Донской политехнический колледж

**Практическая работа**

по дисциплине «Проектирование пользовательских интерфейсов»

на тему:

**«Курсы повышение квалификации - «Курсывест»**

Выполнил:

Студен гр. П-20-1

Горин Александр Дмитриевич

Преподаватель:

Гвоздев Сергей Михайлович

г. Донской

20221

# Определение необходим страниц в программе

Для использования программы нам необходимы следующие страницы

1. Авторизация
2. Регистрация
3. Настройки профиля
4. Главная страница
5. Страница курса
6. Старица профиля с каталогом «Изучаю», «Буду изучать», «Изучил»

# Определение необходимы таблиц БД

User:

|  |  |  |  |
| --- | --- | --- | --- |
| Имя столбца | Тип данных | Длинна | Описание |
| UserID | int |  | Уникальный ID записи |
| FirstName | nchar | 50 | Имя пользователя |
| LastName | nchar | 50 | Фамилия пользователя |
| Email | char | 355 | Почта |
| Password | char | 30 | Пароль |
| RoleID | int | 11 | Роль в системе |
| Avatar | text |  | Картинка пользователя |

Таблица 2.1 – Таблица пользователей

Courses:

|  |  |  |  |
| --- | --- | --- | --- |
| Имя столбца | Тип данных | Длинна | Описание |
| CoursesID | int |  | Уникальный ID курса |
| Title | char | 30 | Название курса |
| Description | text |  | Описание курса |
| Preview | text |  | Изображение курса |
| PublisherID | int |  | ID пользователя, опубликовавшего курс |
| Data | text |  | Содержимое курса |

Таблица 2.2 – Таблица курсов

Subscribe:

|  |  |  |  |
| --- | --- | --- | --- |
| Имя столбца | Тип данных | Длинна | Описание |
| SubscribeID | int |  | Уникальный ID подписки |
| CoursesID | int |  | ID курса |
| UserID | int |  | ID пользователя |
| Library | int |  | Тип подписки (изучаю, изучил, бросил) |

Таблица 2.3 – Таблица каталога

Role

|  |  |  |
| --- | --- | --- |
| Имя столбца | Тип данных | Описание |
| RoleID | int | Уникальный ID роли |
| Name | nchar(20) | Название роли |

Таблица 2.4 – Таблица каталога

Описание столбца Library: библиотека пользователя – то есть,

значение 1 – изучаю,

значение 2 – изучил,

значение 3 – буду изучать

# Создание базы данных

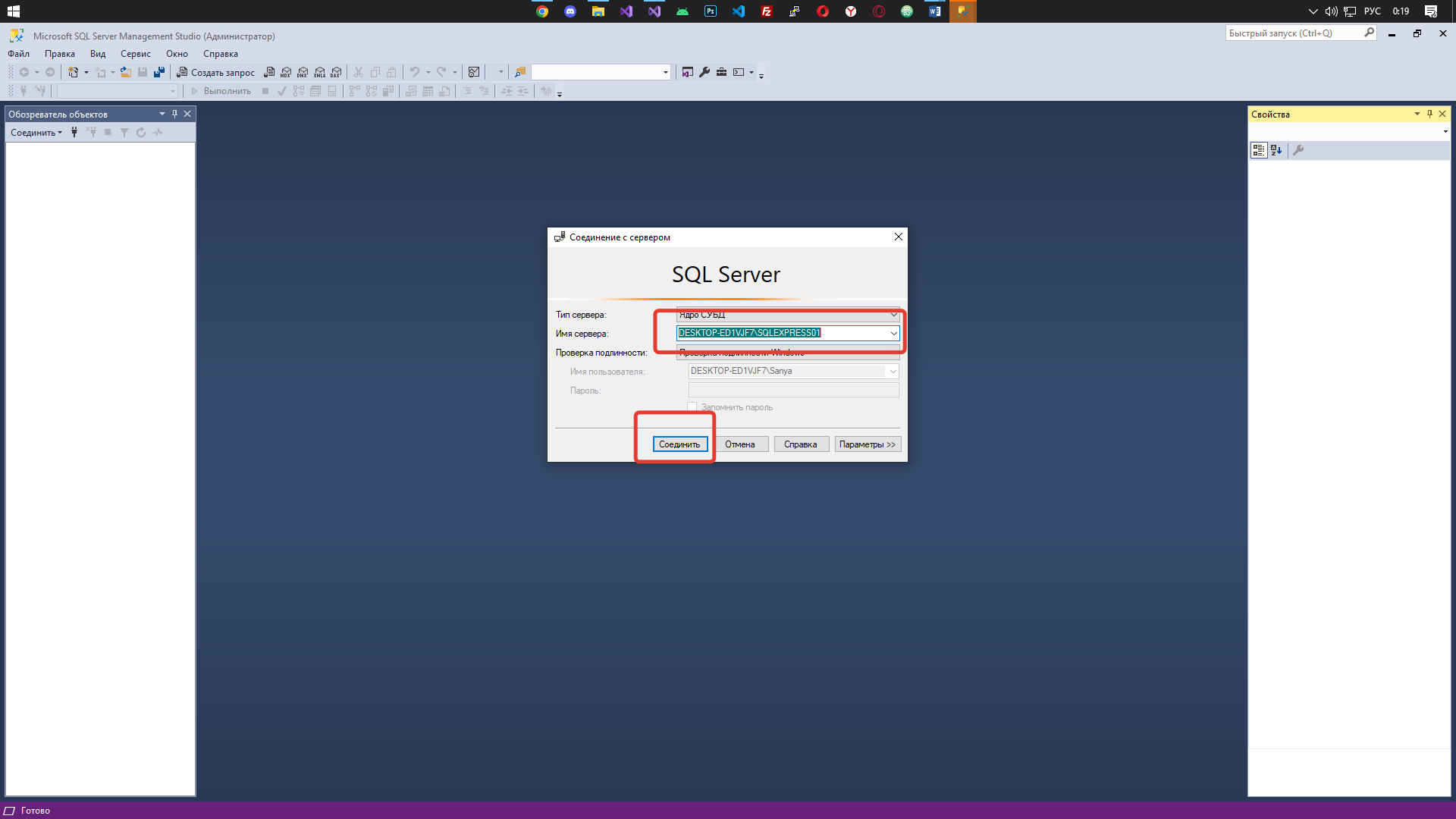
Открываем SQL Server Manager Studio, подключаемся к базе данных. 

