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 «Мессенджер»

Исполнитель

|  |  |  |
| --- | --- | --- |
| cтудентка 2 курса группы 2    Руководитель |  | \_\_\_\_\_\_\_\_\_ А.И. Загнетова  подпись, дата |
| ассистент |  | \_\_\_\_\_\_\_\_\_ А.Д. Новицкая  подпись, дата |
| Курсовой проект защищен с оценкой |  | \_\_\_\_\_\_\_\_\_ |
| Руководитель \_\_\_\_\_\_\_ |  | А.Д. Новицкая |

подпись

Минск 2017

Оглавление

[Введение 3](#_Toc483535249)

[1. Постановка задачи 4](#_Toc483535250)

[1.1. Алгоритмы решения 4](#_Toc483535251)

[1.2. Обзор прототипов 5](#_Toc483535252)

[1.3. Актуальность задачи 7](#_Toc483535253)

[2. Разработка архитектуры проекта 8](#_Toc483535254)

[2.1. Обобщенная структура 8](#_Toc483535255)

[2.2. Диаграмма UML 8](#_Toc483535256)

[3. Разработка функциональной модели данных ПС 10](#_Toc483535257)

[3.1. Выполняемые функции 10](#_Toc483535258)

[4. Руководство пользователя 14](#_Toc483535259)

[Заключение 20](#_Toc483535260)

[Список использованных источников 21](#_Toc483535261)

[Приложение А 22](#_Toc483535262)

[Приложение Б 24](#_Toc483535263)

[Приложение В 27](#_Toc483535264)

[Приложение Г 33](#_Toc483535265)

# **Введение**

В настоящее время, в связи с большим развитием интернета, большинство людей привыкли общаться и решать различные дела не при личной встрече, а с помощью переписки в интернете. Это намного упрощает жизнь, ведь бывают много случаев, когда люди живут друг от друга на большом расстоянии и нет возможности встретиться. Также решаются и всяческие деловые вопросы, нет нужды менять рабочий график ради личной встречи, если можно написать и уточнить все, что интересует.

Благодаря этому, появилась потребность в приложениях, с помощью которых можно было бы общаться. В настоящее время пользователей уже не устроит просто отправить и получить сообщение. Нужно, чтобы приложение имело удобный интерфейс, привлекающий дизайн, а также большой функционал.

Мой мессенджер позволяет переписываться со всеми пользователями, которые зарегистрировались в приложении. Можно просмотреть всех пользователей, заполнить краткую информацию о себе. Приложение имеет простой и удобный интерфейс и спокойный дизайн.

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 мессенджера.

Основной функционал приложения:

* Авторизация;
* Хранение личной информации в базе данных;
* Обмен сообщениями;
* Список пользователей;

## **1.1. Алгоритмы решения**

Основная задача при разработке заключалась в нахождении способа передачи сообщений. Пользователь должен авторизоваться, указав свой логин и пароль, далее он получает доступ к своему профилю. Если пользователь еще не регистрировался, то можно пройти регистрацию, заполнив немного данных о себе. А если он вошел, он может просмотреть список пользователей, которые уже зарегистрировались, написать кому-нибудь из них сообщения, также он может просмотреть уже ранее созданный диалог. Сообщения сохраняются и при следующей авторизации они будут доступны. Сообщения можно удалять.

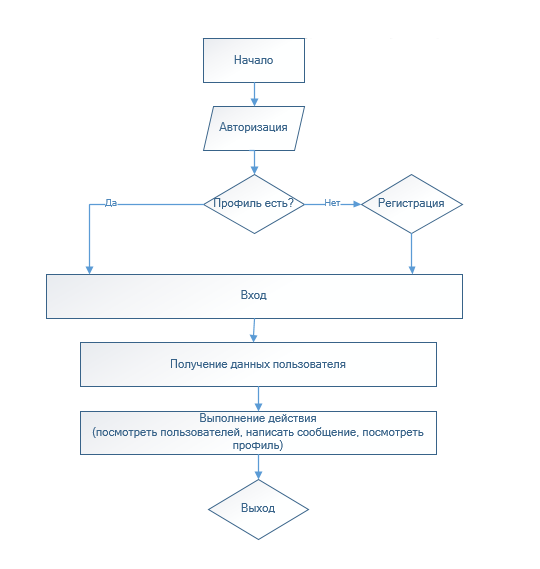


Рисунок 1 – Алгоритм работы с программой

Важным являлось получение сообщения без длительных задержек, то есть, чтобы обновление происходило автоматически и пользователю не приходилось бы это делать.

Для решения данной задачи я использовала таймер, который обновляет данные в базе данных.

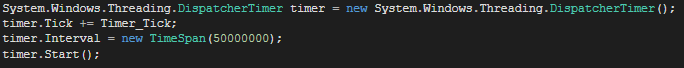


Рисунок 2 – Код таймера

## **1.2. Обзор прототипов**

Сейчас существует очень много мессенджеров. Одним из наиболее популярных является Viber.

