Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра информационных систем и технологий

Специальность 1-40 05 01 03 «Информационные системы и технологии»

Специализация «Издательско-полиграфический комплекс»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА**

по дисциплине «Объектно-ориентированное программирование»

Тема: программное средство «Органайзер студента»

Исполнитель

|  |  |  |
| --- | --- | --- |
| cтудент 2 курса группы 2    Руководитель |  | \_\_\_\_\_\_\_\_\_ Ю.П.Михновец  подпись, дата |
| ассистент |  | \_\_\_\_\_\_\_\_\_ А.Д. Новицкая  подпись, дата |
| Курсовой проект защищен с оценкой |  | \_\_\_\_\_\_\_\_\_ |
| Руководитель \_\_\_\_\_\_\_ |  | А.Д. Новицкая |

подпись

Минск 2017

Оглавление

[Введение 3](file:///D:\Учёба\ООП\2%20семестр\Organizer\Записки\Poyasnitelnaya_zapiska_Zagnetova_A_I.docx#_Toc483535249)

[1. Постановка задачи 4](file:///D:\Учёба\ООП\2%20семестр\Organizer\Записки\Poyasnitelnaya_zapiska_Zagnetova_A_I.docx#_Toc483535250)

[1.1. Алгоритмы решения 4](file:///D:\Учёба\ООП\2%20семестр\Organizer\Записки\Poyasnitelnaya_zapiska_Zagnetova_A_I.docx#_Toc483535251)

[1.2. Обзор прототипов 5](file:///D:\Учёба\ООП\2%20семестр\Organizer\Записки\Poyasnitelnaya_zapiska_Zagnetova_A_I.docx#_Toc483535252)

[1.3. Актуальность задачи 7](file:///D:\Учёба\ООП\2%20семестр\Organizer\Записки\Poyasnitelnaya_zapiska_Zagnetova_A_I.docx#_Toc483535253)

[2. Разработка архитектуры проекта 8](file:///D:\Учёба\ООП\2%20семестр\Organizer\Записки\Poyasnitelnaya_zapiska_Zagnetova_A_I.docx#_Toc483535254)

[2.1. Обобщенная структура 8](file:///D:\Учёба\ООП\2%20семестр\Organizer\Записки\Poyasnitelnaya_zapiska_Zagnetova_A_I.docx#_Toc483535255)

[2.2. Диаграмма UML 8](file:///D:\Учёба\ООП\2%20семестр\Organizer\Записки\Poyasnitelnaya_zapiska_Zagnetova_A_I.docx#_Toc483535256)

[3. Разработка функциональной модели данных ПС 10](file:///D:\Учёба\ООП\2%20семестр\Organizer\Записки\Poyasnitelnaya_zapiska_Zagnetova_A_I.docx#_Toc483535257)

[3.1. Выполняемые функции 10](file:///D:\Учёба\ООП\2%20семестр\Organizer\Записки\Poyasnitelnaya_zapiska_Zagnetova_A_I.docx#_Toc483535258)

[4. Руководство пользователя 14](file:///D:\Учёба\ООП\2%20семестр\Organizer\Записки\Poyasnitelnaya_zapiska_Zagnetova_A_I.docx#_Toc483535259)

[Заключение 20](file:///D:\Учёба\ООП\2%20семестр\Organizer\Записки\Poyasnitelnaya_zapiska_Zagnetova_A_I.docx#_Toc483535260)

[Список использованных источников 21](file:///D:\Учёба\ООП\2%20семестр\Organizer\Записки\Poyasnitelnaya_zapiska_Zagnetova_A_I.docx#_Toc483535261)

[Приложение А 22](file:///D:\Учёба\ООП\2%20семестр\Organizer\Записки\Poyasnitelnaya_zapiska_Zagnetova_A_I.docx#_Toc483535262)

[Приложение Б 24](file:///D:\Учёба\ООП\2%20семестр\Organizer\Записки\Poyasnitelnaya_zapiska_Zagnetova_A_I.docx#_Toc483535263)

[Приложение В 27](file:///D:\Учёба\ООП\2%20семестр\Organizer\Записки\Poyasnitelnaya_zapiska_Zagnetova_A_I.docx#_Toc483535264)

[Приложение Г 33](file:///D:\Учёба\ООП\2%20семестр\Organizer\Записки\Poyasnitelnaya_zapiska_Zagnetova_A_I.docx#_Toc483535265)

# **Введение**

На сегодняшний день каждый студент ВУЗа пытается держать в голове свою успеваемость, расписание, постоянно не может вспомнить, «первая или вторая учебная неделя сейчас?». Староста устала каждому передавать информацию от декана и других преподавателей. Все эти проблемы легко позволяет решить разработанное приложение «Органайзер студент».

Его функционал определён лишь из потребностей студента, что исключает всё лишнее. Таким образом, в нём можно гибко отмечать количество сданных/всех лабораторных работ. При необходимости оставить себе заметку. Старосте или декану всего лишь надо оставить записку в этом приложении, и его увидят все, кто должен. Расписание позволяет редактировать его в любое время и его корректность будет зависеть лишь от заполнявшего. Учебная неделя определяется сама, так что студенту не придётся напрягать извилины!

В качестве интерфейса прикладного программирования был выбран Windows Presentation Foundation (WPF). Для работы с WPF использовался объектно-ориентированный язык программирования C#. Кроме того, в WPF использовался декларативный язык описания интерфейса – XAML.

1. **Постановка задачи**

Целью курсового проекта является разработка органайзера для студентов.

Основной функционал приложения:

* Авторизация;
* Хранение личной информации в базе данных;
* Хранение и отображение прогресса по предметам;
* Расписание предметов;
* Заметки;
* Объявления;

## **1.1. Алгоритмы решения**

Основная задача при разработке заключалась в подборе наиболее дружественного интерфейса для пользователя и соответствующей бизнес-логики. Для начала, пользователь должен авторизироваться чтобы получить доступ к приложению. Если пользователь не имеет не имеет учётной записи, её стоит создать на форме регистрации. Способ хранения данных пользователя позволяет однозначно разделить студентов на курсы-группы-подгруппы.



Рисунок 1 – Алгоритм работы с программой

Наиболее важными аспектами были корректная проектировка базы данных, позволяющая гибко работать с «сущностями», а так же строгая валидация при регистрации.

## **1.2. Обзор прототипов**

Сейчас существует довольно много приложений, облегчающим жизнь студентам.

My Stud Life – кроссплатформенный планировщик для студентов, учителей, преподавателей, делающий слежение за учебными буднями проще. Позволяет вам легко следить за уроками, домашним заданием или экзаменами.

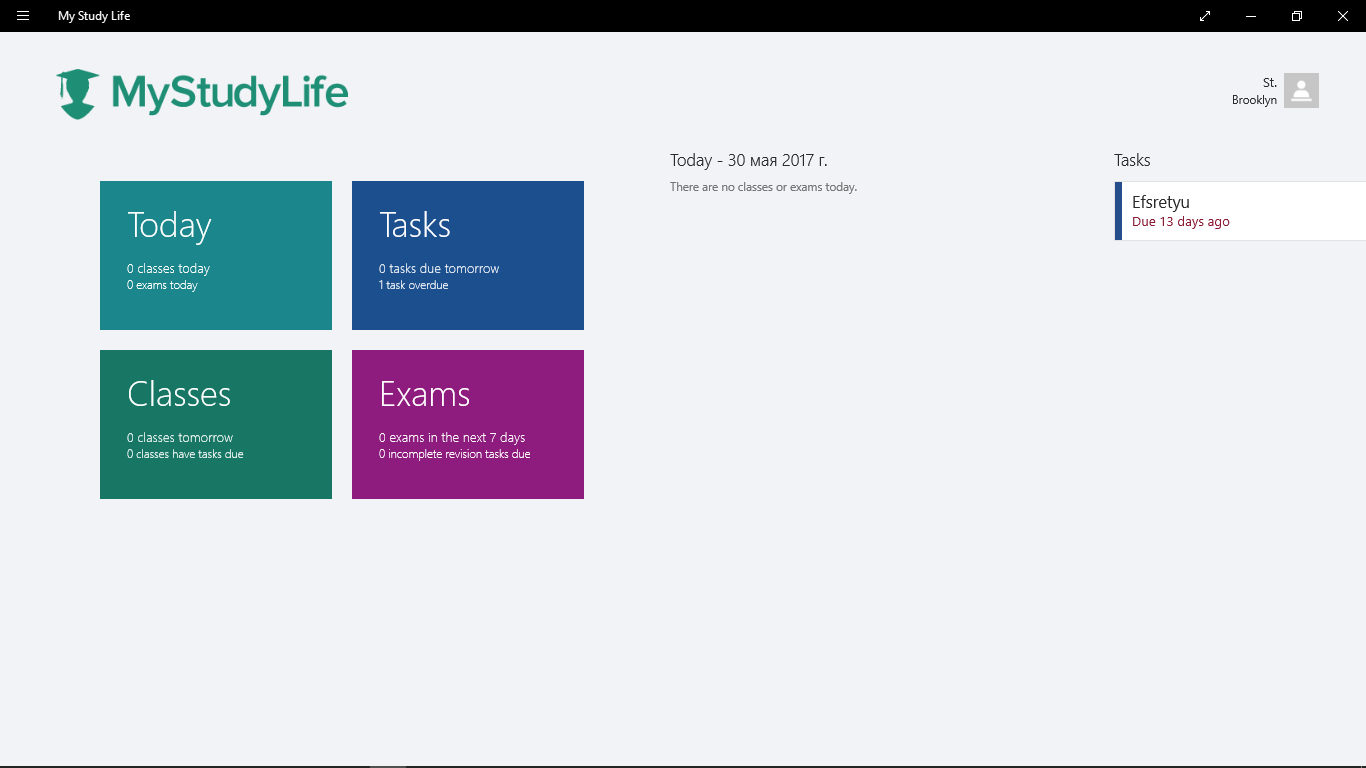


Рисунок 2 – Основное окно в My Stud Life

Student’s TimeTable – простое приложение, позволяющее пользователю легко контролировать его расписание, показывая только нужные предметы на ближайшую неделю. Приложение идеально подходит для студентов, школьников и других людей с богатым расписанием.

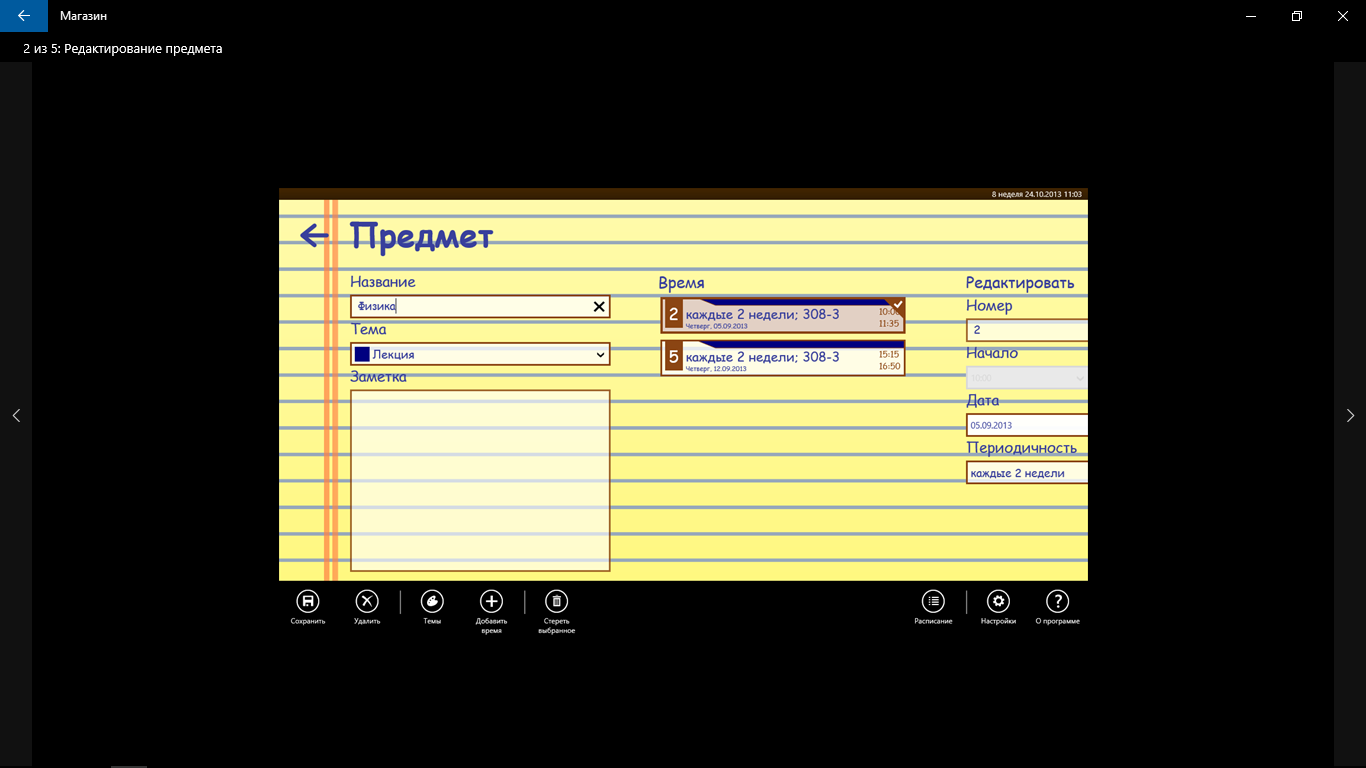


Рисунок 3 – Настройка предмета

## **1.3. Актуальность задачи**

В настоящее время существует большое множество приложений для студентов или школьников. Каждое из них поражает своими возможностями, дизайном и гибкостью использования. Но, вероятно, никто из вас не любил заполнять дневник в школе? Это и есть та проблема, которую никакой из аналогов не решает — вы должны всё делать сами для себя.