Рисунок 3.1 – Подключение к базе

Разворачиваем список «Базы данных» и нажимаем по нему правой кнопкой мыши

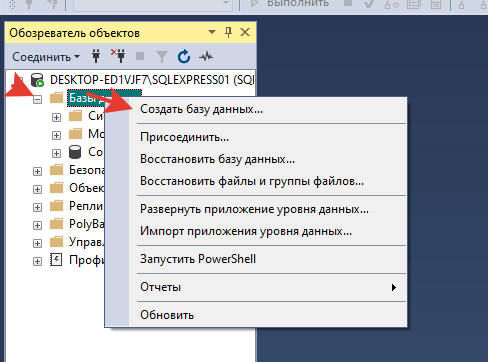


Рисунок 3.2 – Создание новой БД

Разворачиваем только что созданную базу данных. Далее жмем правой кнопкой мыши по «Диаграммы баз данных» и выбираем пункт «Создать диаграмму базы данных»

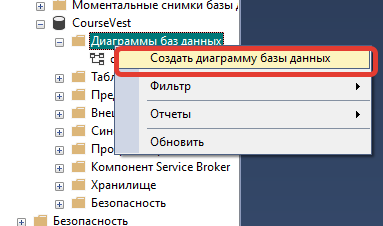


Рисунок 3.3 – Создание диаграммы БД

Появится новое окно. После этого в пустом месте нажимаем правой кнопкой мыши и выбираем пункт «Создать таблицу»

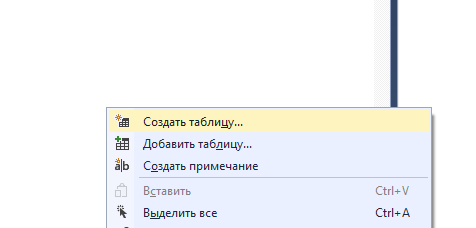


Рисунок 3.4 – Создание таблицы

Далее создаем необходимые столбы в таблицах из таблиц 2.1, 2.2, 2.3, 2.4

В конечном итоге получаем следующую диаграмму базы данных

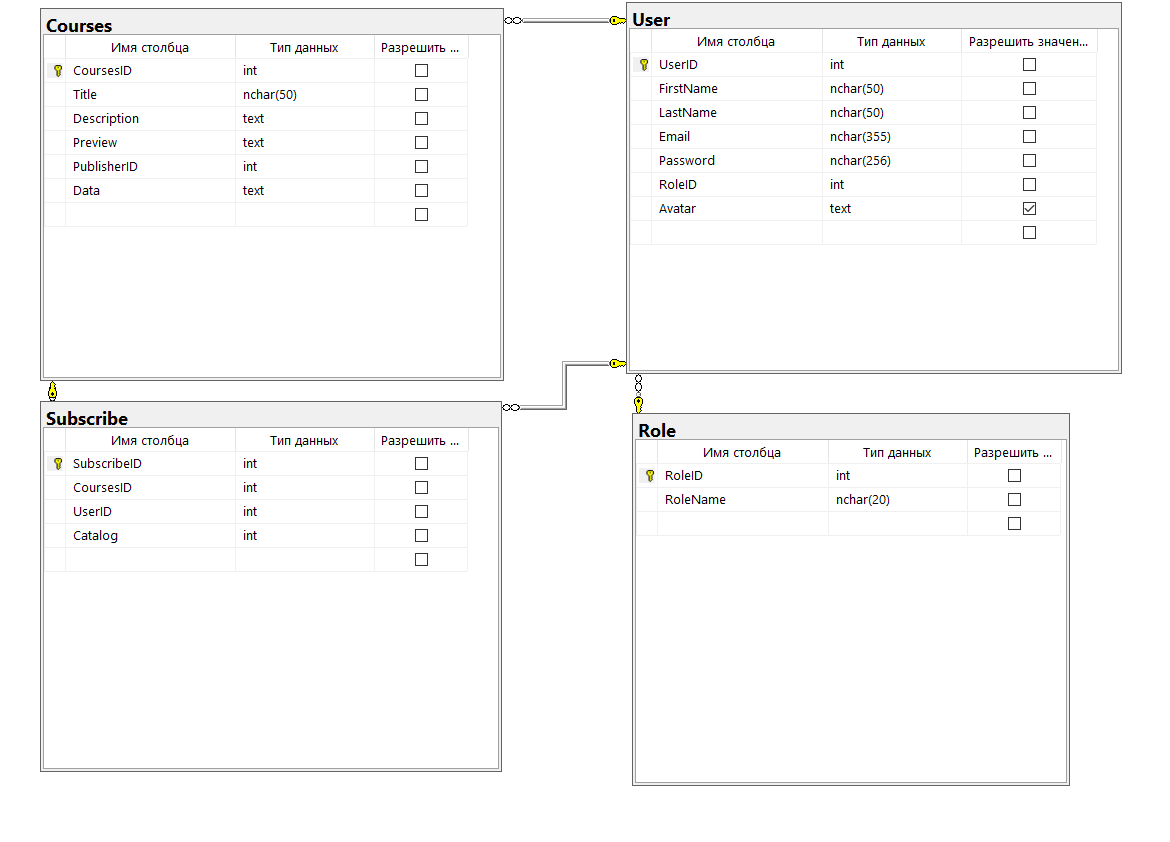


Рисунок 3.5 – Диаграмма базы данных

# Создание проекта

Создадим новый проект – выбираем «Приложение WPF (.Net Framework)» и нажимаем «Далее».