Viber – приложение для мгновенного обмена сообщениями и голосовыми звонками. Его преимущество в том, что он является кроссплатформенным. Viber автоматически синхронизируется со списком контактов в телефоне, определяя, у кого еще из твоих контактов установлено приложение.

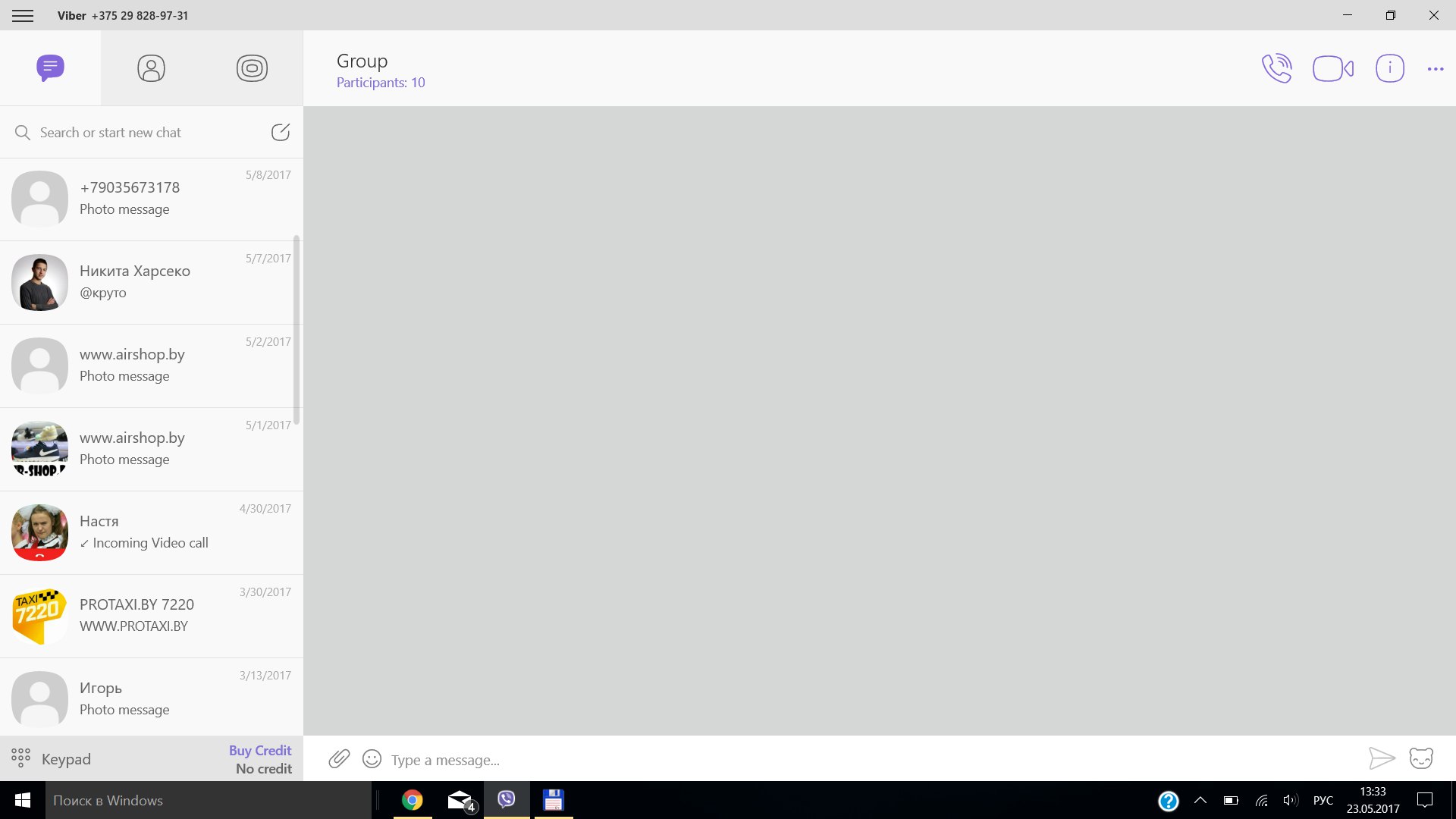


Рисунок 3 – Диалог в Viber

Telegram – сервис по обмену мгновенными сообщениями. С его помощью можно отправлять сообщения, фотографии, изображения, видеозаписи, а также текущую геопозицию. Главное преимущество – открытость протоколов, шифрование всего и вся.