В тот момент, когда мне стало интересно отметить свои успехи, я не смог найти подходящего мне приложения, поэтому и выбрал это темой курсового проекта. В данном приложении реализовано всё для удобства студента, а так же старост и администрации. Здесь расписание заполняют все для всех. То есть если из вашей подгруппы это кто-то сделал, то вам не придётся это делать. Имея расписание, вы легко можете начать следить за прогрессом по своим предметам, даже не зная сколько всего вам задач предстоит.

1. **Разработка архитектуры проекта**

## **2.1. Обобщенная структура**

Структура проекта представлена на рисунке 4.

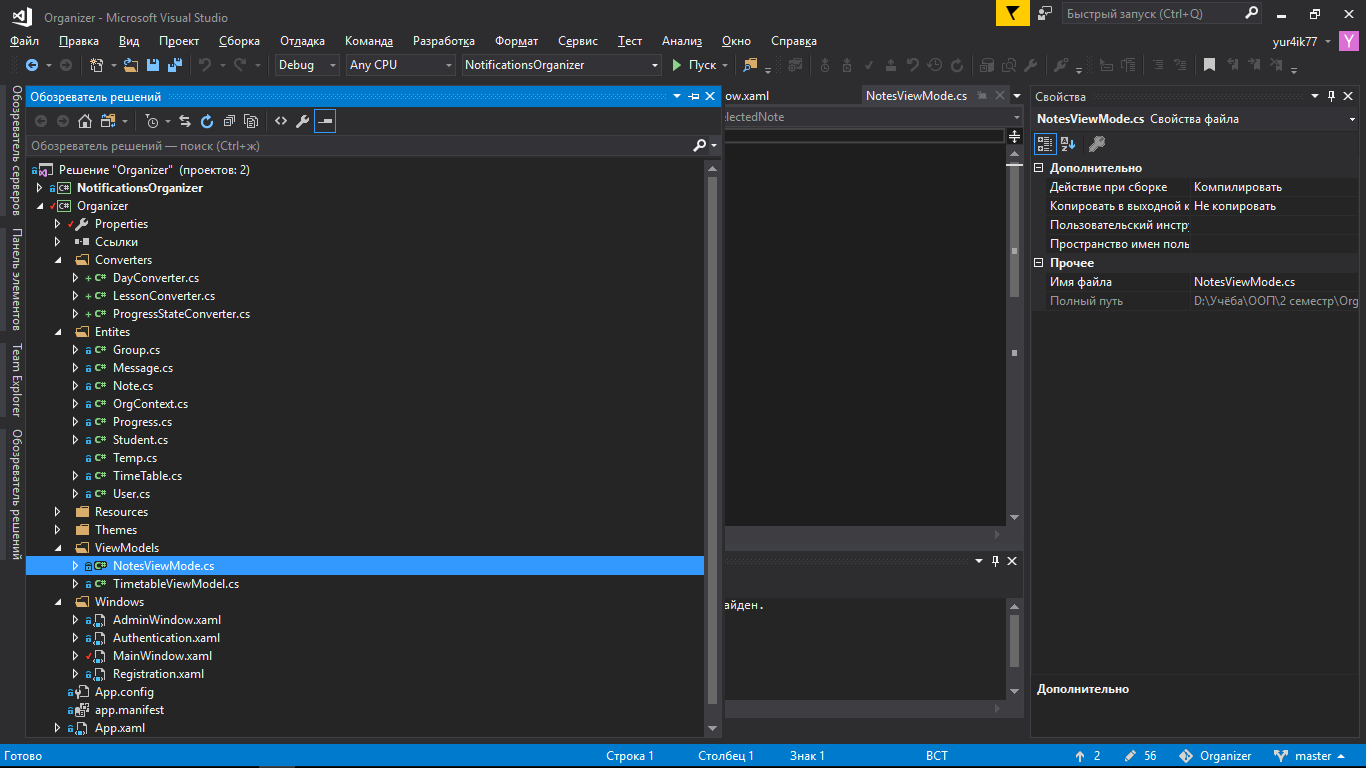


Рисунок 4 – Структура проекта

Таблица 1 – Описание структуры проекта

|  |  |
| --- | --- |
| Имя файла | Содержание |
| Properties | Свойства проекта, содержит информацию о сборке используемых ресурсах и настройках. |
| App.config | Файл с параметрами проекта. |
| Entities | Подпространство имён, содержащее опосание сущностей для работы с БД |
| AdminWindow.xaml | Форма приложения для администратора |
| MainWindow.xaml | Форма для обычно пользователя |
| Authentication.xaml | Форма авторизации |
| Registration.xaml | Фора для регистрации. |
| Converters | Конвертеры представлений и типов данных для отображения их в ListBox |
| app.manifest | Файл манифеста сборки, определяющий её конфигурацию |
| ViewModels | Модель, определяющая логику работы интерфейса |

## **2.2. Диаграмма UML**

Диаграмма классов представлена на рисунке 5.

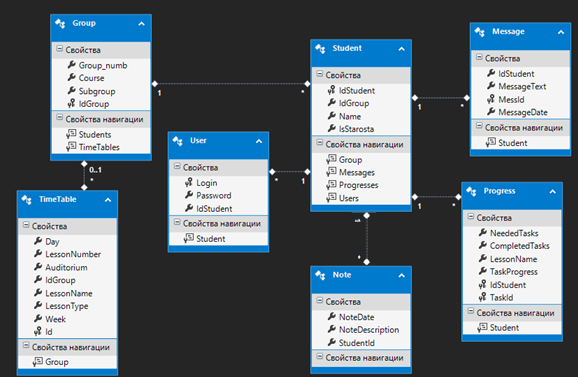


Рисунок 5 – Диаграмма классов

В вершине иерархии зависимостей стоит класс Group. Он характеризует группу курсом, номером группы, подгруппой и уникальным идентификатором. Далее, с ним связаны классы TimeTable, описывающий расписание для одной подгруппы и Student — абстракция студента. Со студентом взаимодействуют такие сущности, как Message, Progress, Note и User. Через них предоставляется основной функционал хранения/записи данных пользователя в базу данных.

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

## **3.1. Выполняемые функции**

Рассмотрим реализацию нескольких функций для работы с базой данных.

**public void \_log\_in\_Click** (object sender, RoutedEventArgs e) – метод, производящий авторизацию пользователя. Если данные введены корректно, метод создаёт соответствующее основное окно приложения — для администратора или обычного юзера. Анализируется, был ли выбран чекбокс \_rememberUser и создаёт файл восстановления данных пользователя для последующих запусков приложения. Если соответствующей записи в БД не найдено — обрабатывается исключение, связанное с этим и сообщается пользователю.

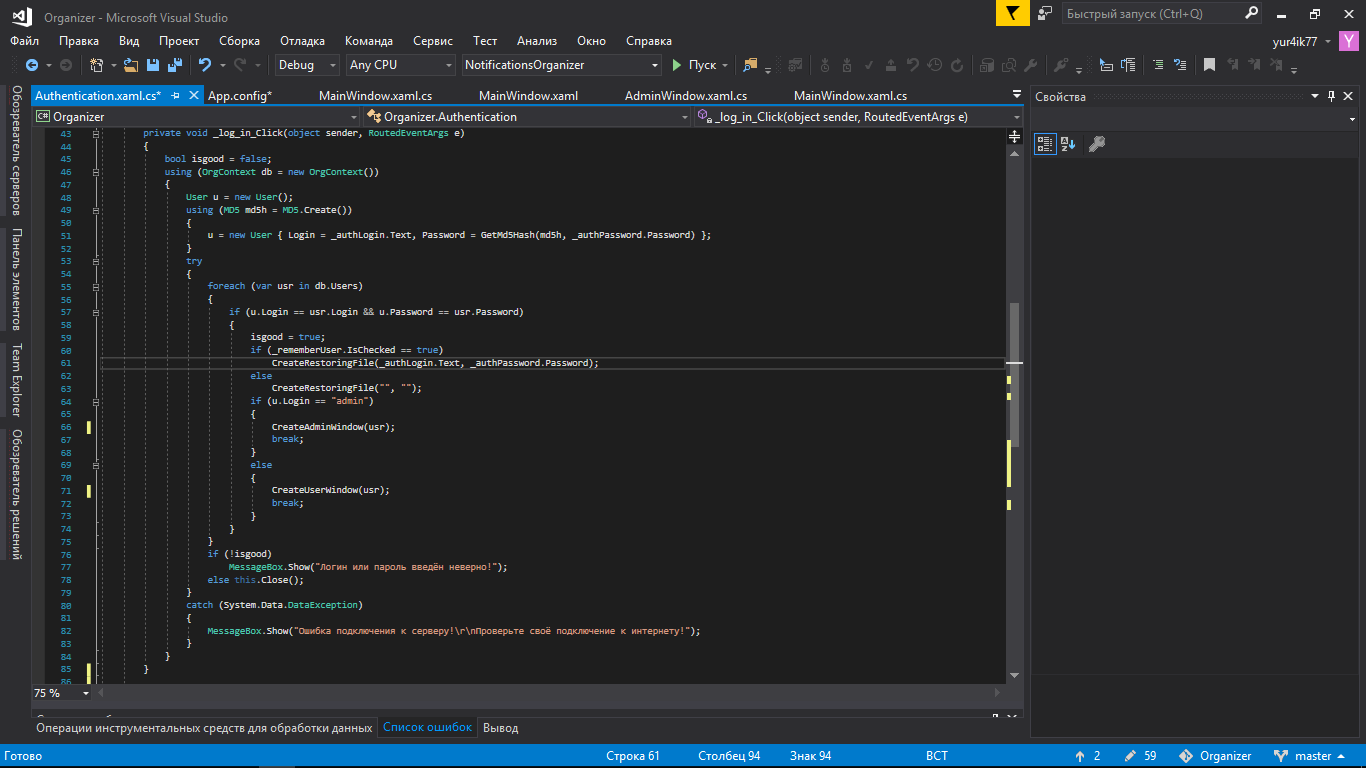


Рисунок 6 – Функция авторизации

Конструктор **public MainWindow** (User u) – данный метод инициализирует основную форму приложения и подписывает методы, содержащие бизнес-логику на соответствующие события. Создаёт объект класса Student, через который идёт дальнейшая работа с БД. Если это первый запуск приложения на устройстве, добавляет приложение, отвечающие за уведомление в автозапуск, путём создания соответствующего ключа в реестре.