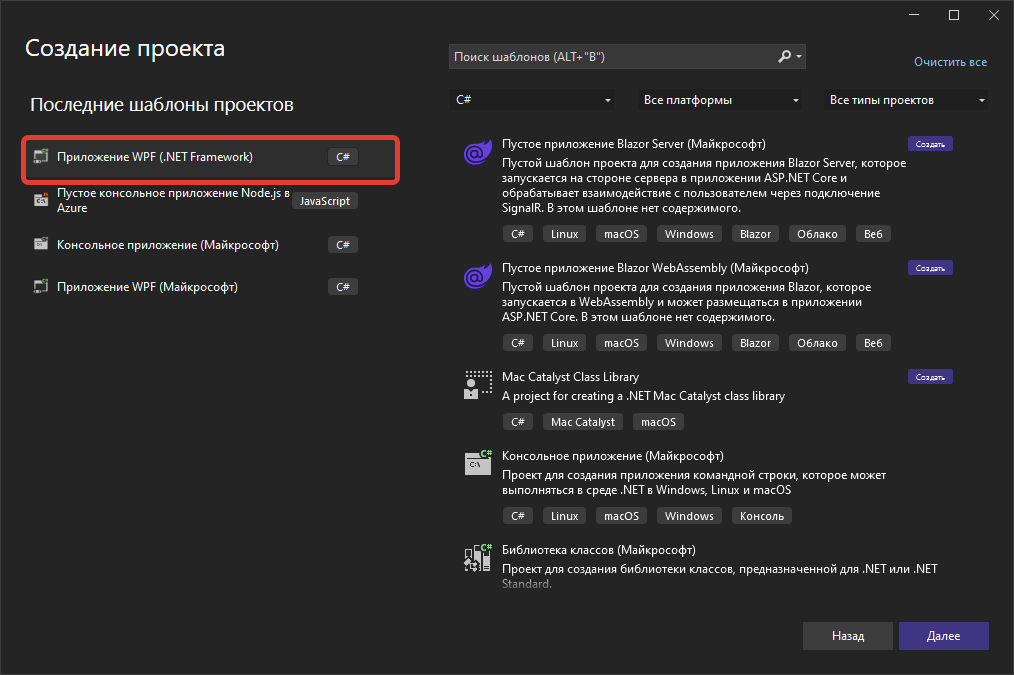


Рисунок 4.1 – Создание проекта

В окне «Настроить новый проект» заполняем поля:

* Имя проекта;
* Расположение;
* Выбираем платформу.

После заполнения всех полей нажимаем на кнопку «Создать»

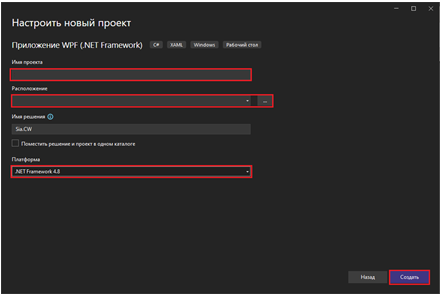


Рисунок 4.2 – Настройка нового проекта

Создадим общую структуру проекта. Она будет состоять из следующих директорий:

* Core – директория которая содержит вспомогательные классы;
* Images – директория которая содержит графические ресурсы приложения;
* Model – директория будет содержать классы данных.
* View – директория которая содержит: окна, страницы и пользовательские элементы используемые в приложении.

Теперь необходимо установить графическую библиотеку «Material Design». Для этого ПКМ нажимаем по проекту в появившемся окне выбираем «Управление пакетами NuGet…», выбираем вкладку «Обзор», в поиске вводим «Material Design» и устанавливаем два компонента «MaterialDesignThemes» и «MaterialDesignColors».

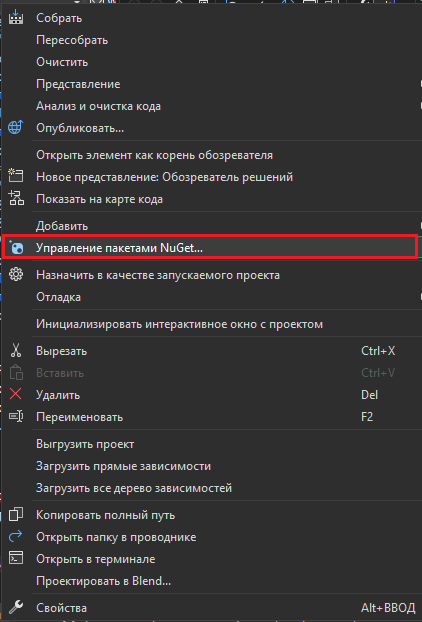


Рисунок 4.3 – Контекстное меню «Управление пакетами NuGet…»

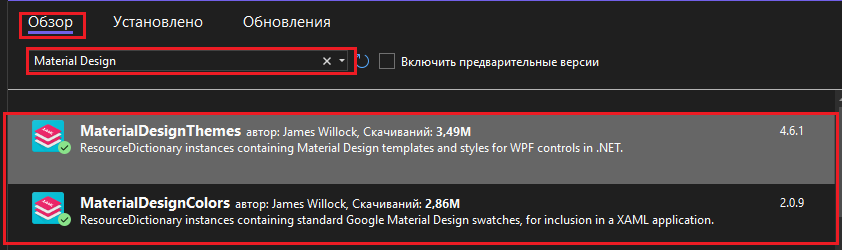


Рисунок 4.4 – Установка «MaterialDesignThemes» и «MaterialDesignColors».

Теперь перенесем «MainWindow» в директорию «View» и изменим файл «App.xaml». Изменим путь к «MainWindow» в «StartupUri» и добавим справочник ресурсов, а также добавим новое пространство имен «materialDesign».

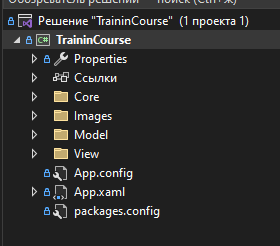


Рисунок 4.5 – Выбор файла «App.xaml»

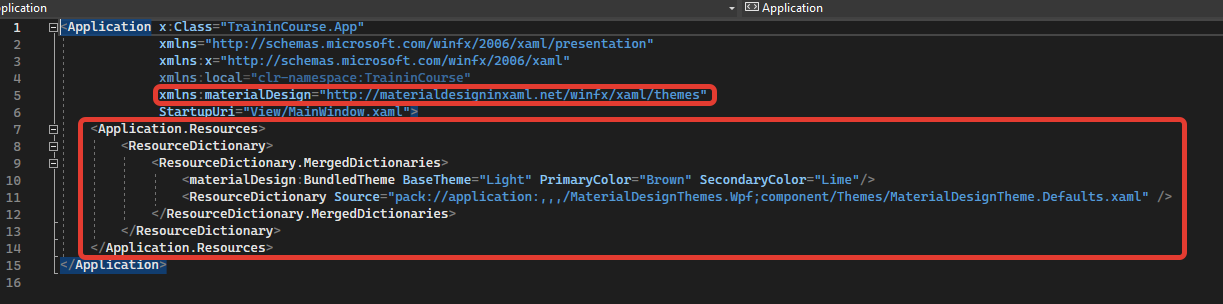


Рисунок 4.6 – Редактирование файла «App.xaml»