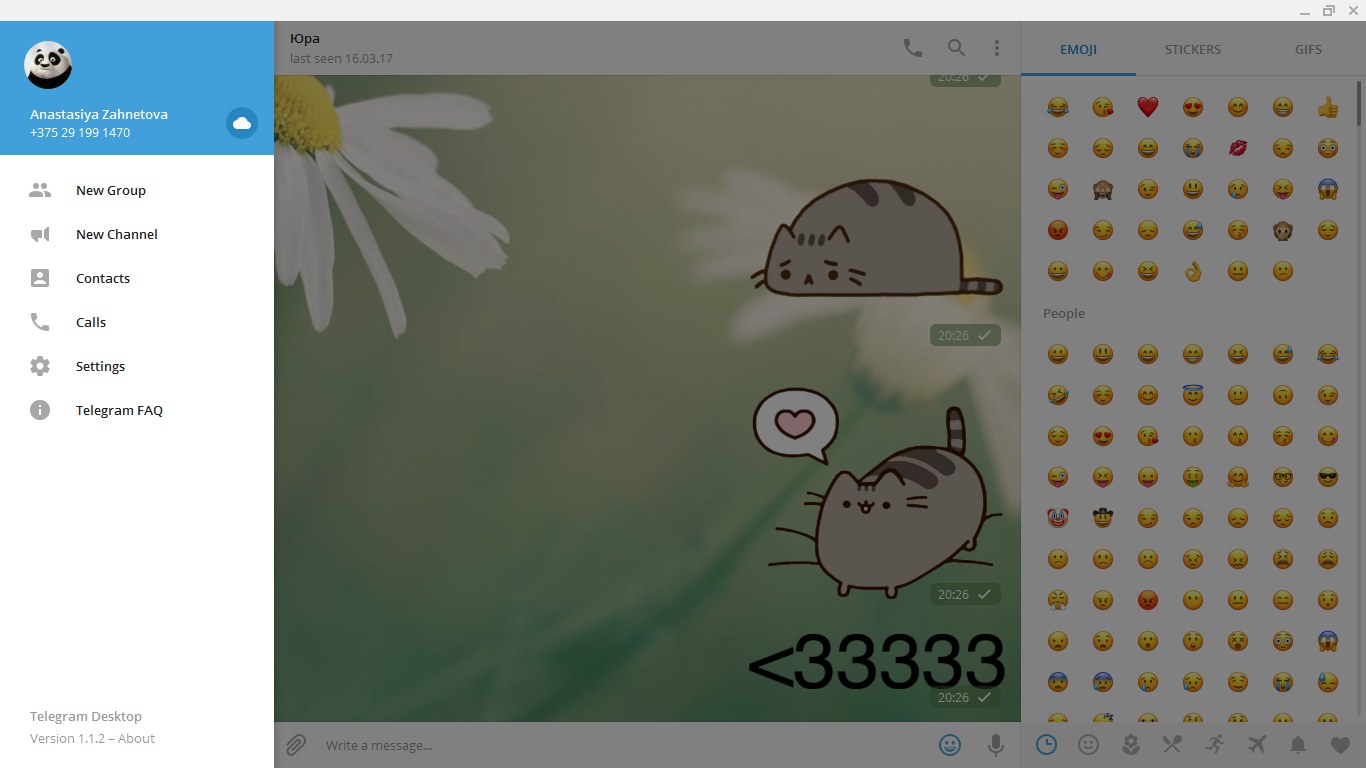


Рисунок 4 – Диалог в Telegram

WhatsApp – удобное кросс-платформенное приложение, с помощью которого можно обмениваться сообщениями. Можно создавать группы с друзьями, обмениваться видео, фото и аудиофайлами.