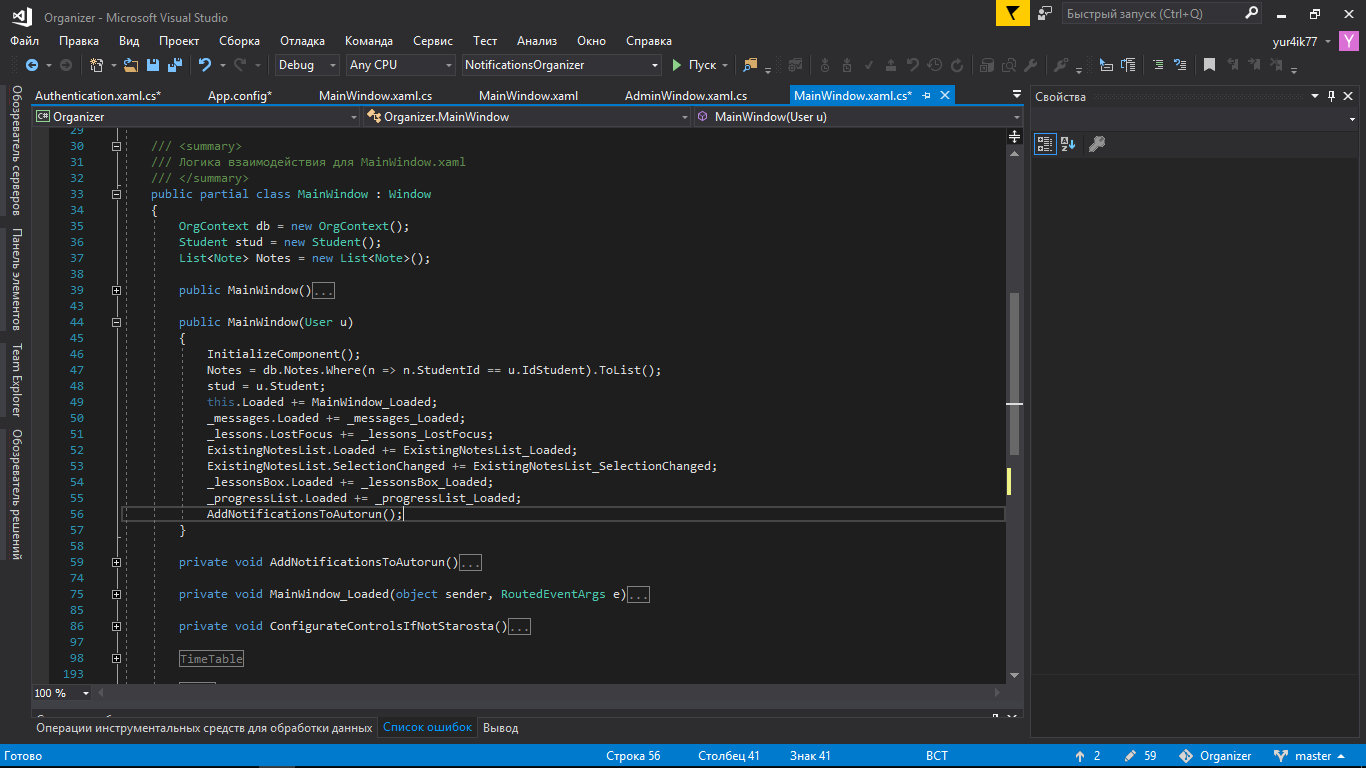


Рисунок 7 – Метод создания основной формы.

**3.2. Модель базы данных**

База данных для приложения MSSQL, располагается в облачном хранилище, что позволяет всем пользователям приложения пользоваться одной базой данных, концентрирующей в себе все данные.

База данных содержит в себе 7 таблиц: Users, Students, Messages, Groups, TimeTables, Notes, Progresses. Они представлены ниже вместе со своими структурами.

**Groups**

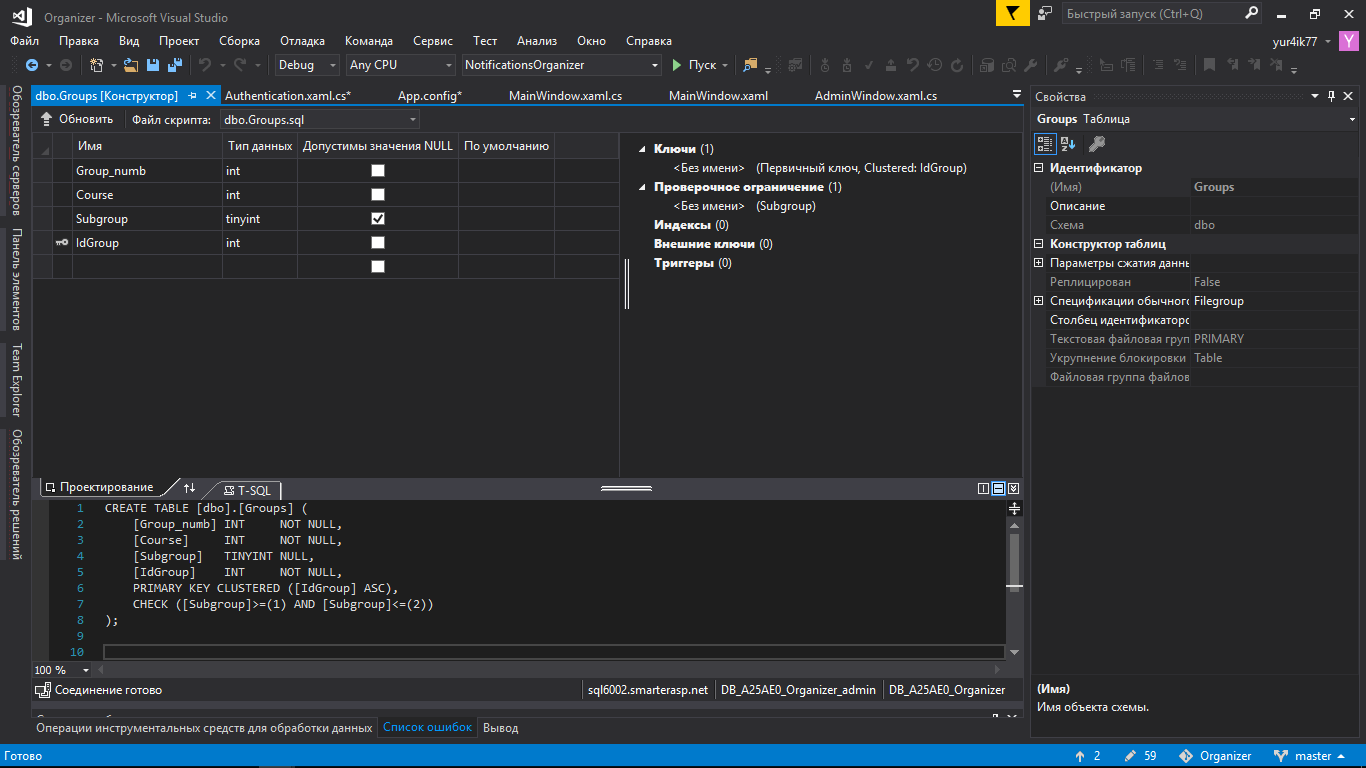


Рисунок 8 – Таблица Groups

Таблица 2 – структура таблицы Groups

|  |  |
| --- | --- |
| Название столбца | Содержание |
| Group\_numb | Номер группы |
| Course | Курс |
| Subgroup | Подгруппа |
| IdGroup | Идентификатор группы |

**Messages**

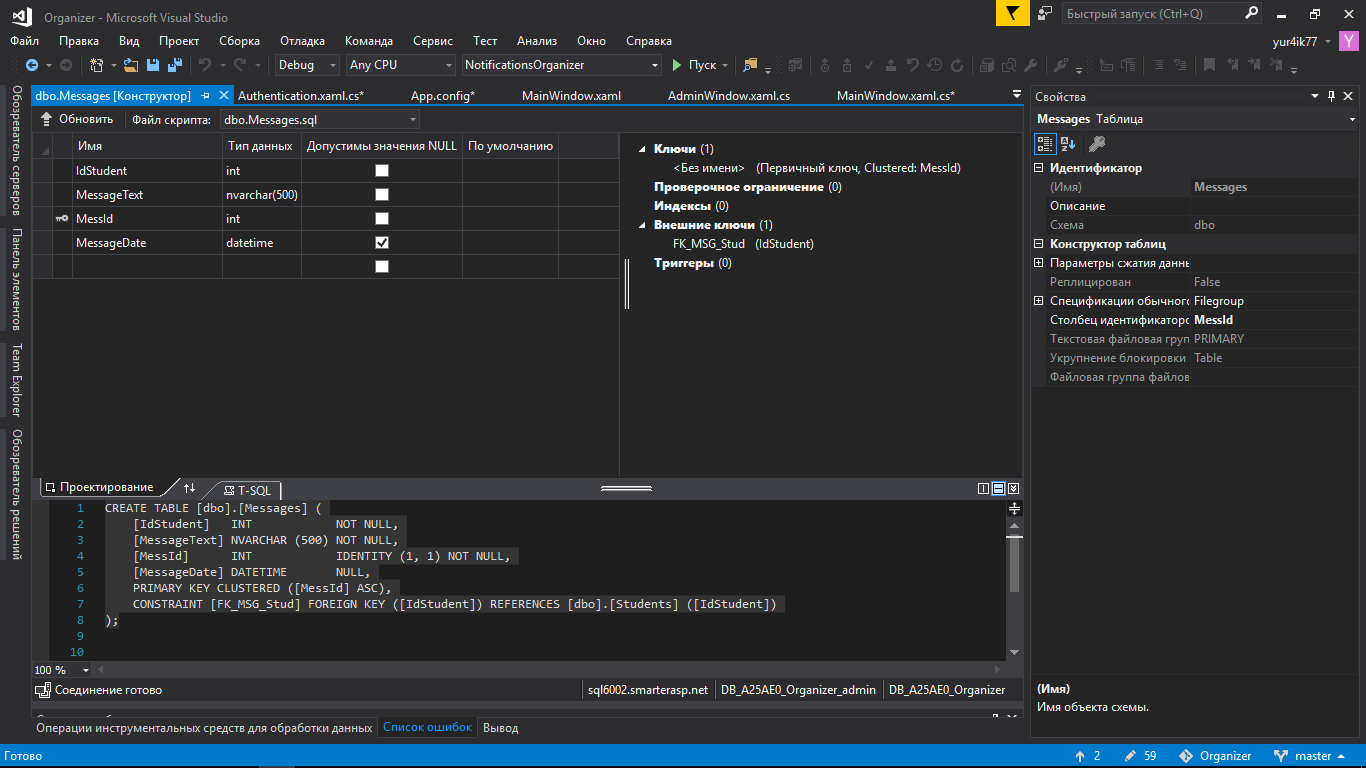


Рисунок 9 – Таблица Messages

Таблица 3 – содержание таблицы Messages

|  |  |
| --- | --- |
| Название столбца | Содержание |
| IdStudent | Идентификатор студента |
| MessageText | Текст сообщения |
| MessageDate | Дата отправки сообщения |
| MessId | Уникальный номер записи |

**Notes**

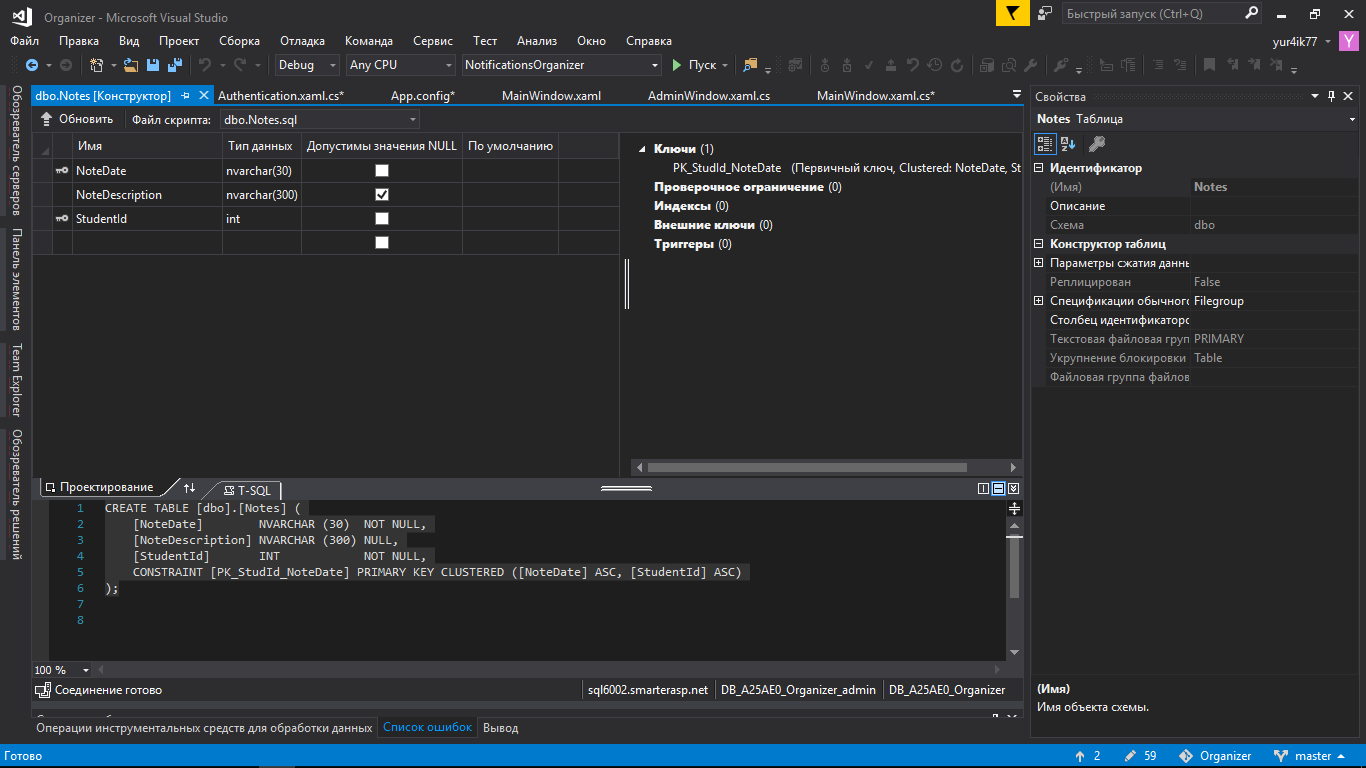


Рисунок 10 – Таблица Notes

Таблица 4 – структура таблицы Dialogs

|  |  |
| --- | --- |
| Название столбца | Содержание |
| NoteDate | Дата заметки |
| NoteDescription | Содержание заметки |
| StudentId | Идентификатор студента |