Далее переходим в директорию «Model». В данную директорию импортируем, используя «EntityFramework» объектно-ориентированную модель базы данных (более детально данная тема разбирается в «Практической работе №1 – Работа с EntityFramework»). После импорта переходим к разметке «MainWindow»

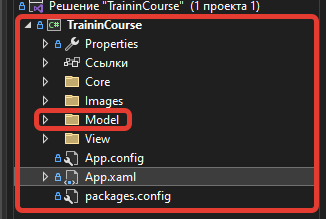


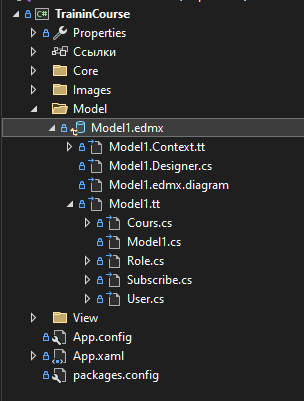
Рисунок 4.7 – Выбор директории файла «Model» 

Рисунок 4.8 – Импорт данных

В «MainWindow» добавим ссылку на «materialDesign», изменим параметры высоты и ширины, отключим стандартные стили окна, добавим событие «MouseDown» (данное событие будет отвечать за возможность перемещения пользователем окна) и последним параметром мы запретим пользователю менять размер окна (Разметка «MainWindow» предоставлена на рисунках 1.10 – 1.).

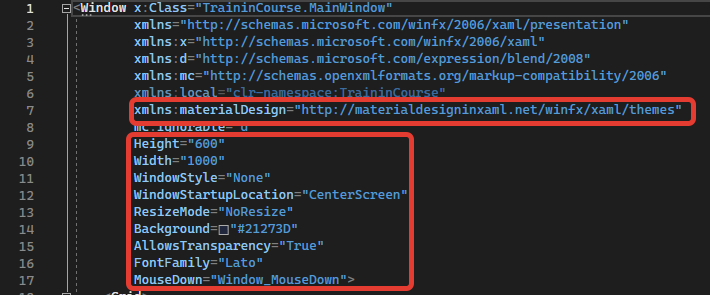


Рисунок 4.9 – Разметка «MainWindow»

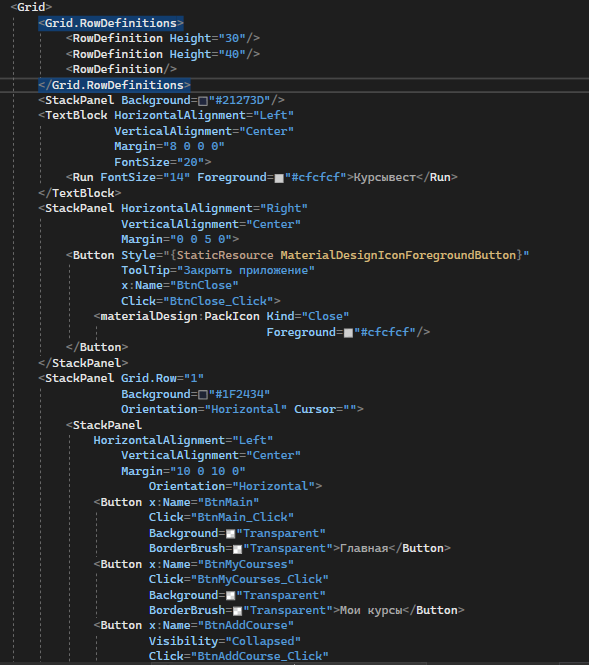


Рисунок 4.10 – Разметка «MainWindow»

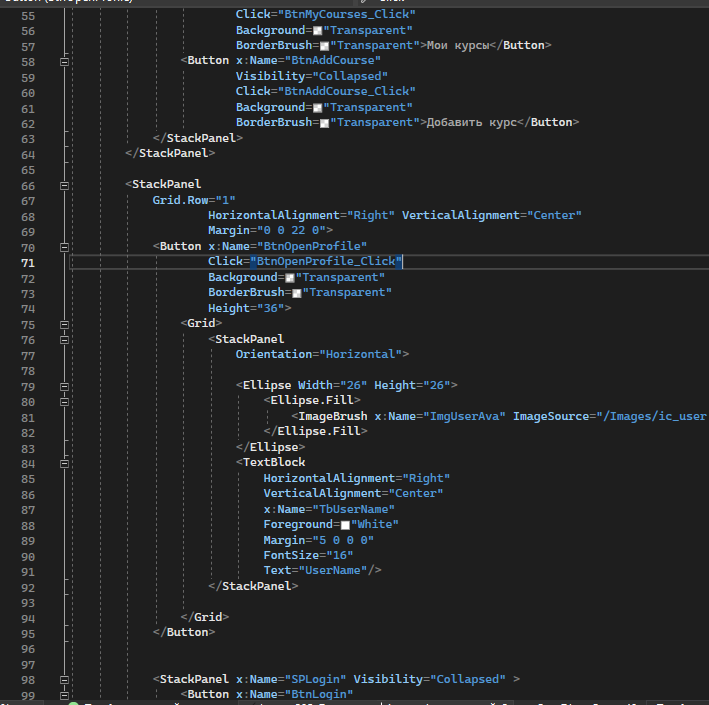


Рисунок 4.11 – Разметка «MainWindow»

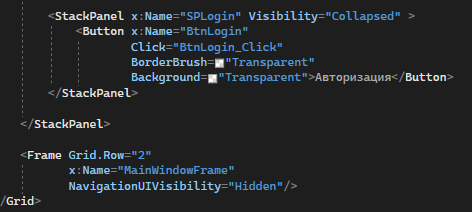


Рисунок 4.12 – Разметка «MainWindow»

