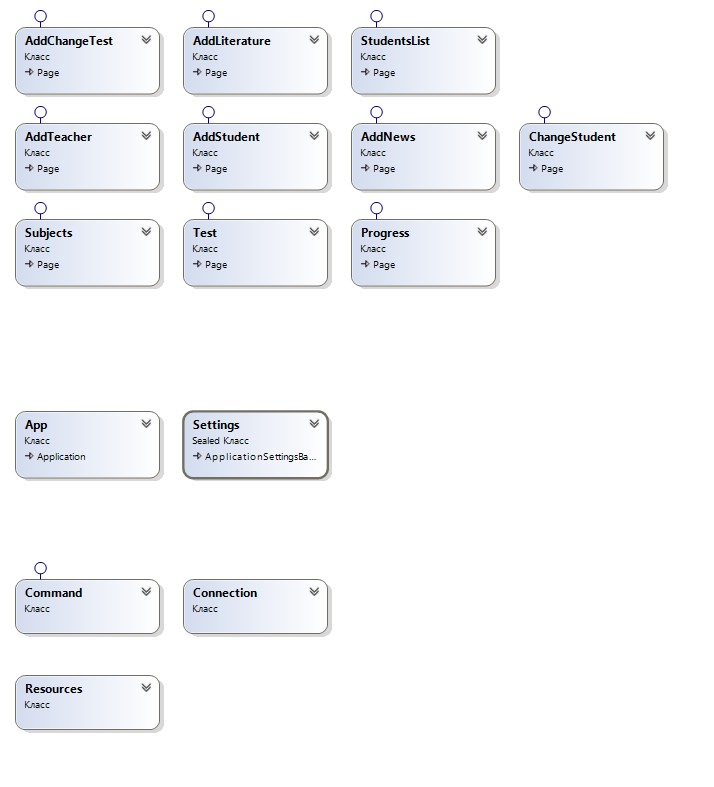
**ПРИЛОЖЕНИЕ Г**

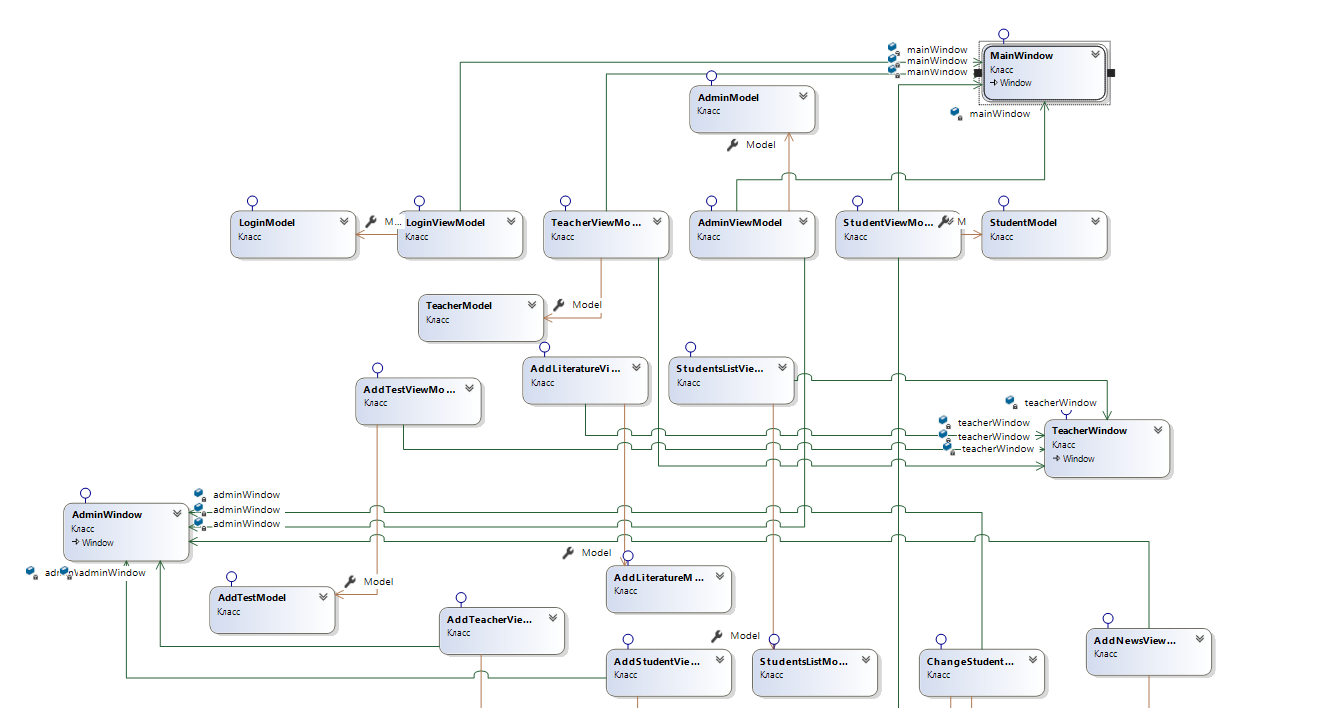
****

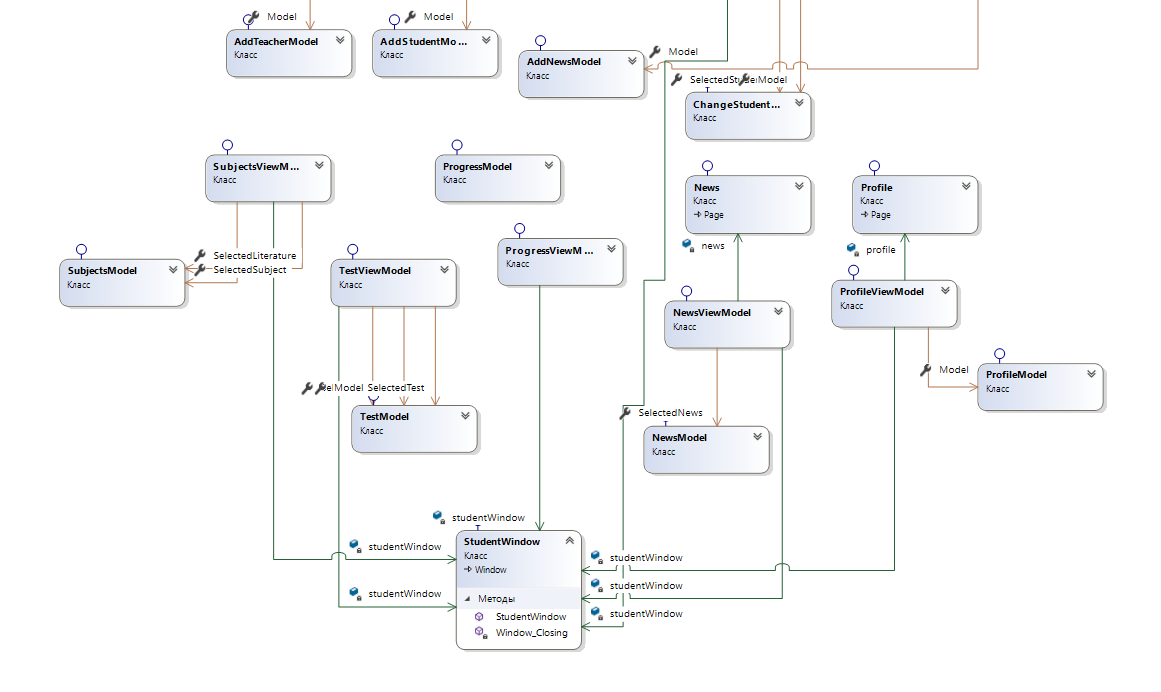
**ПРИЛОЖЕНИЕ Д**

** **

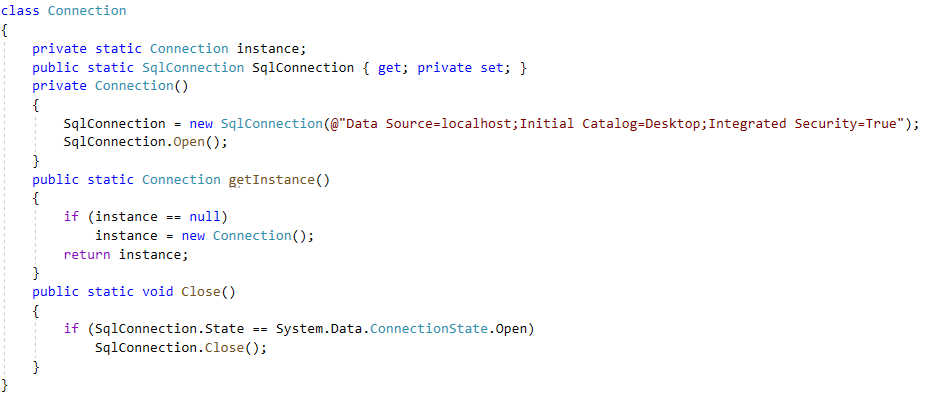