**Progresses**

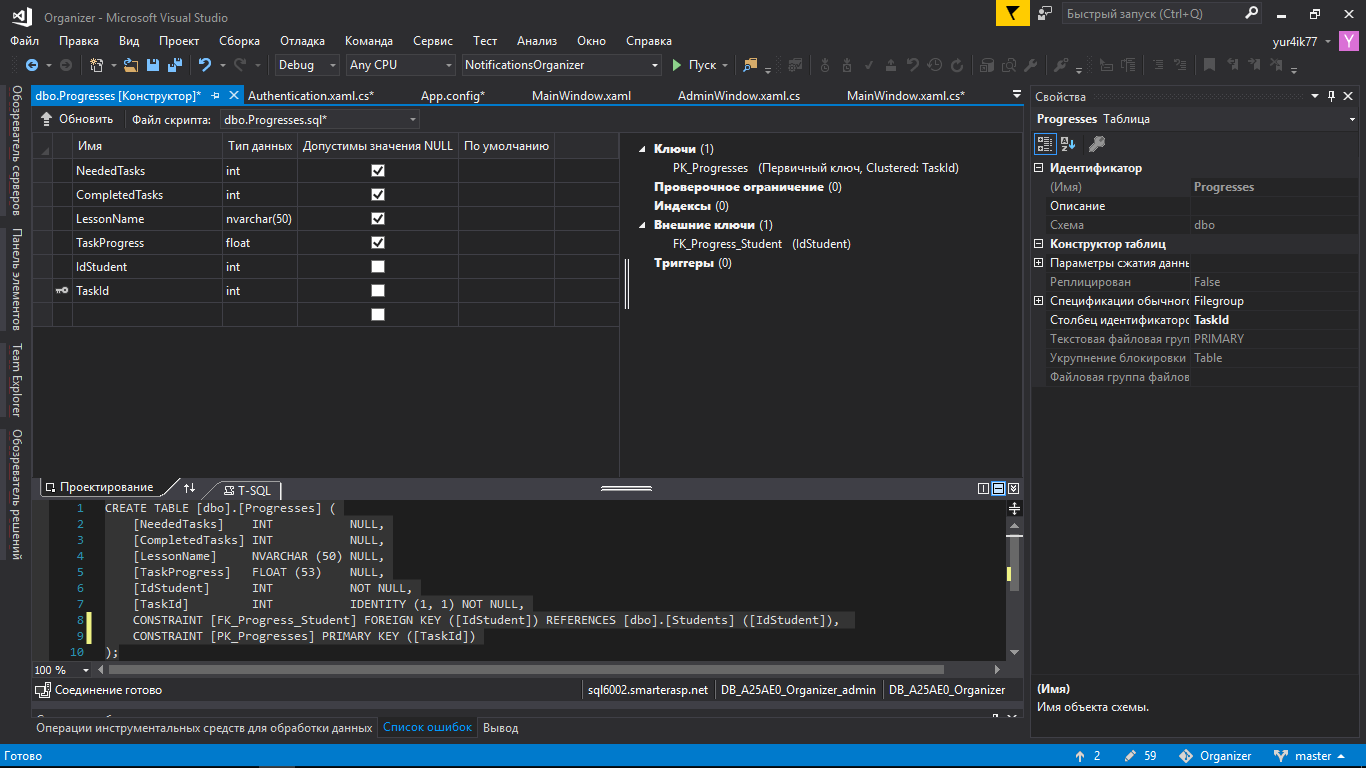


Рисунок 11 – Таблица Progresses

Таблица 5 – содержание таблицы Progresses

|  |  |
| --- | --- |
| Название столбца | Содержание |
| NeededTasks | Кол-во заданий к выполнению |
| CompletedTasks | Кол-во выполненных заданий |
| LessonName | Название предмета |
| TaskProgress | Процент выполнения |
| IdStudent | Идентификатор студента |
| TaskId | Уникальный номер записи |

**Students**

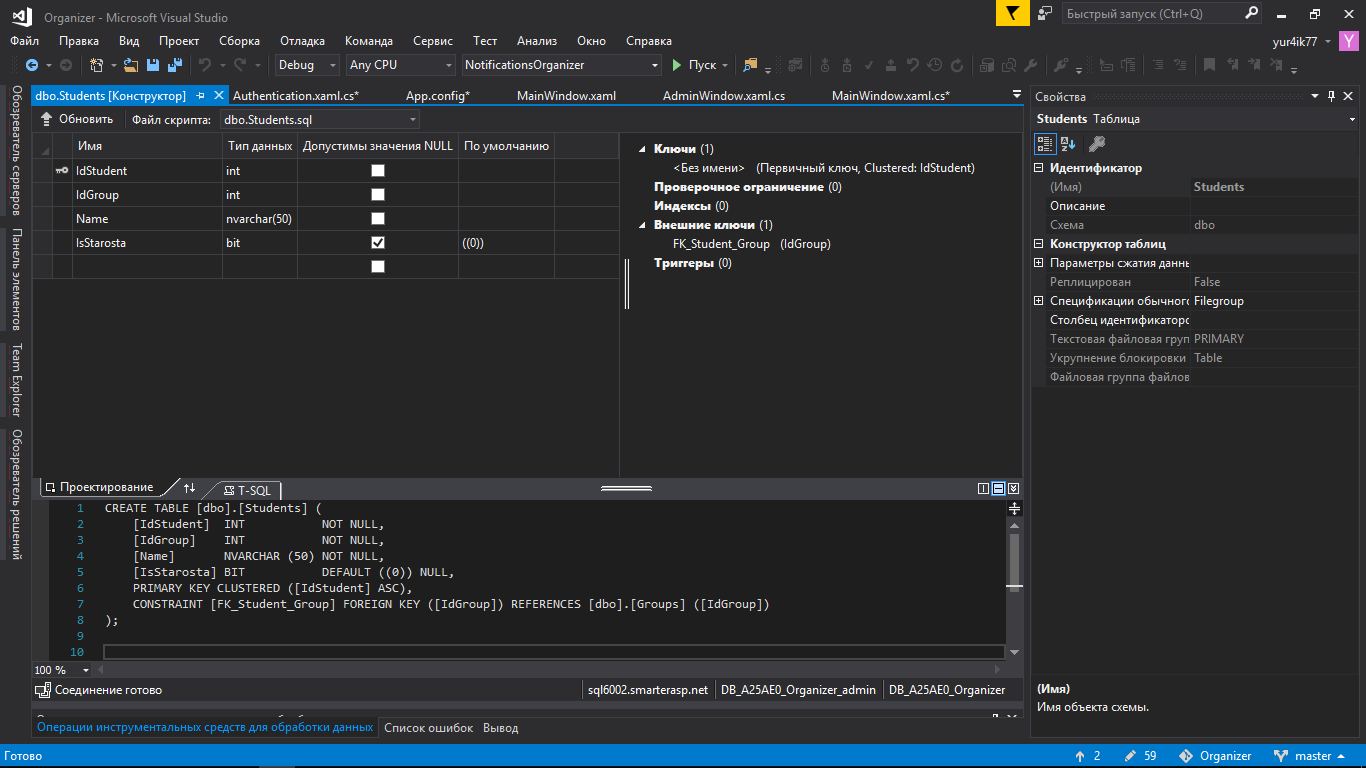


Рисунок 12 – Таблица Students

Таблица 6 – содержание таблицы Students

|  |  |
| --- | --- |
| Название столбца | Содержание |
| IdStudent | Идентификатор студента |
| IdGroup | Идентификатор группы |
| Name | ФИО студента |
| IsStarosta | Является ли старостой |

**TimeTables**

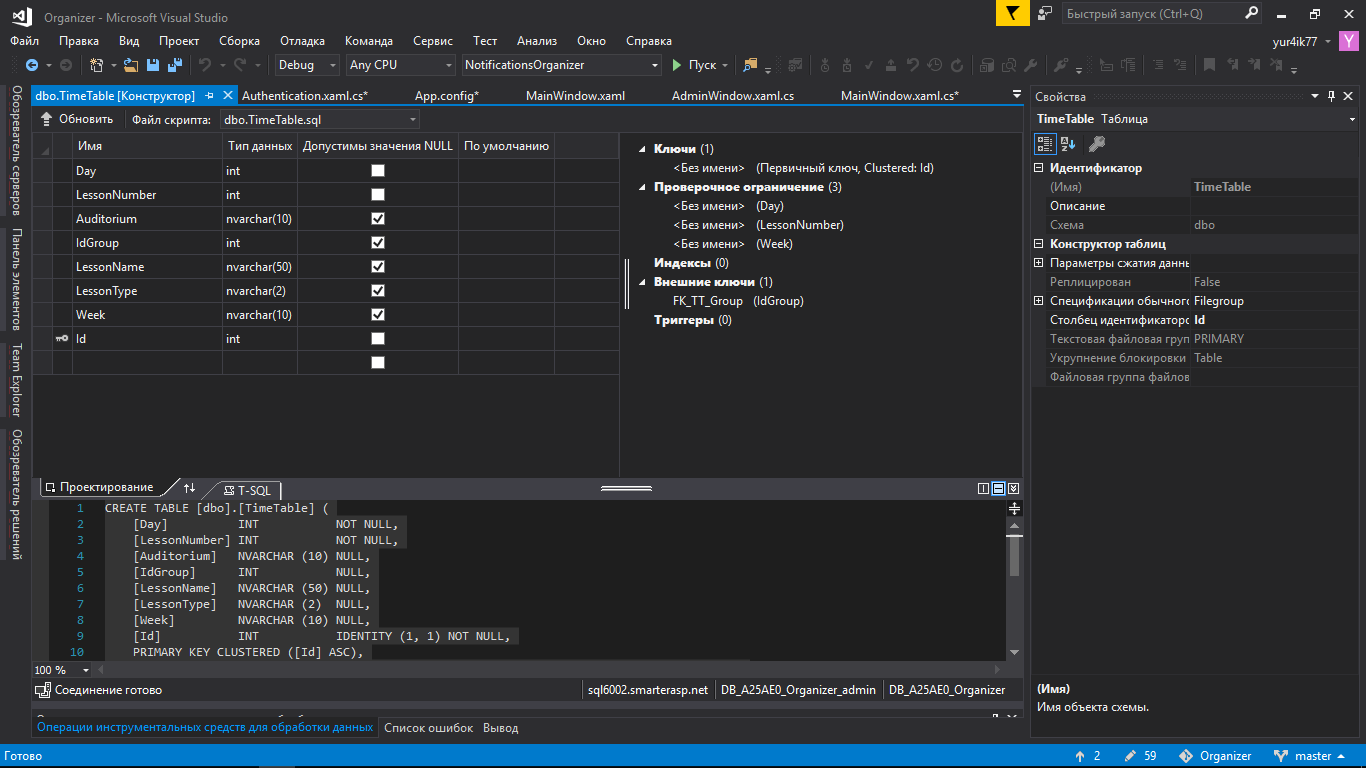


Рисунок 13 – Таблица TimeTables

Таблица 7 – содержание таблицы TimeTables

|  |  |
| --- | --- |
| Название столбца | Содержание |
| Day | Номер дня в неделе |
| LessonNumber | Номер пары в расписании |
| Auditorium | Номер аудитории |
| IdGoup | Идентификатор группы |
| LessonName | Название предмета |
| LessonType | Тип пары |
| Week | Номер учебной недели |
| Id | Уникальный номер записи |