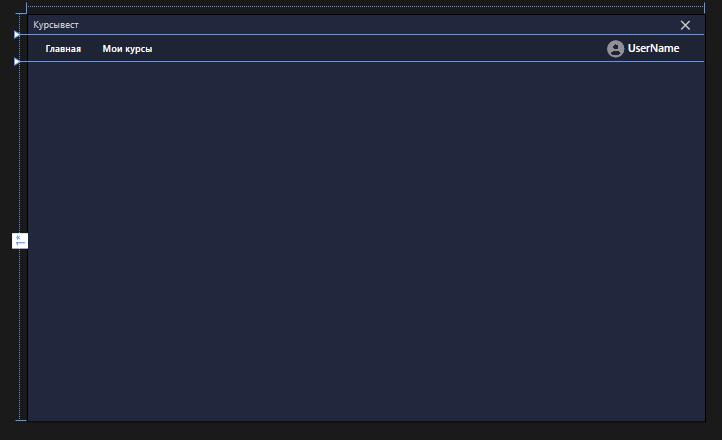


Рисунок 4.13 – Итоги разметки «MainWindow»

Перейдем в директорию «Core». В этой директории создадим статический класс «MainUtil» с тремя статическими свойствами, одной статичной переменной и один методом «Bool». «FrameObject» (будет иметь тип данных «Frame»), «DB» (будет иметь тип данных «CourseVestEntities») и «MyPerson» (будет иметь тип данных объектно-ориентированной модели данных «User»). Переменной «isAuth» с типом данных «bool» - для обозначение, авторизирован ли пользоваетль или нет и методом «auth» c типом данных «bool», которое вернет значение, успешно ли пройдена авторизации». Поле «FrameObject» будет использоваться для упрощенной навигации в приложении. Поле «DB» предоставить доступ объектно-ориентированной модели данных.

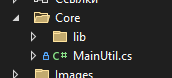


Рисунок 4.14 – Дериктория «Core»



Рисунок 4.15 – Реализация класса «MainUtil»

Далее в папке «Core» создадим папку «lib», которая будет содержать два класса и папку с вспомогательным классом для работы с локальной базы данных SQLite. Создадим новый класс «ImageLoader.cs», который поможет нам подставлять изображение из интернета в элемент «Image».

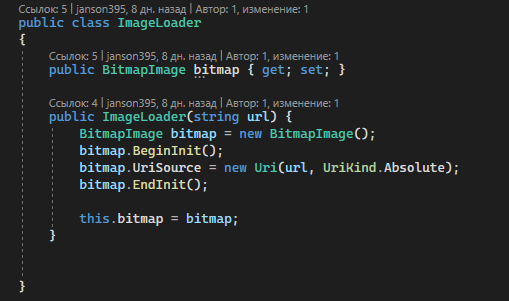


Рисунок 4.16 – Класс «ImageLoader.cs»

Затем создадим класс «CourseList.cs». Он будет являться объектом для списка курсов.

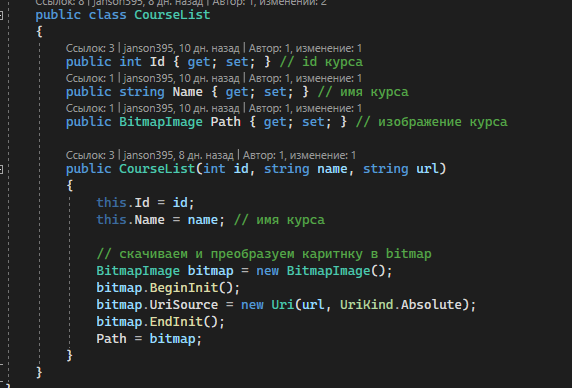


Рисунок 4.17 – Класс «CourseList.cs»

Создаем вспомогательный класс работы с БД «AuthDB.cs» в папке DB.

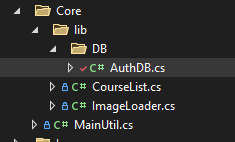


Рисунок 4.18 – Общий вид папки «Core»

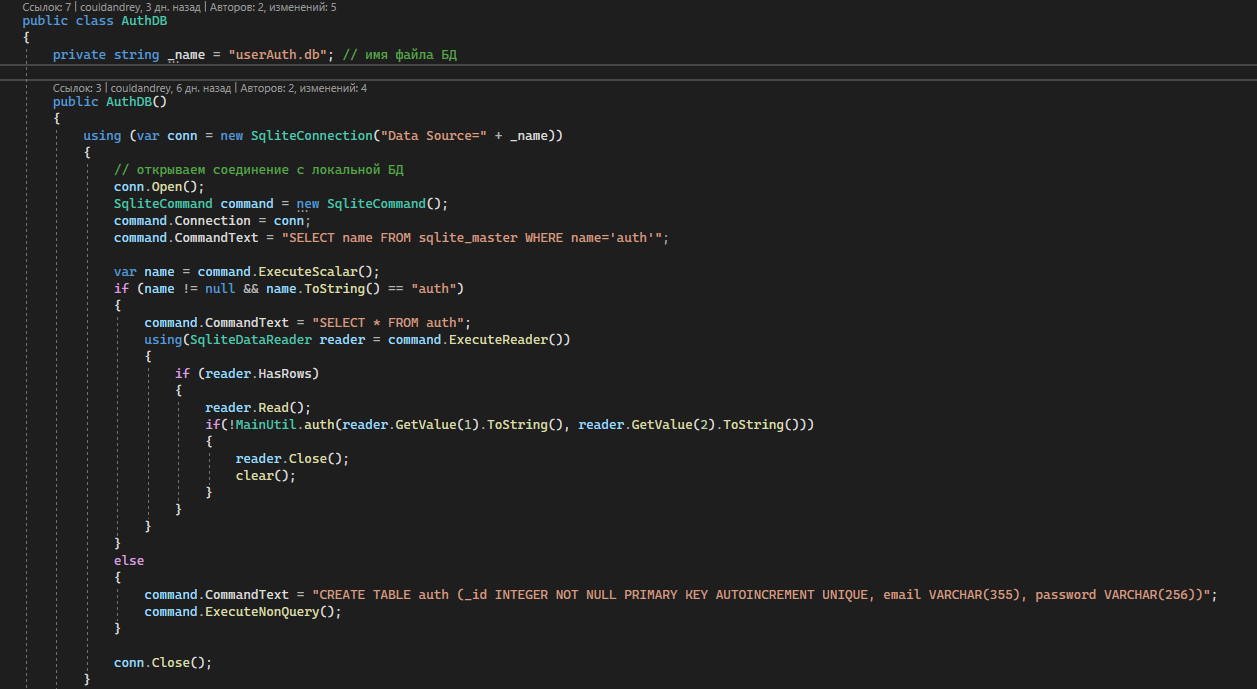


Рисунок 4.19 – Реализация класса «AuthDB.cs»