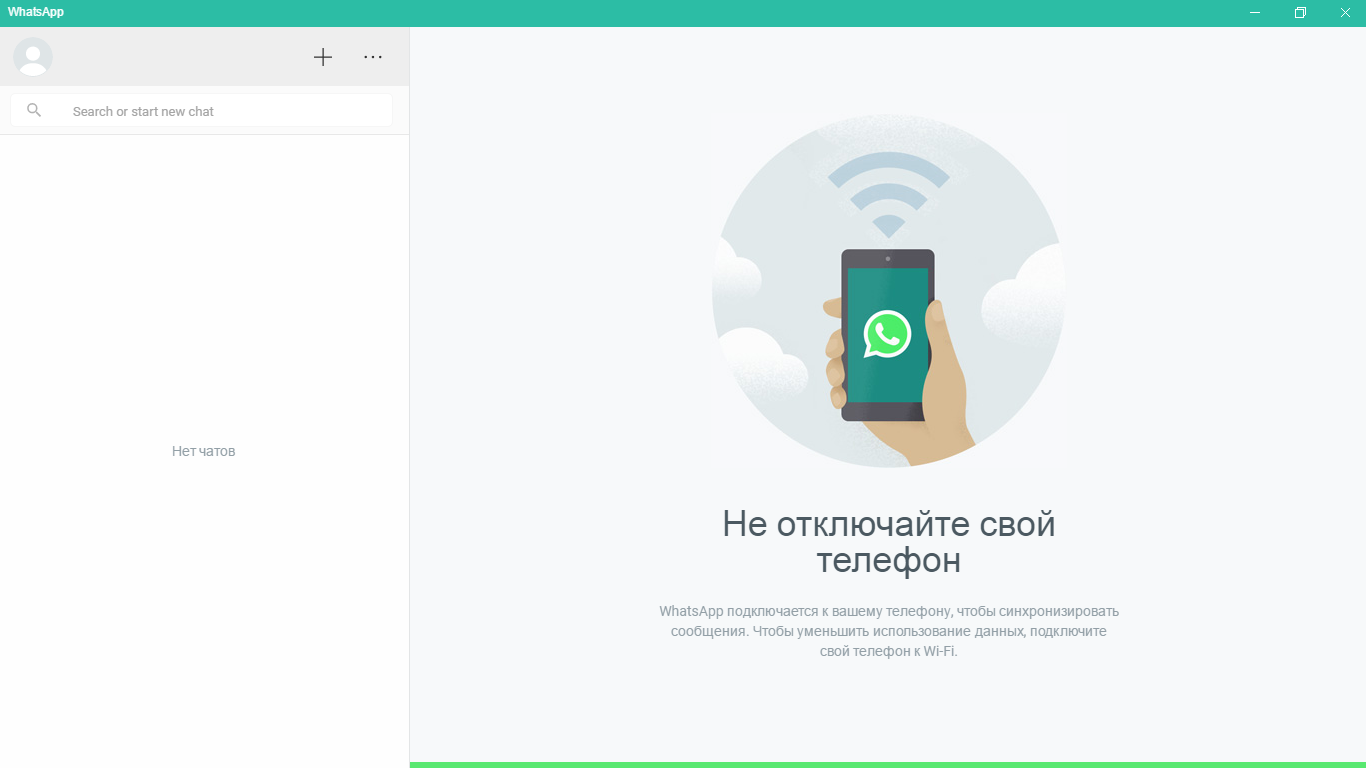


Рисунок 5 – WhatsApp

## **1.3. Актуальность задачи**

На сегодняшний день очень много людей пользуются интернетом, поэтому большинство из них нуждаются в простом и удобном программном средстве коммуникации.

Существует много аналогов мессенджеров, главной задачей которых является связь между людьми. Они имеют большой функционал: отправка сообщений, документов, фотографий, звонки, видеосвязь и т.д. На современном рынке выигрывает то приложение, которые может дать пользователю больше возможностей.

Немаловажным является безопасность и защита личных данных, но не все приложения могут это гарантировать.

Поэтому при выборе темы на курсовую работу я остановилась на мессенджере, так как это является очень актуальным.

Таким образом, разрабатываемое приложение позволит обмениваться сообщениями, просматривать зарегистрированных пользователей, искать пользователей по имени и фамилии, добавлять в друзья и просматривать личную информацию, указанную при регистрации. Авторизация будет осуществляться с помощью логина и пароля, где пароль будет храниться в базе данных в зашифрованном виде.

1. **Разработка архитектуры проекта**

## **2.1. Обобщенная структура**

Структура проекта представлена на рисунке 6.

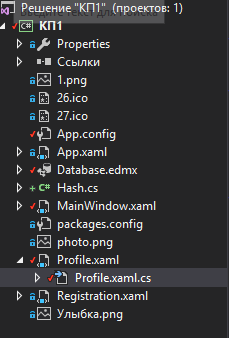


Рисунок 6 – Структура проекта

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. |
| Database.edmx | Файл, в котором представлены классы для работы с таблицами базы данных. |
| Hash.cs | Класс, содержащий методы для хеширования пароля. |
| MainWindow.xaml | Форма для авторизации. |
| Profile.xaml | Форма личного профиля. |
| Registration.xaml | Фора для регистрации. |

## **2.2. Диаграмма UML**

Диаграмма классов представлена на рисунке 7.

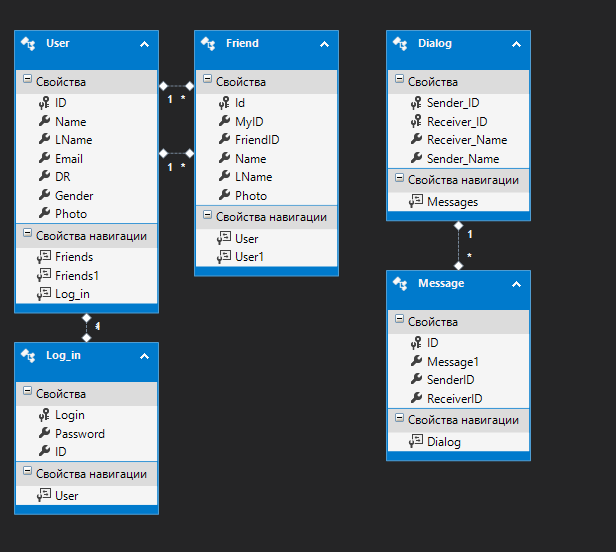


Рисунок 7 – Диаграмма классов

Самым главным классов является класс User, он содержит личные данные пользователя, такие, как идентификатор (первичный ключ), имя, фамилию, Е-мэйл, дату рождения, пол и фотографию. От него зависит класс Log\_in по идентификатору (вторичный ключ), в котором содержатся логин и пароль пользователя. Также от класса User зависит класс Friend по идентификаторам пользователя и добавленного друга.