**Users**

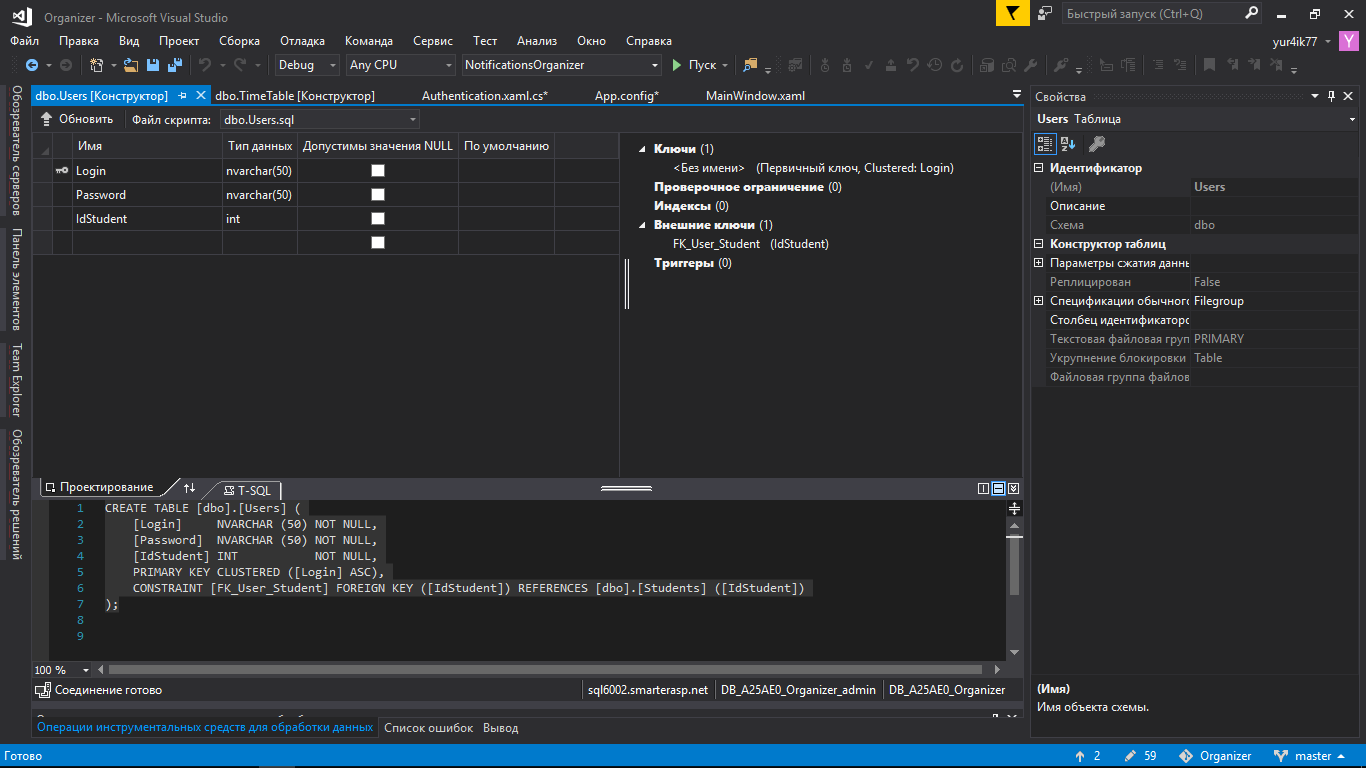


Рисунок 14 – Таблица Users

Таблица 8 – содержание таблицы Users

|  |  |
| --- | --- |
| Название столбца | Содержание |
| Login | Логин пользователя |
| Password | Пароль пользователя |
| IdStudent | Идентификатор пользователя |

# **4. Руководство пользователя**

При запуске приложения пользователь видит окно авторизации.

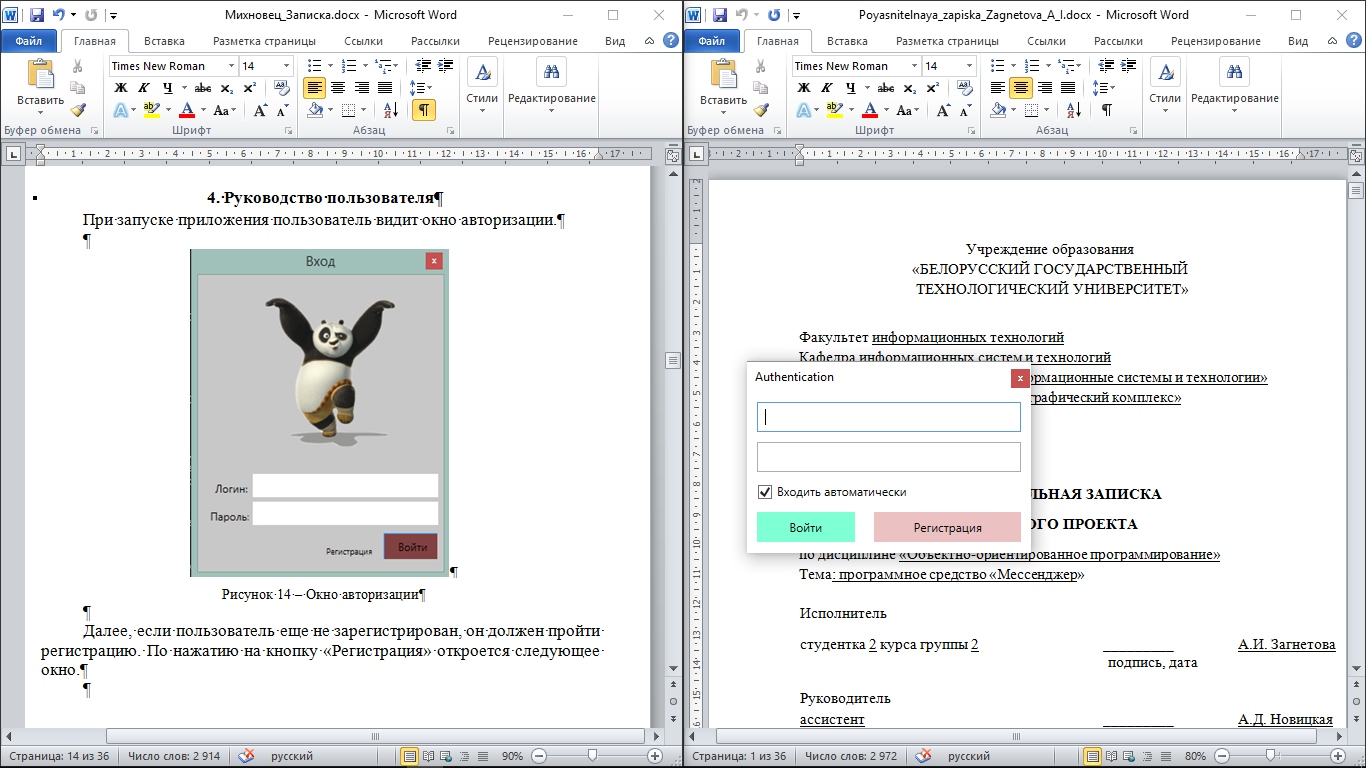


Рисунок 15 – Окно авторизации

Далее, если пользователь еще не зарегистрирован, он должен пройти регистрацию. По нажатию на кнопку «Регистрация» откроется следующее окно.

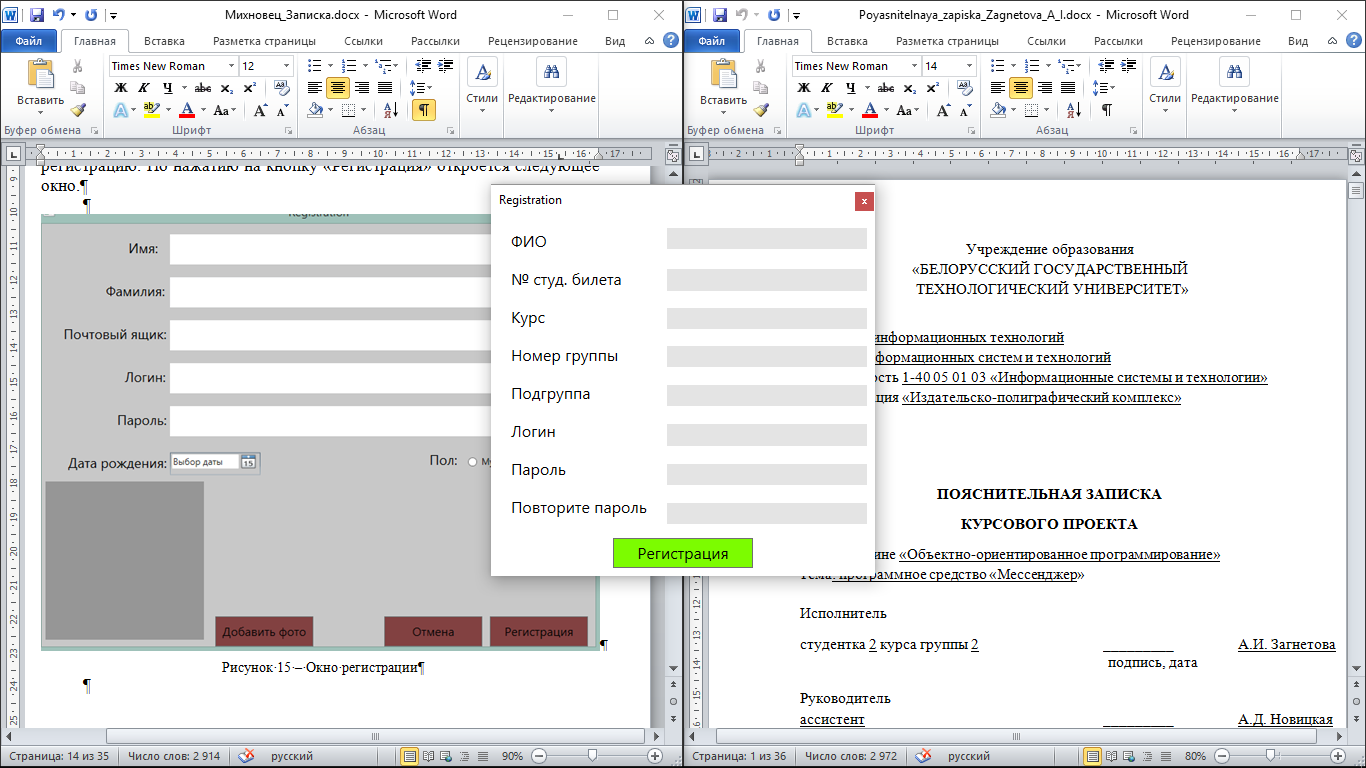


Рисунок 16 – Окно регистрации

При вводе некорректных данных пользователь увидит соответствующие сообщения об ошибках.

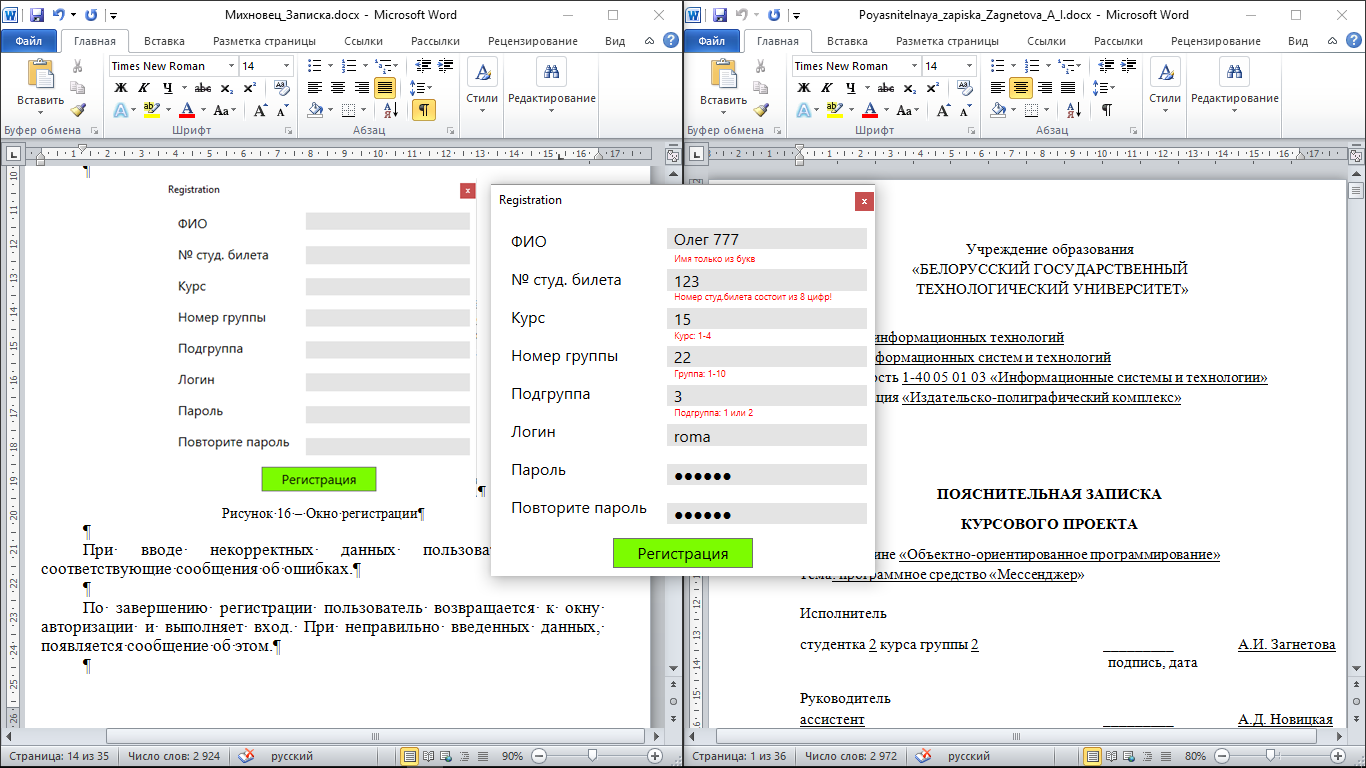


Рисунок 17 – Некорректные данные при регистрации

По завершению регистрации пользователь возвращается к окну авторизации и выполняет вход. При неправильно введенных данных, появляется сообщение об этом.