Рисунок 4.20 – Реализация класса «AuthDB.cs»

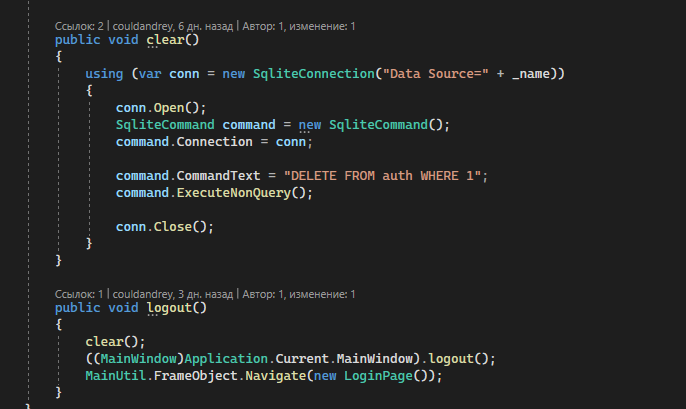


Рисунок 4.21 – Реализация класса «AuthDB.cs»

Далее переходим в директорию «View». Создадим три директории:

- CoursesPage – будет содержать страницу курса и папку администрирования

- LoginPages – будет содержать страницы для входа и регистрации

- Profile – будет содержать страницы профиля и настроек профиля

В директории «LoginPages» создаем две страницы «LoginPage» и «RegPage». Далее переходим в файл «MainWindow.xaml.cs». В этом файле свяжем поле (из класса «FrameNavigate») «FrameObject» с «MainWindowFrame», а в поле «DB» создадим экземпляр «CourseVestEntities». Также реализуем два события: «Window\_MouseDown» – данное событие разрешить пользователю перемещать «окно» приложения; «BtnClose\_Click» – данное событие будет завершать работу приложения.

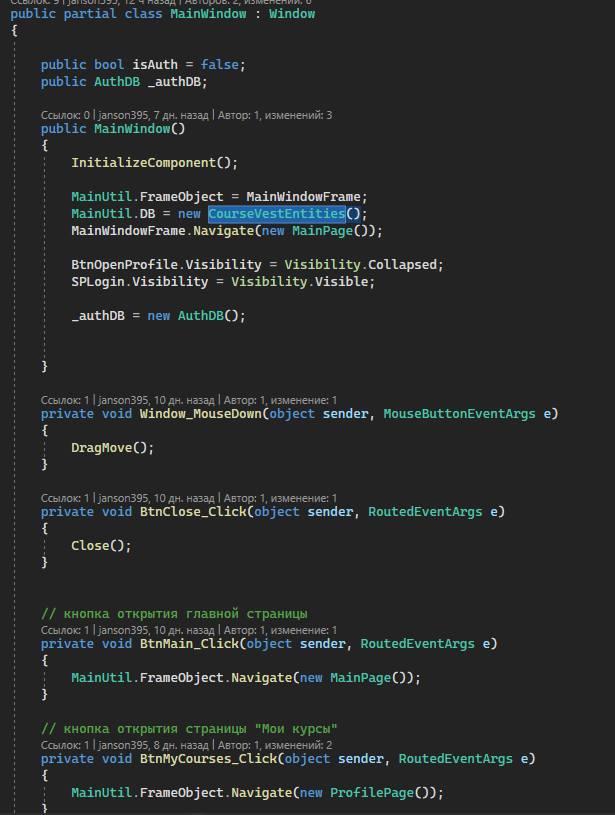


Рисунок 4.22 – Реализация логики в файле «MainWindow.xaml.cs»



Рисунок 4.31 – Реализация логики в файле «MainWindow.xaml.cs»

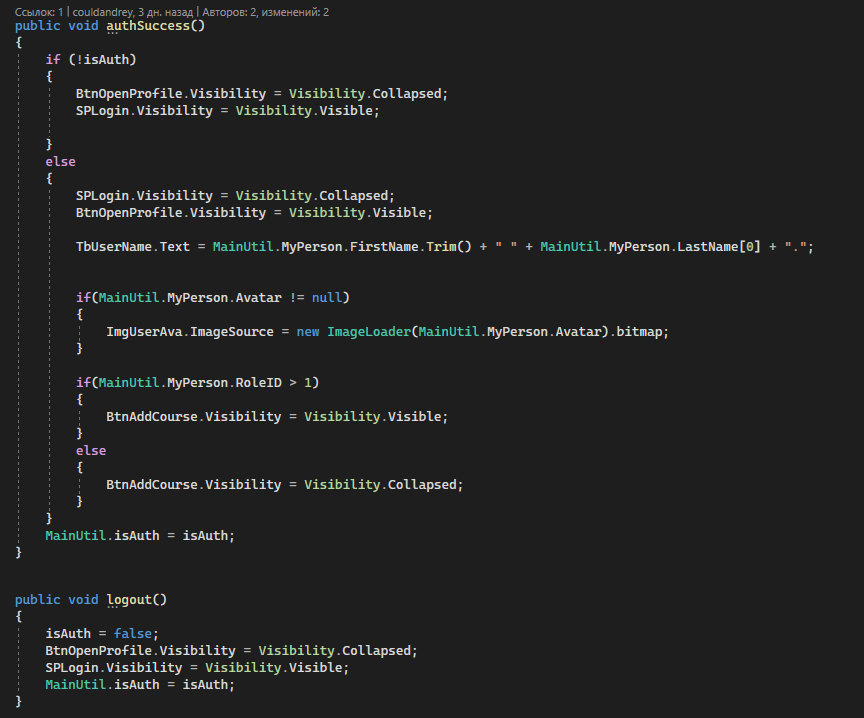


Рисунок 4.32 – Реализация логики в файле «MainWindow.xaml.cs»

Переходим на страницу «LoginPage». Импортируем пространство имен «materialDesign» и переходим к разметке.