В класс Dialog содержатся имя отправителя и получателя и их идентификаторы (первичные ключи). От него зависит класс Message по идентификатору отправителя и получателя (вторичные ключи), в котором хранятся идентификаторы отправителя и получателя и сообщение, которое было отправлено.

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

## **3.1. Выполняемые функции**

Рассмотрим реализацию нескольких функций для работы с базой данных.

**public void Log\_in** (object sender, RoutedEventArgs e) – функция для авторизации пользователя, проверяющая, есть ли в базе данных такие логин и пароль, которые ввел пользователь. В кэш из таблицы Log\_in загружаются данные, если введенный логин совпадает с логином в базе данных. Далее хешируется введенный пароль и проверяется и захешированным паролем совпадающим с данным логином. Если все совпало, то открывается форма профиля, а если таких данных нет, то выводит сообщение о неверных данных.

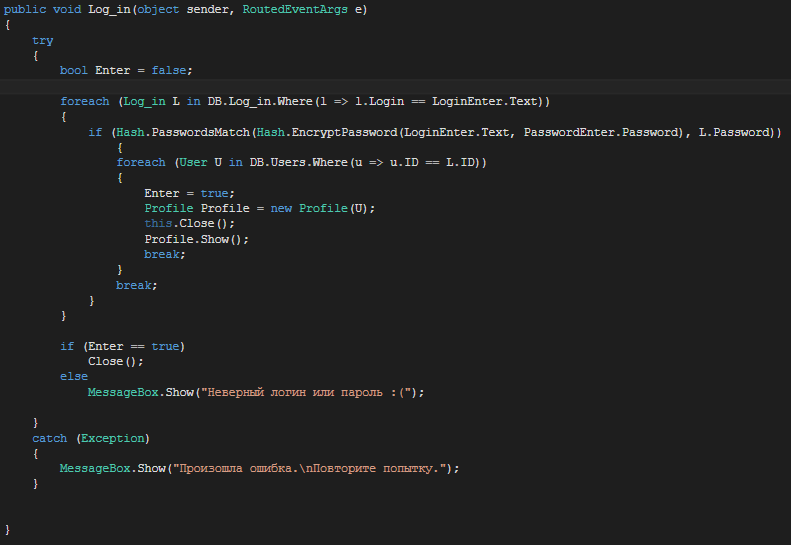


Рисунок 8 – Функция авторизации

Конструктор **public Profile** (User U) – в данной функции происходит загрузка данных о пользователе на вкладку «Профиль», вызываются функции для загрузки существующих диалогов у этого пользователя и сообщений в эти диалоги на вкладку «Диалоги», а также загружаются все пользователи на вкладку «Пользователи».

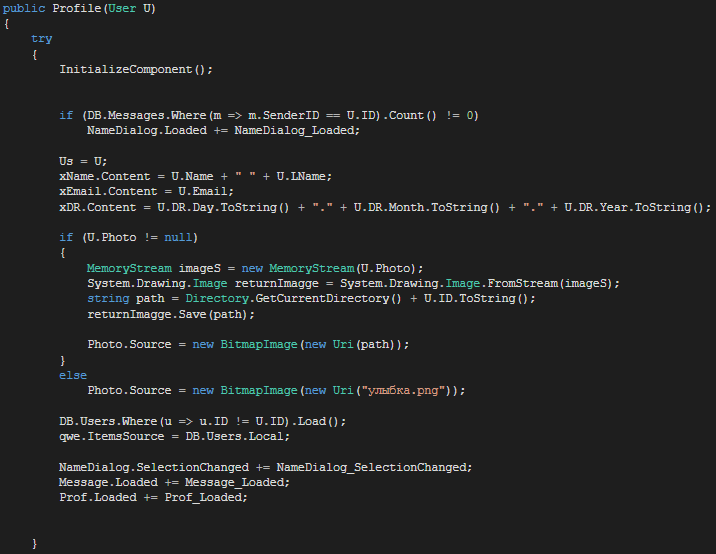


Рисунок 9 – Функция загрузки профиля.

**3.2. Модель базы данных**

Я создала базу данных в среде Microsoft SQL Server. Загрузила её на хостинг SmarterASP.NET для того, чтобы для разных пользователей была одна база данных, и при добавлении в нее данных одним пользователем, они бы появлялись в приложении другого пользователя.

В ней имеются 5 таблиц: User, Log\_in, Dialogs, Messages и Friend. Они представлены ниже вместе со своими структурами.