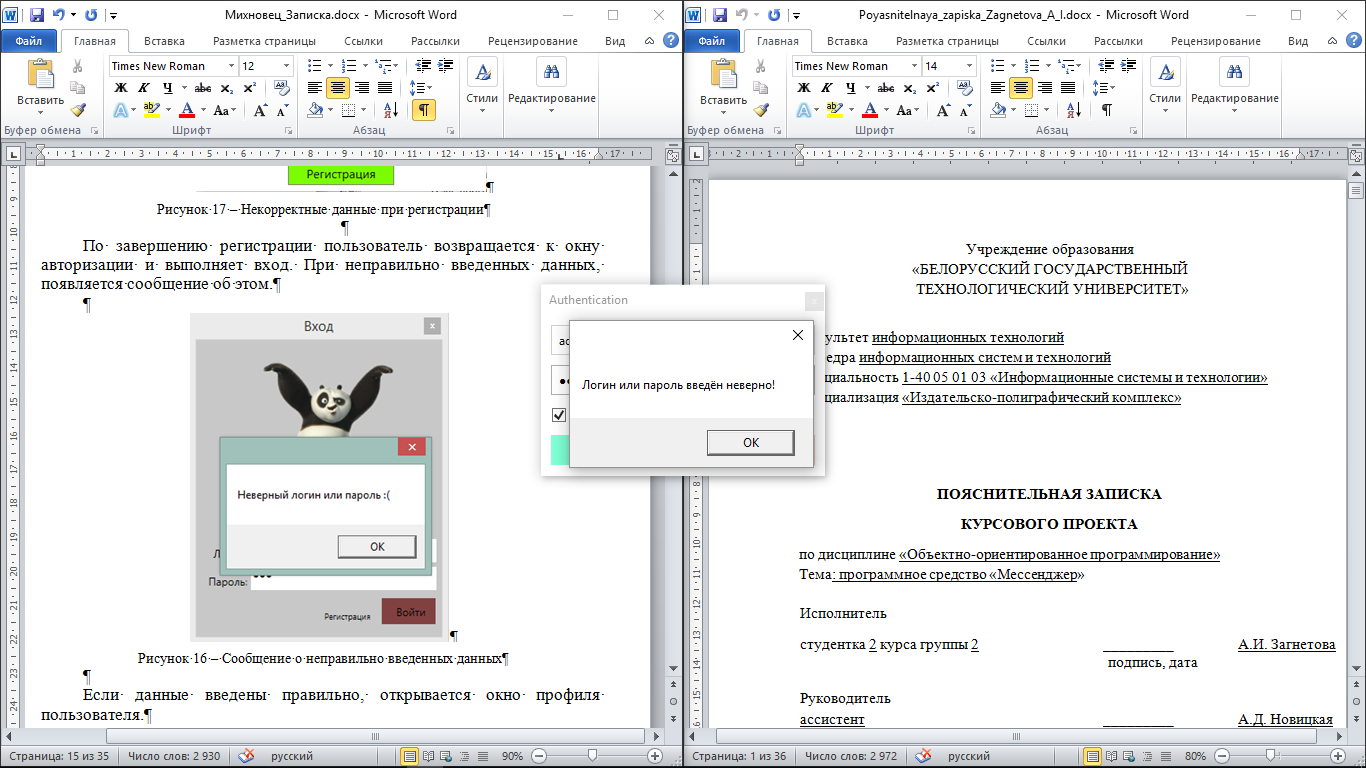


Рисунок 18 – Сообщение о неправильно введенных данных

Если данные введены правильно, открывается окно профиля пользователя.

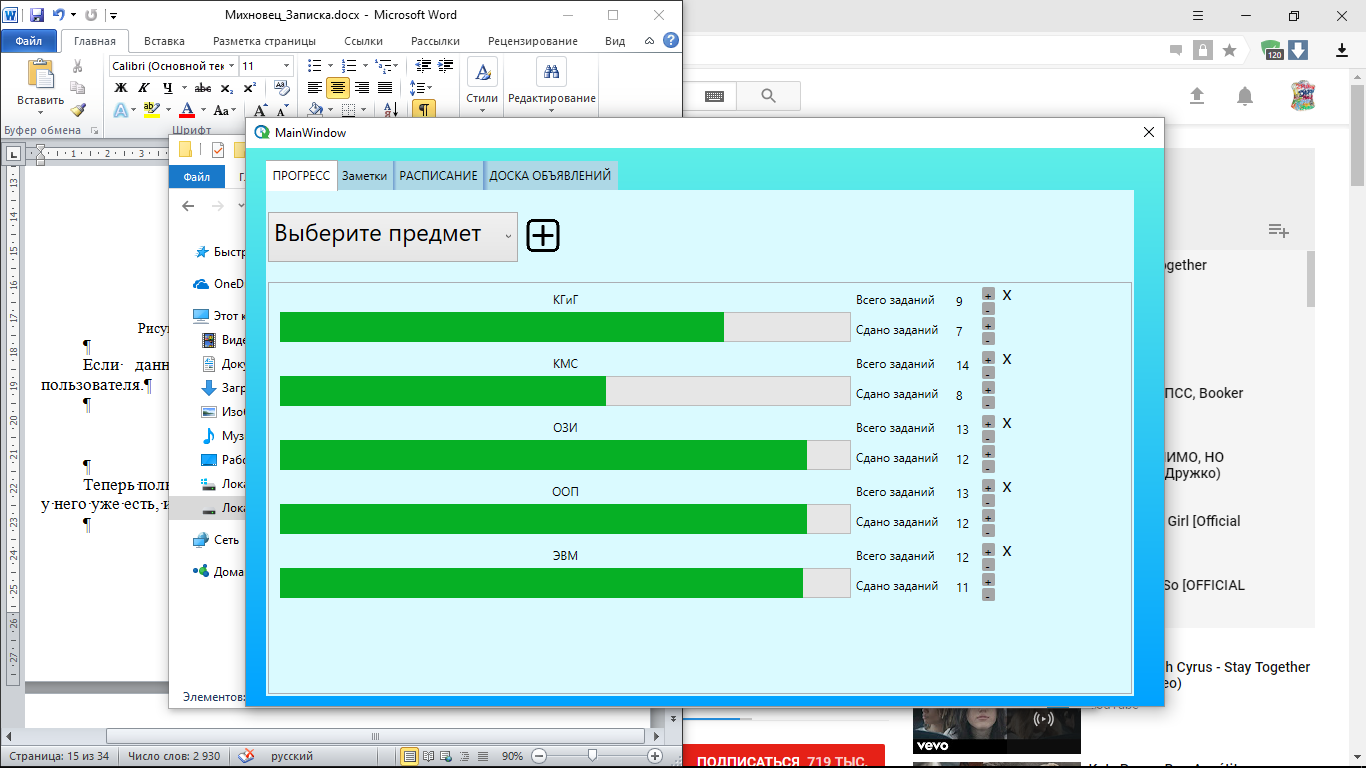


Рисунок 19 – Основное окно программы

Теперь пользователь имеет доступ к заметкам, расписанию, сообщениям и возможности отслеживать прогресс по предметам.

Для отслеживания прогресса по предмету стоит выбрать его в выпадающем списке и нажать на кнопку “+”. Для настройки прогресса есть кнопки “+” и “-” для соответствующих действий.

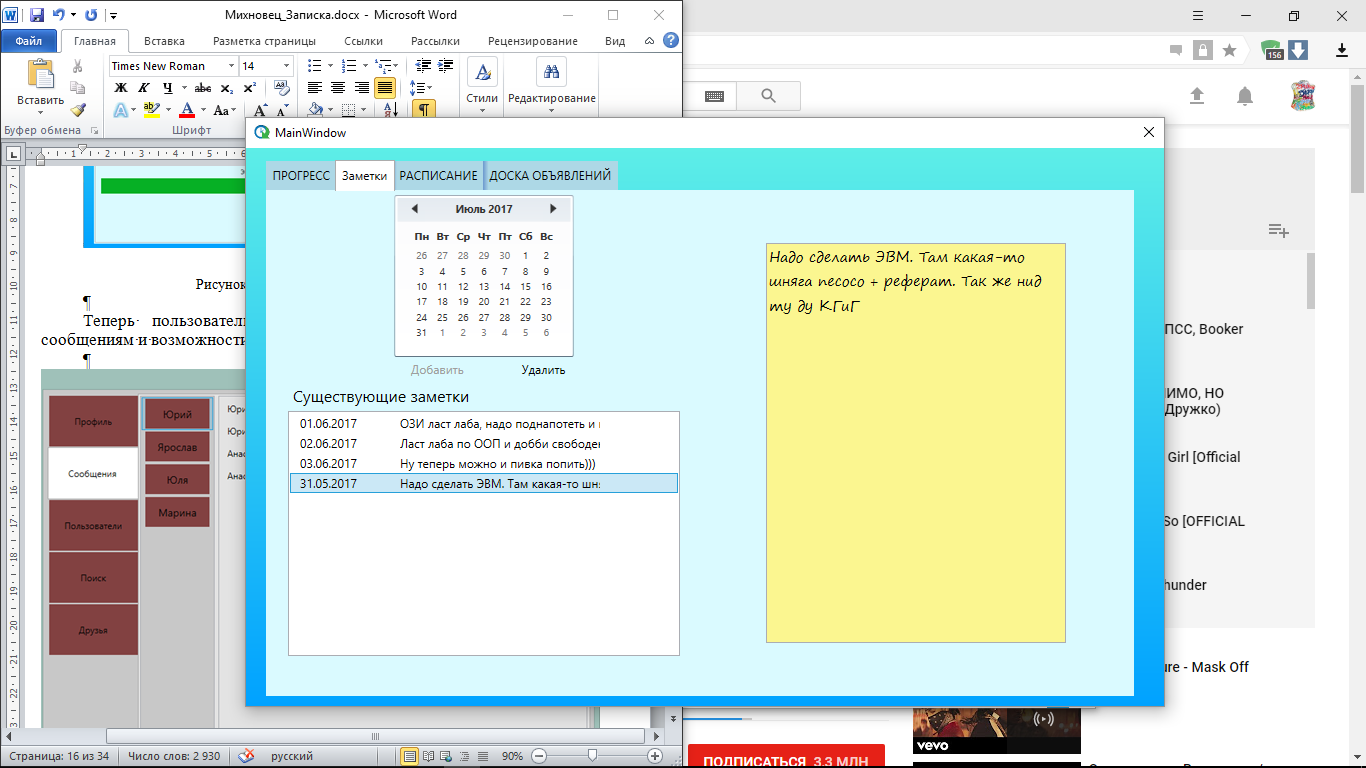


Рисунок 20 – Заметки пользователя

Переключая даты на календаре можно просматривать свои заметки. Если заметка на дату существует — она отобразится в поле справа. Так же заметки можно выбирать в списке заметок под календарём. Если в заметке нет текста(пустая), то она удаляется.