**ПРИЛОЖЕНИЕ E**

****

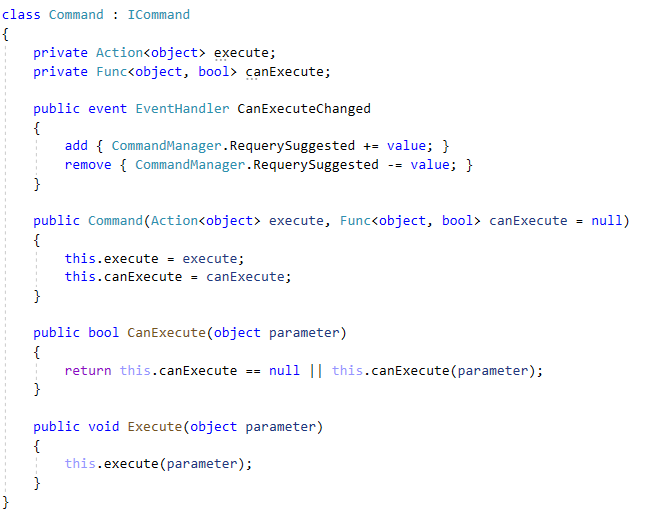
****

****

**ПРИЛОЖЕНИЕ Ж**

****

**ПРИЛОЖЕНИЕ З**



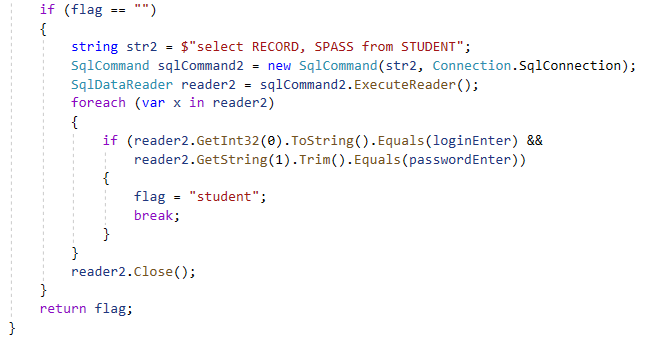
**ПРИЛОЖЕНИЕ И**

MainWindow.xaml (текстовые поля для ввода логина и пароля, а так же кнопка «Войти»)



LoginModel.cs (метод Check, который проверяет правильность введенных данных)





Login.ViewModel.cs (команда LogIn, которая проверяет значение функции Check, и в зависимость от того, что вернула функция, решает, пропускать пользователя дальше или нет)