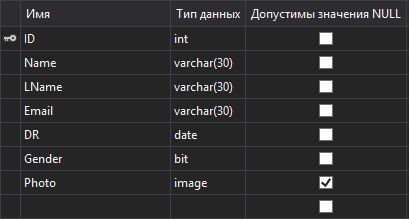


Рисунок 10 – Таблица User

Таблица 2 – структура таблицы User

|  |  |
| --- | --- |
| Название столбца | Содержание |
| ID | Идентификатор пользователя |
| Name | Имя пользователя |
| LName | Фамилия пользователя |
| Email | Е-мейл пользователя |
| DR | Дата рождения пользователя |
| Gender | Пол пользователя |
| Photo | Фотография пользователя |

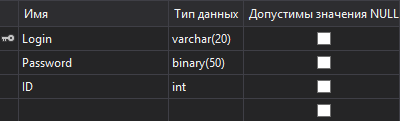


Рисунок 11 – Таблица Message

Таблица 3 – содержание таблицы Log\_in

|  |  |
| --- | --- |
| Название столбца | Содержание |
| ID | Идентификатор пользователя |
| Password | Пароль пользователя |
| Login | Логин пользовтеля |

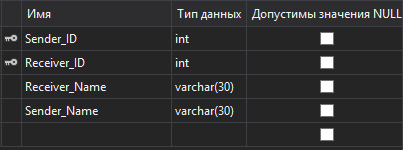


Рисунок 12 – Таблица Dialogs

Таблица 4 – структура таблицы Dialogs

|  |  |
| --- | --- |
| Название столбца | Содержание |
| Sender\_ID | Идентификатор отправителя |
| Receiver\_ID | Идентификатор получателя |
| Receiver\_Name | Имя получателя |
| Sender\_Name | Имя отправителя |

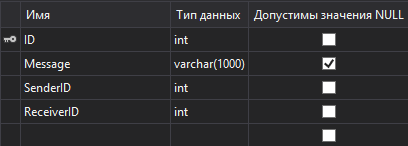


Рисунок 13 – Таблица Messages

Таблица 5 – содержание таблицы Messages

|  |  |
| --- | --- |
| Название столбца | Содержание |
| ID | Идентификатор сообщения |
| Message | Сообщение |
| SenderID | Идентификатор отправителя |
| ReceiverID | Идентификатор получателя |

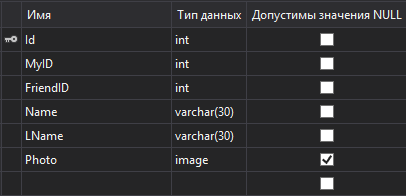


Рисунок 14 – Таблица Friend

Таблица 6 – содержание таблицы Friend

|  |  |
| --- | --- |
| Название столбца | Содержание |
| Id | Идентификатор таблицы |
| MyID | Идентификатор пользователя |
| FriendID | Идентификатор друга |
| Name | Имя друга |
| LName | Фамилия друга |
| Photo | Фотография друга |

# **4. Руководство пользователя**

При запуске приложения пользователь видит окно авторизации.

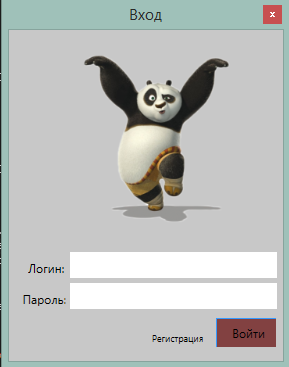


Рисунок 14 – Окно авторизации

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

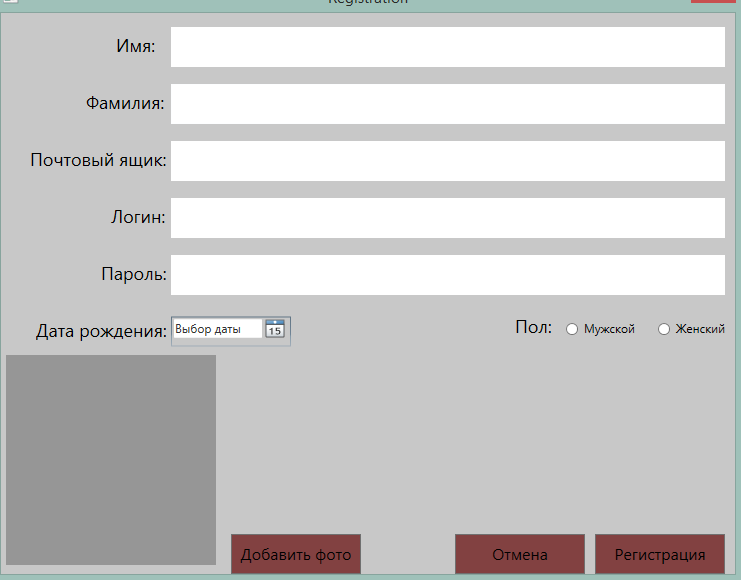


Рисунок 15 – Окно регистрации

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

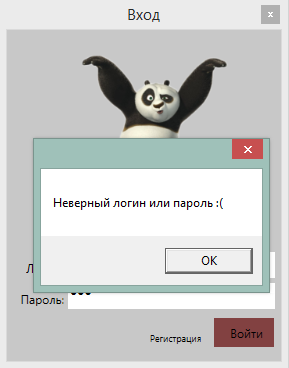


Рисунок 16 – Сообщение о неправильно введенных данных

Если данные введены правильно, открывается окно профиля пользователя.