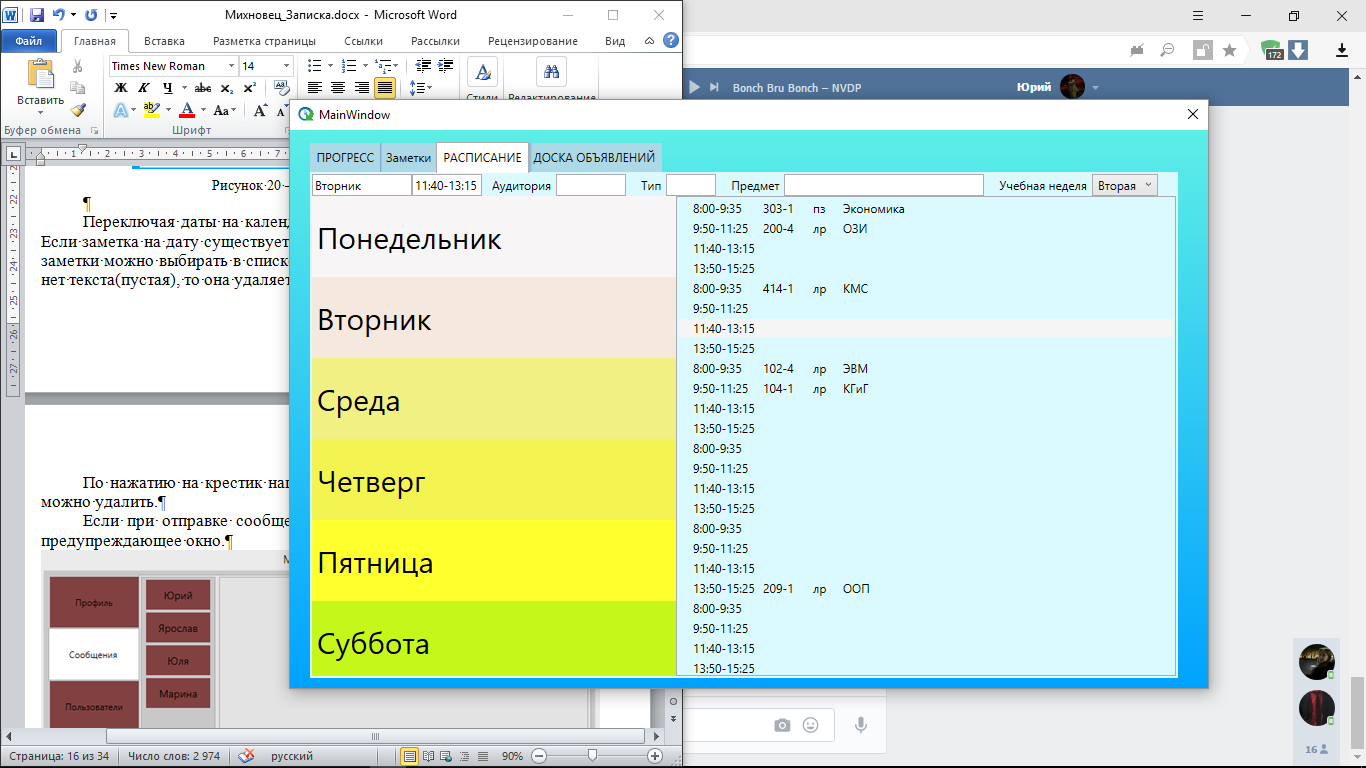


Рисунок 21 – Вкладка для работы с расписанием

Для редактирования расписания следует выбрать строку с нужной парой и ввести необходимые данные в поля сверху. Данные сохраняются автоматически. Для переключения между первой и второй недели используется combobox в правом верхнем углу.



Рисунок 22 – Вкладка с сообщениями

В режиме администратора можно редактировать учётные записи пользователей, с соответствующими ограничениями.

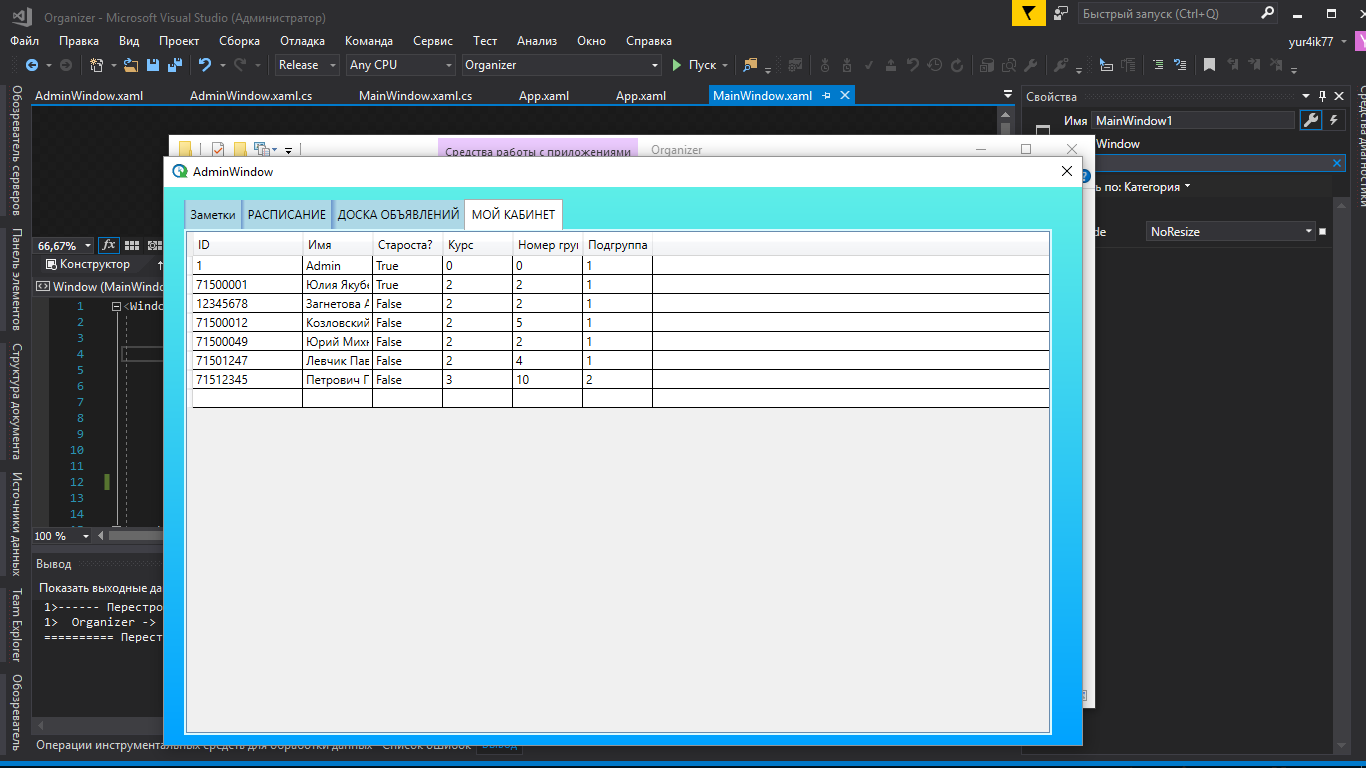


Рисунок 23 – Режим администратора

После первого запуска приложение в автозапуск добавляется приложение, оповещающее о лабораторных работах на завтра и заметки на ближайшие дни.

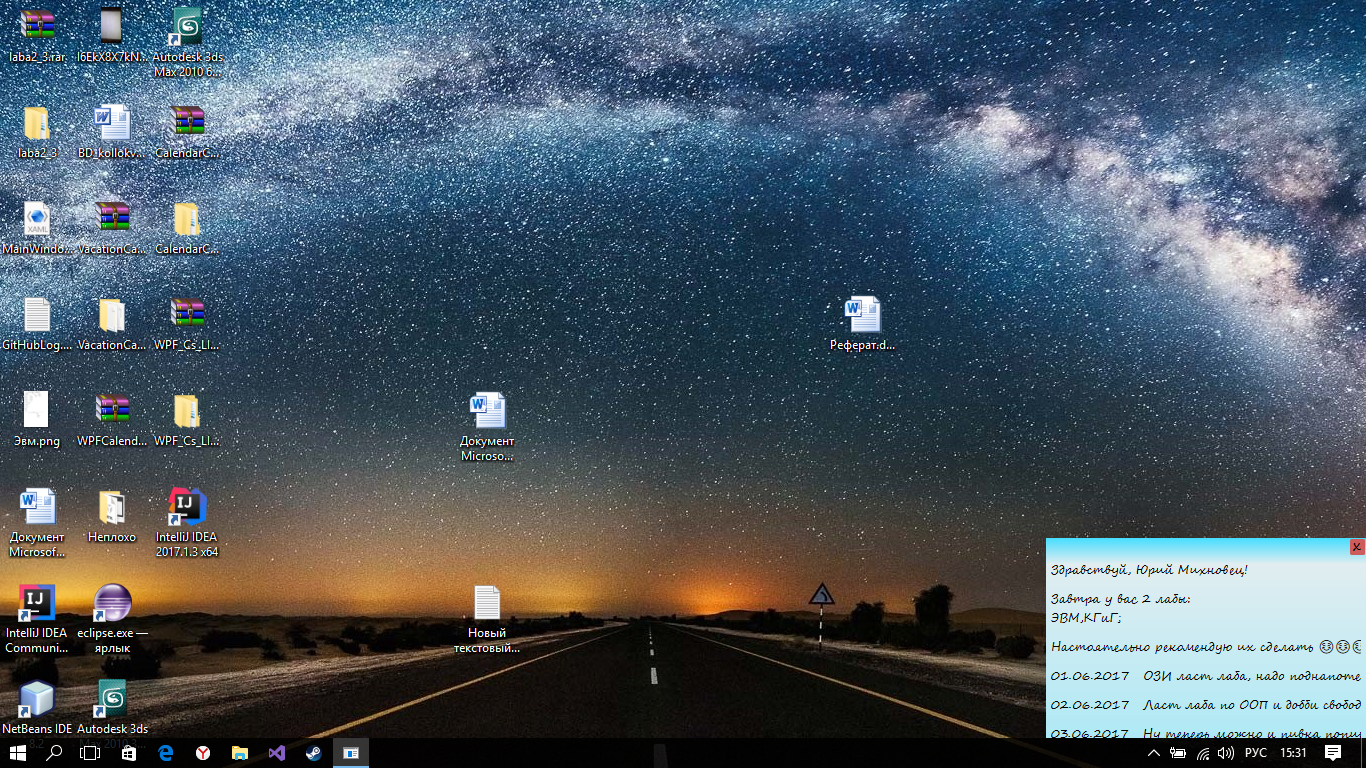


Рисунок 24 – Оповещение

# **7.** **Тестирование и отладка программы**

Объектом испытания является программа Organizer, находящаяся в папке Organizer тренажёр (листинг программы находится в приложении А).

Испытания проводятся с целью проверки правильности функционирования программы и выявления имеющихся недостатков.

Приложение работает в среде Visual Studio 2017 под платформой Windows 10.

Программу разработана на языке C#.

Для проведения испытания программы необходимо открыть исполняемый файл Organizer.ехе и проверить работу всех вариантов использования системы. Разработанное приложение может использоваться на персональном компьютере с операционной системой Windows.

Испытания проводятся по следующему алгоритму:

а) запустить на выполнение Organizer.ехе, находящийся в папке Organizer;

б) в появившейся на экран форме необходимо зарегистрироваться или при имеющейся учётной записи ввести логин и пароль;

в) в появившейся на экран форме, на вкладке «Расписание» необходимо необходимо заполнить несколько позиций;

г) перейти на вкладку «Прогресс», выбрать любой предмет и добавить его в список отслеживания;

д) изменить на любую величину количество всех заданий и выполненных. Удалить прогресс.

е) перейти на вкладку «Заметки». Создать и удалить несколько заметок.

ж) перейти на вкладку «Доска объявлений». Если учётная запись имеет ограничение доступа «Староста» — отправить сообщение.

Проведённые испытания показали, что программа работает верно и без каких-либо ошибок.

# **Заключение**

В результате выполнения данной курсовой работы было разработано программное средство «Мессенджер». Дающий возможность обмениваться сообщениями пользователям, а также просматривать пользователей данного программного средства.

Разработанное программное средство реагирует на ошибочный ввод данных.

Данное средство предоставляет перечень функций:

* Пользователь может зарегистрироваться;
* Зайти в приложение;
* Просмотреть свой профиль;
* Просмотреть имеющиеся диалоги;
* Просмотреть имеющиеся сообщения в диалогах;
* Удалить сообщение из диалога;
* Просмотреть имеющихся пользователей;
* Создать диалог с пользователем, если его нет;
* Найти пользователя;
* Добавить пользователя в друзья;

В процессе выполнения данной курсовой работы были закреплены навыки в программировании на языке C#, проектировании баз данных и реализации их в СУБД MS SQL Server 2012.

# **Список использованных источников**

1. MSDN сеть разработчиков в Microsoft [Электронный ресурс] / Э– Режим доступа: http://msdn.microsoft.com/library/rus/ -- Дата доступа:14.03.2015
2. Шилдт Г «Полный справочник по C#», 2004, 752 с
3. Эндрю Троелсон «Язык программирования C# 6.0 и платформа .NET 4.6»
4. Рихтер Дж. «CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#»

# **Приложение А**

Функции работы с формой авторизации.