Рисунок 4.33 – Разметка «LoginPage»

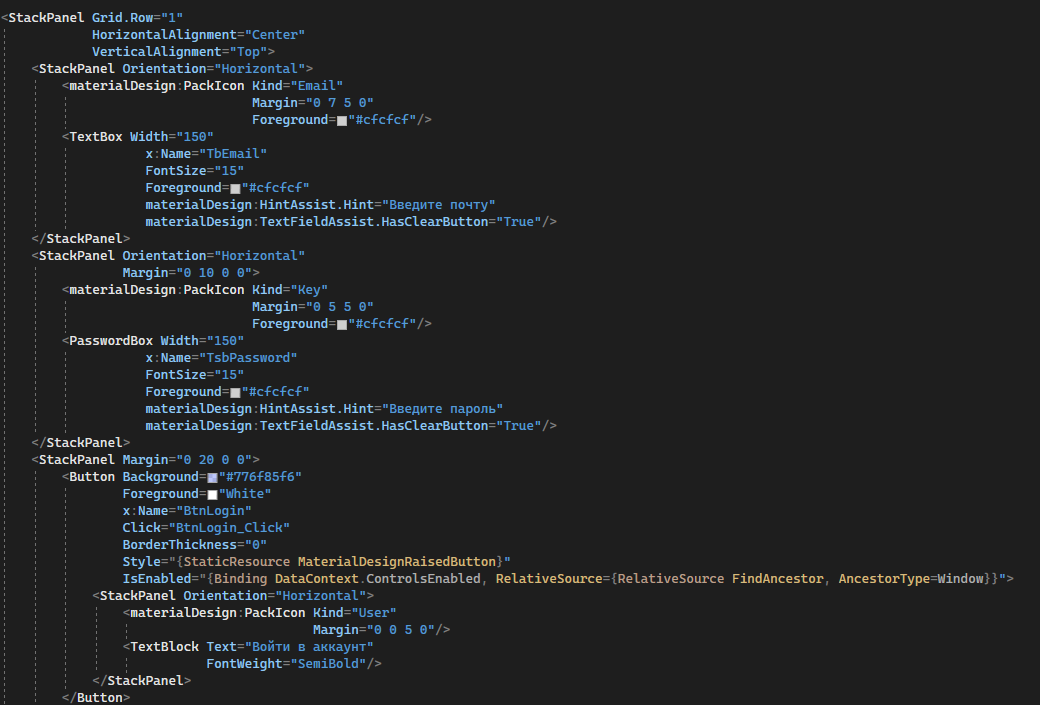


Рисунок 4.34 – Разметка «LoginPage»



Рисунок 4.35 – Разметка «LoginPage»

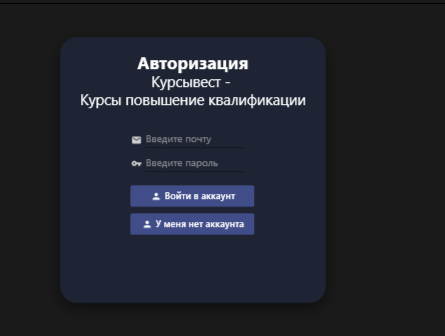


Рисунок 4.36 – Результат разметки «LoginPage»

Далее переходим в файл «LoginPage.xaml.cs». Этот файл будет содержать логику авторизации при клике на кнопку «Войти в аккаунт» будет открываться главная страница с курсами. При нажатии на кнопку «У меня нет аккаунта» будет открываться страница «RegPage».

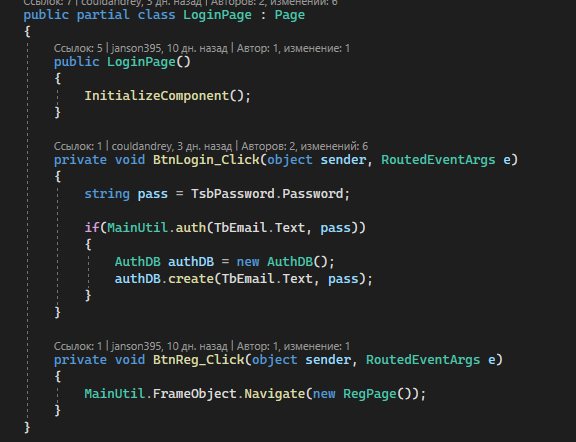


Рисунок 4.37 – Реализации логики файла «LoginPage.xaml.cs»

Приступим к реализации «RegPage».



Рисунок 4.38 – Разметка «RegPage»

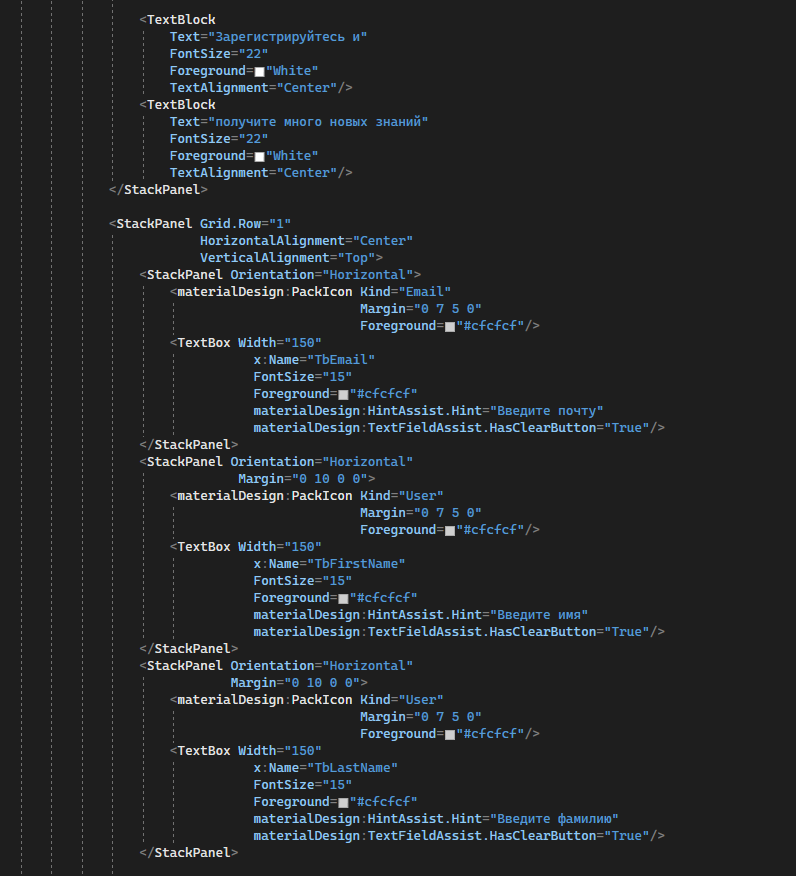


Рисунок 4.39 – Разметка «RegPage»