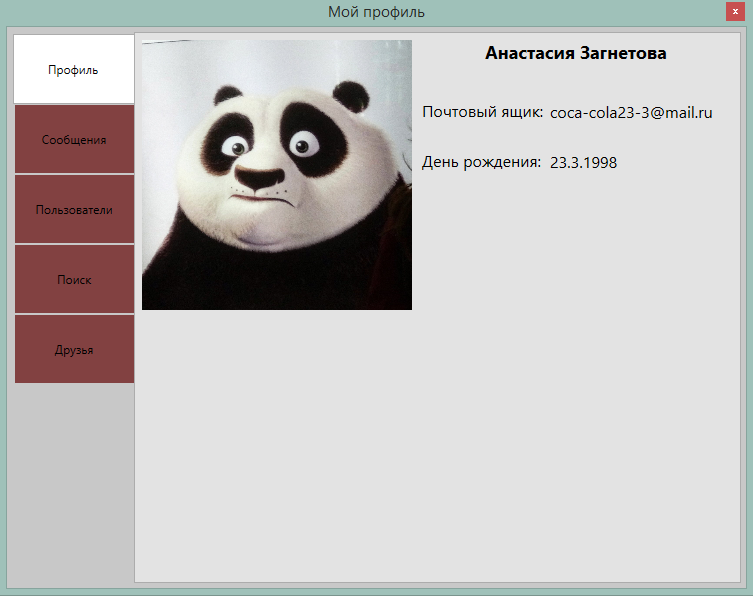


Рисунок 17 – Окно профиля пользователя

Теперь пользователь может посмотреть диалоги и сообщение, которые у него уже есть, и пользователей.

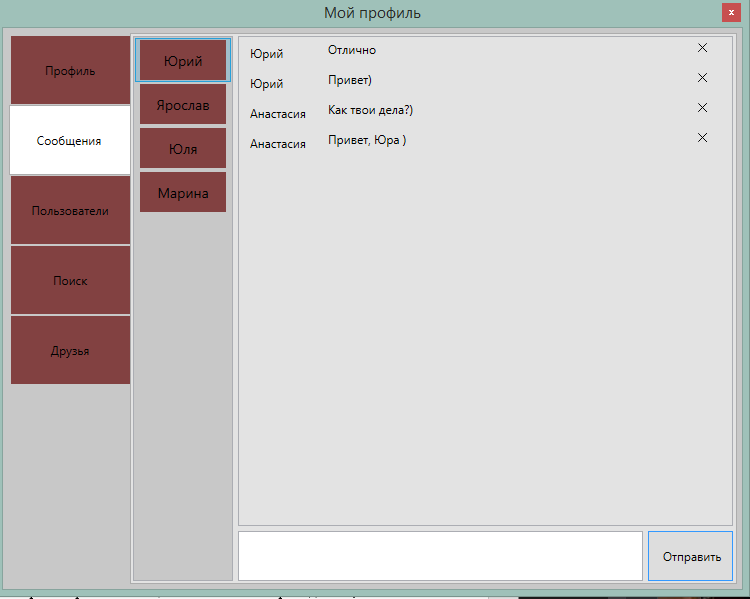


Рисунок 18 – Диалоги пользователя

Для того, чтобы отправить сообщение, пользователю достаточно ввести его в окно для сообщения и нажать кнопку отправить или enter.

По нажатию на крестик напротив сообщения, выбранное сообщение можно удалить.

Если при отправке сообщения не был выбран диалог, то появится предупреждающее окно.