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Net;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using System.Security.Cryptography;

namespace КП1

{

public partial class MainWindow : Window

{

Database DB = new Database();

public MainWindow()

{

try

{

InitializeComponent();

}

catch (Exception)

{

MessageBox.Show("Произошла ошибка.\nПовторите попытку.");

}

}

public void Registr(object sender, RoutedEventArgs e)

{

try

{

Registration RegistrationForm = new Registration();

RegistrationForm.Show();

}

catch (Exception)

{

MessageBox.Show("Произошла ошибка.\nПовторите попытку.");

}

}

public void Log\_in(object sender, RoutedEventArgs e)

{

try

{

bool Enter = false;

foreach (Log\_in L in DB.Log\_in.Where(l => l.Login == LoginEnter.Text))

{

if (Hash.PasswordsMatch(Hash.EncryptPassword(LoginEnter.Text, PasswordEnter.Password), L.Password))

{

foreach (User U in DB.Users.Where(u => u.ID == L.ID))

{

Enter = true;

Profile Profile = new Profile(U);

this.Close();

Profile.Show();

break;

}

break;

}

}

if (Enter == true)

Close();

else

MessageBox.Show("Неверный логин или пароль :(");

}

catch (Exception)

{

MessageBox.Show("Произошла ошибка.\nПовторите попытку.");

}

}

}

}

# **Приложение Б**

Функции работы с формой регистрации.

using Microsoft.Win32;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using System.Drawing;

using System.Data.SqlClient;

using System.Data;

using System.Drawing.Imaging;

using System.Text.RegularExpressions;

using System.Security.Cryptography;

namespace КП1

{

public partial class Registration : Window

{

Database DB = new Database();

public Registration()

{

InitializeComponent();

}

public void Registr(object sender, RoutedEventArgs e)

{

try

{

User U = new User();

Us(U);

Close();

}

catch(Exception)

{

MessageBox.Show("Произошла ошибка.\nПовторите попытку.");

}

}

private static byte[] ConvertImageToByteArray(string fileNameS)

{

//try

//{

string fileName = "";

Regex r = new Regex(@"file:///");

if (r.IsMatch(fileNameS))

{

int i = fileNameS.Length;

for (int j = 8; j < i; j++)

fileName += fileNameS[j];

}

else fileName = fileNameS;

Bitmap bitMap = new Bitmap(fileName);

ImageFormat bmpFormat = bitMap.RawFormat;

var imageToConvert = System.Drawing.Image.FromFile(fileName);

using (MemoryStream ms = new MemoryStream())

{

imageToConvert.Save(ms, bmpFormat);

return ms.ToArray();

}

//}

//catch (Exception)

//{

// MessageBox.Show("Произошла ошибка.\nПовторите попытку.");

//}

}

void Us(User Prof)

{

try

{

Prof.Name = regName.Text;

Prof.LName = regLName.Text;

Prof.Email = regEmail.Text;

Prof.DR = Convert.ToDateTime(regDR.Text);

if (regPhoto.Source == null)

{

Prof.Photo = ConvertImageToByteArray("..\\..\\Улыбка.png");

}

else

{

Prof.Photo = ConvertImageToByteArray(Convert.ToString(regPhoto.Source));

}

if (regM.IsChecked == true)

Prof.Gender = false;

else

Prof.Gender = true;

Prof.Log\_in.Add(new Log\_in

{

Login = regLogin.Text,

Password = Hash.EncryptPassword(regLogin.Text, regPassword.Password),

ID = Prof.ID

});

DB.Users.Add(Prof);

DB.SaveChanges();

}

catch (Exception)

{

MessageBox.Show("Произошла ошибка.\nПовторите попытку.");

}

}

private void open\_ButonClick(object sender, RoutedEventArgs e)

{

try

{

string Path;

OpenFileDialog myDialog = new OpenFileDialog();

myDialog.Filter = "Картинки(\*.JPG;\*.GIF)|\*.JPG;\*.GIF" + "|Все файлы (\*.\*)|\*.\* ";

myDialog.CheckFileExists = true;

myDialog.Multiselect = true;

if (myDialog.ShowDialog() == true)

{

Path = myDialog.FileName;

regPhoto.Source = new BitmapImage(new Uri(Path));

}

}

catch (Exception)

{

MessageBox.Show("Произошла ошибка.\nПовторите попытку.");

}

}

private void Close(object sender, RoutedEventArgs e)

{

try

{

Close();

}

catch (Exception)

{

MessageBox.Show("Произошла ошибка.\nПовторите попытку.");

}

}

}

}

# **Приложение В**

Функции работы с формой профиля.

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using System.Drawing;

using System.Windows.Controls;

using System.Data.Entity;

using System.Threading;

using System.Security.Cryptography;

namespace КП1

{

public partial class Profile : Window

{

Database DB = new Database();

User Us;

int i = 0;

public Profile()

{

InitializeComponent();

}

public Profile(User U)

{

try

{

InitializeComponent();

if (DB.Messages.Where(m => m.SenderID == U.ID).Count() != 0)

NameDialog.Loaded += NameDialog\_Loaded;

Us = U;

xName.Content = U.Name + " " + U.LName;

xEmail.Content = U.Email;

xDR.Content = U.DR.Day.ToString() + "." + U.DR.Month.ToString() + "." + U.DR.Year.ToString();

if (U.Photo != null)

{

MemoryStream imageS = new MemoryStream(U.Photo);

System.Drawing.Image returnImagge = System.Drawing.Image.FromStream(imageS);

string path = Directory.GetCurrentDirectory() + U.ID.ToString();

returnImagge.Save(path);

Photo.Source = new BitmapImage(new Uri(path));

}

else

Photo.Source = new BitmapImage(new Uri("улыбка.png"));

DB.Users.Where(u => u.ID != U.ID).Load();

qwe.ItemsSource = DB.Users.Local;

NameDialog.SelectionChanged += NameDialog\_SelectionChanged;

Message.Loaded += Message\_Loaded;

Prof.Loaded += Prof\_Loaded;

}

catch (Exception)

{

MessageBox.Show("Упс. Что-то пошло не так.");

}

}

private void Prof\_Loaded(object sender, RoutedEventArgs e)

{

try

{

NameDialog.SelectedItem = null;

}

catch (Exception)

{

MessageBox.Show("Произошла ошибка.\nПовторите попытку.");

}

}

private void Message\_Loaded(object sender, RoutedEventArgs e)

{

}

private void Timer\_Tick(object sender, EventArgs e)

{

try

{

LoadMessageList();

}

catch (Exception)

{

MessageBox.Show("Произошла ошибка.\nПовторите попытку.");

}

}

private void NameDialog\_Loaded(object sender, RoutedEventArgs e)

{

try

{

DB.Dialogs.Where(d => d.Sender\_ID == Us.ID && d.Receiver\_ID != Us.ID).Load();

NameDialog.ItemsSource = DB.Dialogs.Local;

if (NameDialog.SelectedItem == NameDialog.Items[i])

{

LoadMessageList();

i++;

}

}

catch (Exception)

{

MessageBox.Show("Произошла ошибка.\nПовторите попытку.");

}

}

private void NameDialog\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

try

{

if (NameDialog.SelectedItem != null)

{

LoadMessageList();

System.Windows.Threading.DispatcherTimer timer = new System.Windows.Threading.DispatcherTimer();

timer.Tick += Timer\_Tick;

timer.Interval = new TimeSpan(50000000);

timer.Start();

}

}

catch (Exception)

{

MessageBox.Show("Произошла ошибка.\nПовторите попытку.");

}

}

////

private void TextBox\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

try

{

qwe.SelectionChanged += Qwe\_SelectionChanged;

}

catch (Exception)

{

MessageBox.Show("Произошла ошибка.\nПовторите попытку.");

}

}

////

private void Dialog(User Receiver)

{

try

{

Dialog D1 = new Dialog { Receiver\_ID = Receiver.ID, Sender\_ID = Us.ID, Receiver\_Name = Receiver.Name, Sender\_Name = Us.Name };

Dialog D2 = new Dialog { Sender\_ID = Receiver.ID, Receiver\_ID = Us.ID, Receiver\_Name = Us.Name, Sender\_Name = Receiver.Name };

DB.Dialogs.Add(D1); DB.Dialogs.Add(D2);

DB.SaveChanges();

DB.Dispose();

DB = new Database();

DB.Dialogs.Where(d => d.Sender\_ID == Us.ID && d.Receiver\_ID != Us.ID).Load();

NameDialog.ItemsSource = DB.Dialogs.Local;

OpenMessages(Receiver.ID);

}

catch (Exception)

{

MessageBox.Show("Произошла ошибка.\nПовторите попытку.");

}

}

private void Qwe\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

try

{

DB.Dispose();

DB = new Database();

User U = qwe.SelectedItem as User;

User Receiver = qwe.SelectedItem as User;

if (DB.Dialogs.Count() == 0)

{

Dialog(Receiver);

}

else

{

int rec\_ID = (qwe.SelectedItem as User).ID;

if (DB.Dialogs.Where(d => d.Sender\_ID == Us.ID && d.Receiver\_ID == rec\_ID).Count() == 0)

{

Dialog(Receiver);

}

else

{

NameDialog.SelectedItem = null;

MainTb.SelectedItem = Message;

List<Dialog> List = DB.Dialogs.Where(d => d.Receiver\_ID == U.ID && d.Sender\_ID == Us.ID).ToList();

foreach (Dialog d in List)

{

NameDialog.SelectedItem = d;

}

}

}

qwe.SelectionChanged -= Qwe\_SelectionChanged;

}

catch (Exception)

{

MessageBox.Show("Произошла ошибка.\nПовторите попытку.");

}

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

try

{

DB.Dispose();

DB = new Database();

DB.Users.Load();

qwe.ItemsSource = DB.Users.Local;

}

catch (Exception)

{

MessageBox.Show("Произошла ошибка.\nПовторите попытку.");

}

}

private void OpenMessages(int id\_receiver)

{

try

{

MainTb.SelectedItem = Message;

foreach (Dialog d in NameDialog.Items)

{

if (d.Receiver\_ID == id\_receiver)

{

NameDialog.SelectedItem = d;

Dialog dd = new Dialog();

dd = NameDialog.SelectedItem as Dialog;

DB.Messages.Where(m => m.Dialog.Receiver\_ID == dd.Receiver\_ID && m.Dialog.Sender\_ID == dd.Sender\_ID && m.Dialog.Receiver\_ID != dd.Sender\_ID && m.Dialog.Sender\_ID != dd.Receiver\_ID).Load();

\_messagesList.ItemsSource = DB.Messages.Local;

NameDialog.SelectedItem = d;

}

}

}

catch (Exception)

{

MessageBox.Show("Произошла ошибка.\nПовторите попытку.");

}

}

private void Send\_Message(object sender, RoutedEventArgs e)

{

try

{

if (NameDialog.SelectedItem == null)

{

MessageBox.Show("Выберите диалог!");

}

else

{

if (MyMessage.Text == null)

MessageBox.Show("Введите сообщение!");

else

{

DB.Dispose();

DB = new Database();

Message M = new Message

{

SenderID = (NameDialog.SelectedItem as Dialog).Sender\_ID,

ReceiverID = (NameDialog.SelectedItem as Dialog).Receiver\_ID,

Message1 = MyMessage.Text

};

DB.Messages.Add(M);

DB.SaveChanges();

MyMessage.Clear();

}

}

}

catch (Exception)

{

MessageBox.Show("Произошла ошибка.\nПовторите попытку.");

}

}

private void LoadMessageList()

{

try

{

DB.Dispose();

DB = new Database();

if (NameDialog.SelectedItem != null)

{

Dialog dd = (NameDialog.SelectedItem as Dialog);

DB.Messages.Where(m => (m.ReceiverID == dd.Receiver\_ID || m.ReceiverID == dd.Sender\_ID) && (m.SenderID == dd.Sender\_ID || m.SenderID == dd.Receiver\_ID)).OrderByDescending(o => o.ID).Load();

\_messagesList.ItemsSource = DB.Messages.Local;

}

}

catch (Exception)

{

MessageBox.Show("Произошла ошибка.\nПовторите попытку.");

}

}

private void DeleteMessage(object sender, MouseButtonEventArgs e)

{

\_messagesList.SelectionChanged += \_messagesList\_SelectionChanged;

}

private void \_messagesList\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

try

{

Message M = \_messagesList.SelectedItem as Message;

if (\_messagesList.SelectedItem != null)

{

DB.Messages.Remove(M);

DB.SaveChanges();

}

}

catch (Exception)

{

MessageBox.Show("Произошла ошибка.\nПовторите попытку.");

}

\_messagesList.SelectionChanged -= \_messagesList\_SelectionChanged;

}

}

}

# **Приложение Г**

Класс Hash для хеширования пароля.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Security.Cryptography;

using System.Text;

using System.Threading.Tasks;

namespace КП1

{

static class Hash

{

public static byte[] EncryptPassword(string userName, string password)

{

//пароль+ соль из логина

string tmpPassword = userName.ToLower() + password;

//Строка пароля преобразуется в массив байтов.

UTF8Encoding textConverter = new UTF8Encoding();

byte[] passBytes = textConverter.GetBytes(tmpPassword);

//Возвращаются зашифрованные байты

return new SHA384Managed().ComputeHash(passBytes);

}

// Сравниваются два массива на равенство

// Можно добавить сравнение длины, но обычно

// размер всех хешей одинаков.

public static bool PasswordsMatch(byte[] psswd1, byte[] psswd2)

{

try

{

for (int i = 0; i < psswd1.Length; i++)

{

if (psswd1[i] != psswd2[i])

return false;

}

return true;

}

catch (IndexOutOfRangeException)

{

return false;

}

}

}

}