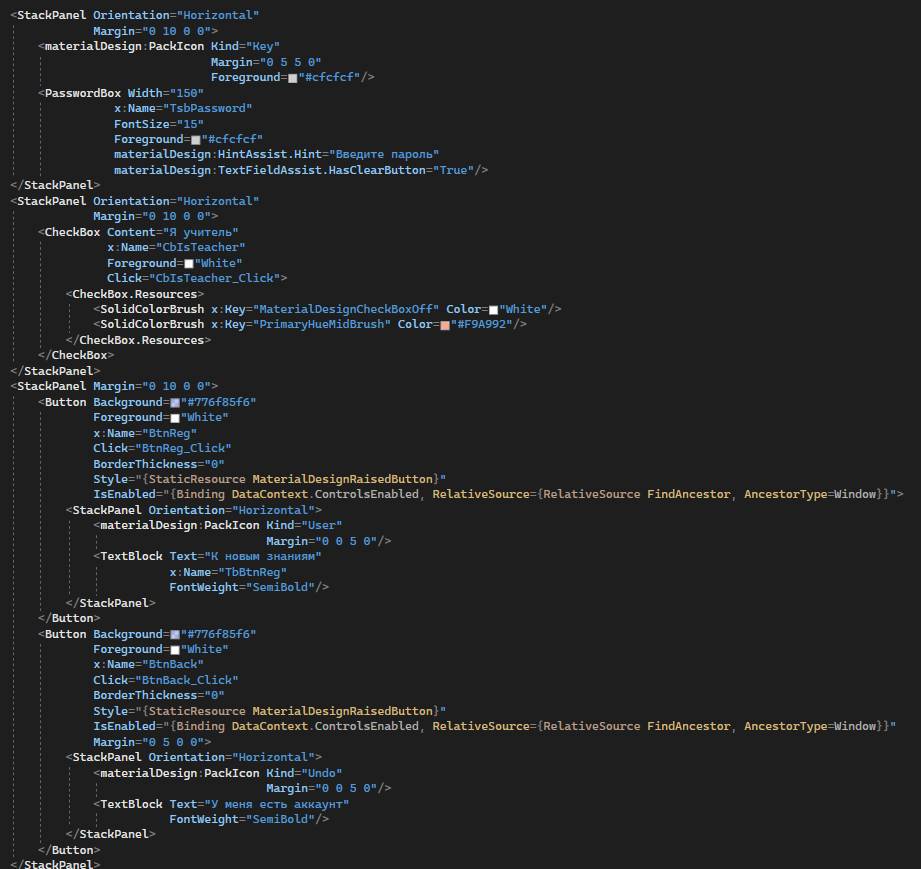


Рисунок 4.40 – Разметка «RegPage»

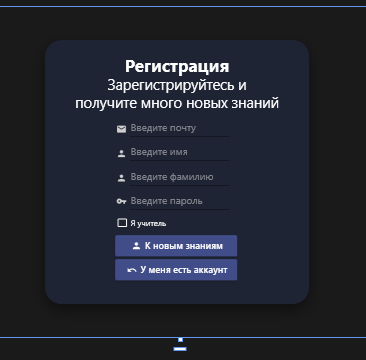


Рисунок 4.41 – Результат разметки «RegPage»

Теперь перейдем в файл «RegPage.xaml.cs» для реализации логики регистрации. Сделаем проверку заполнения полей, проверку на дубликат почты, проверку на ошибку соединения с базой данных, также при успешной регистрации будет выводит диалоговое окно и перенаправлять на страницу «LoginPage»

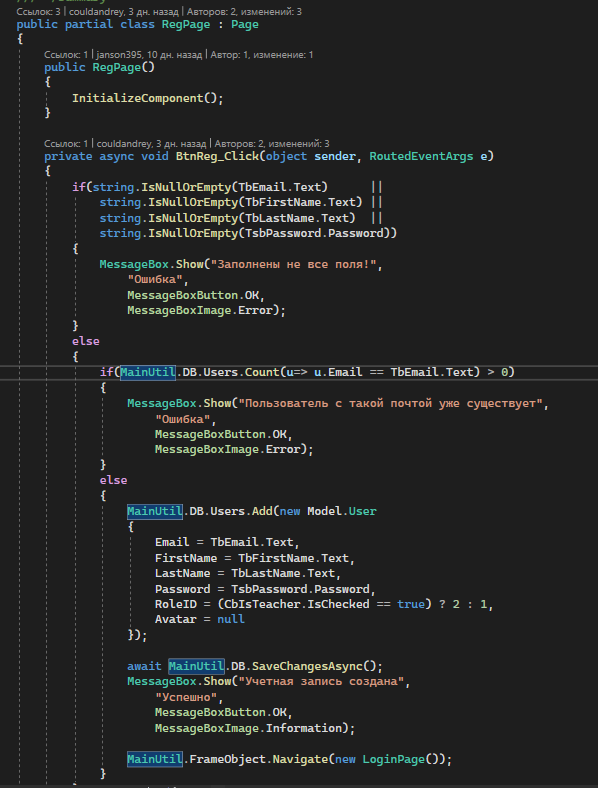


Рисунок 4.41 – Реализация файлы «RegPage.xaml.cs»

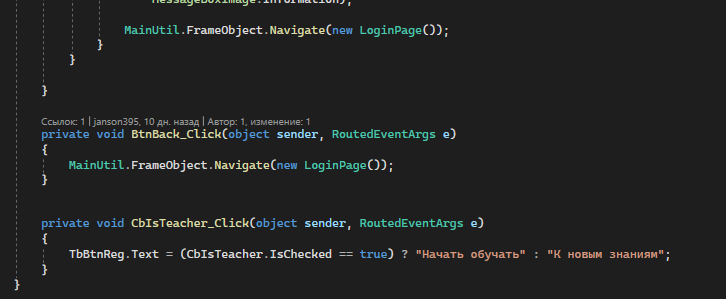


Рисунок 4.42 – Реализация файлы «RegPage.xaml.cs»