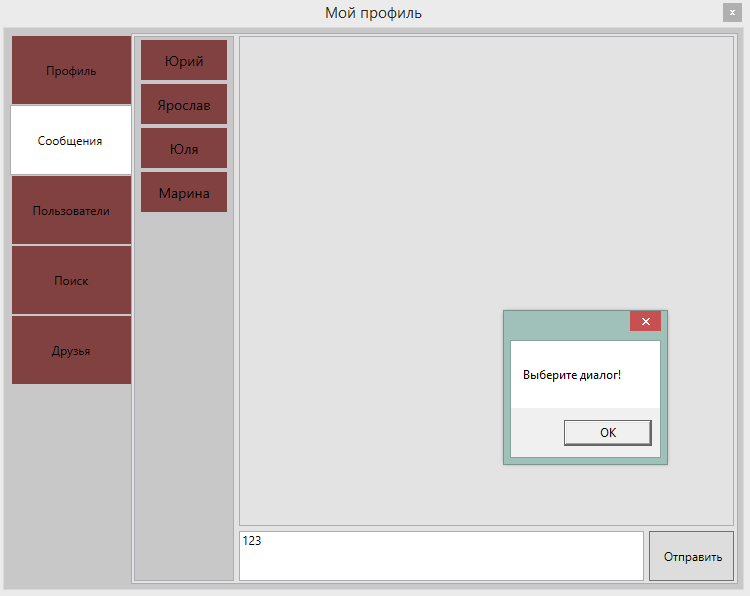


Рисунок 20 – Ошибка, при отправке сообщения, но не выбранном диалоге

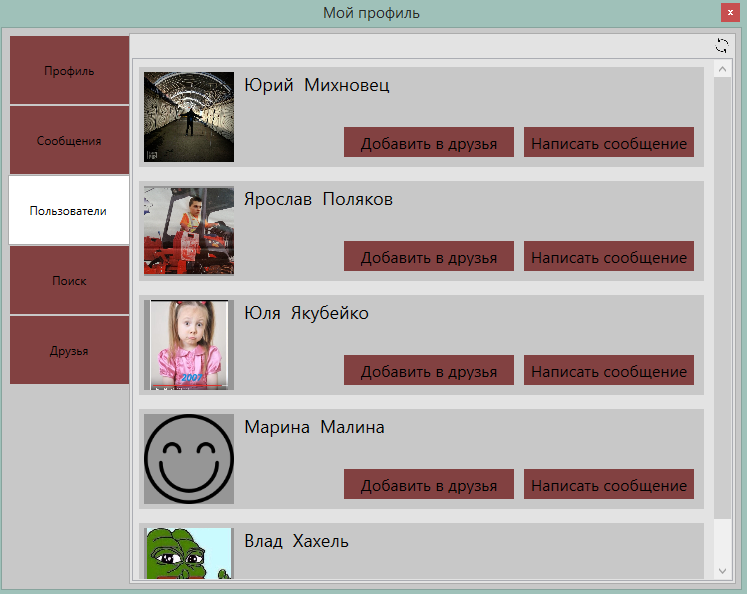


Рисунок 19 – Список пользователей

В списке пользователей можно нажать кнопку «Написать сообщение», если диалог с пользователем уже есть, то он откроется и загрузятся все сообщения, если его нет, то создастся диалог с ним. Также можно добавить выбранного пользователя в друзья.

На вкладке «Поиск» можно найти пользователя либо по имени, либо по фамилии, либо по имени и фамилии.

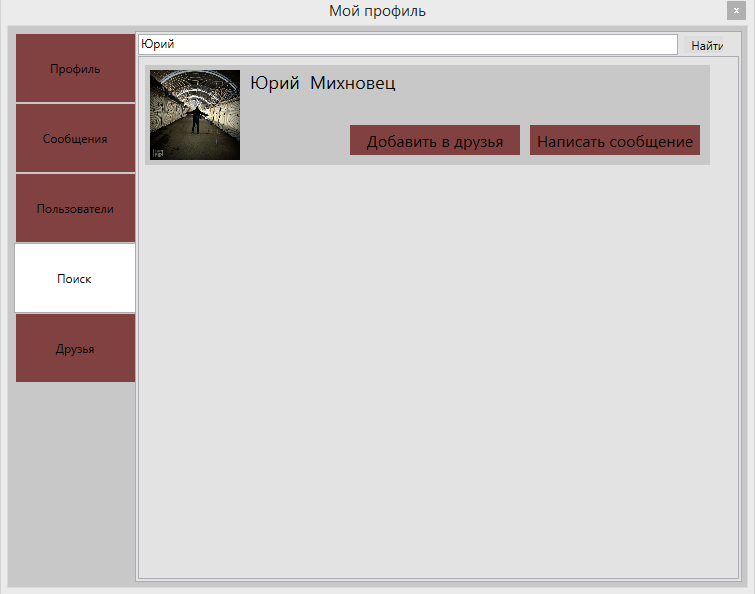


Рисунок 20 – Поиск пользователей

А на вкладке «Друзья» можно просмотреть пользователей, которых добавили в друзья.

При добавлении пользователя в друзья, на вкладе друзья нужно обновить список с помощью стрелки в правом верхнем углу, для того, чтобы пользователь появился в списке.

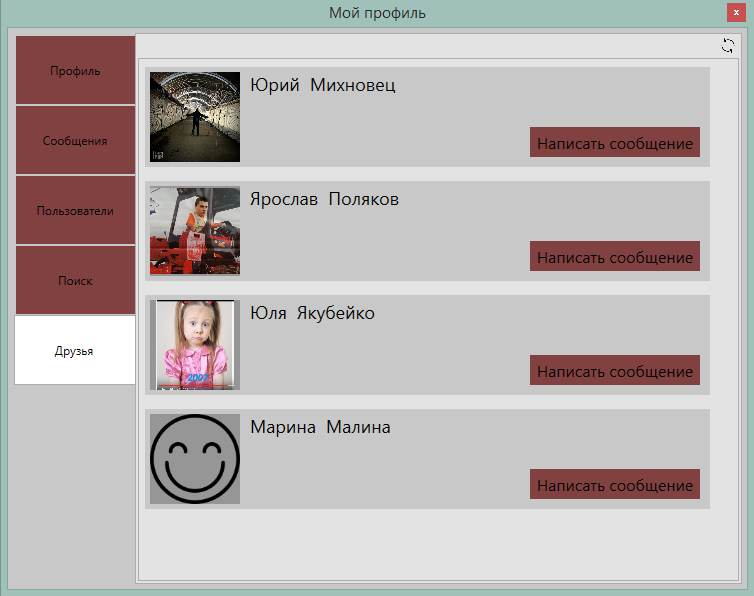


Рисунок 21 - Друзья

# **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;

}

}

}

}