

## 1. Создание проекта

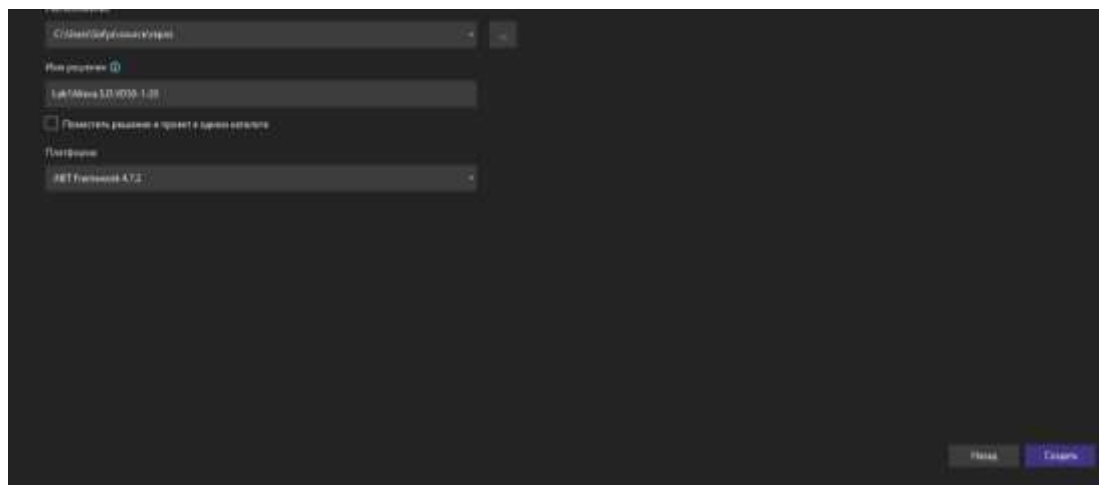


Рисунок 1 - Создание проекта

## 2. Созданный проект

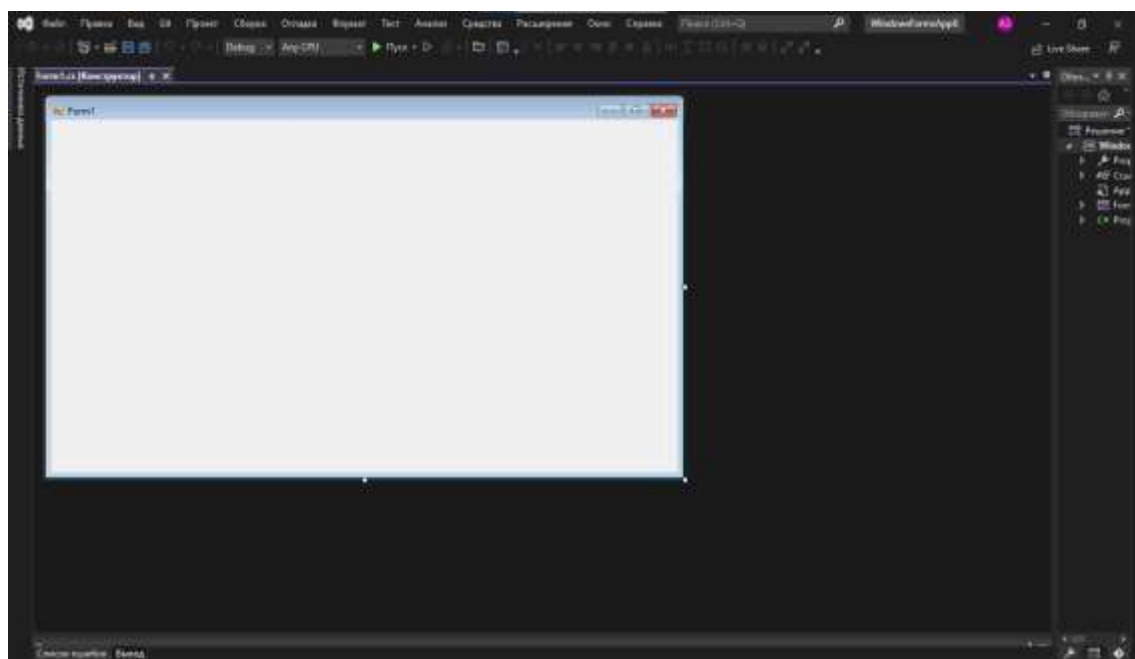


Рисунок 2 - Созданный проект

### 3. Удаление Form1

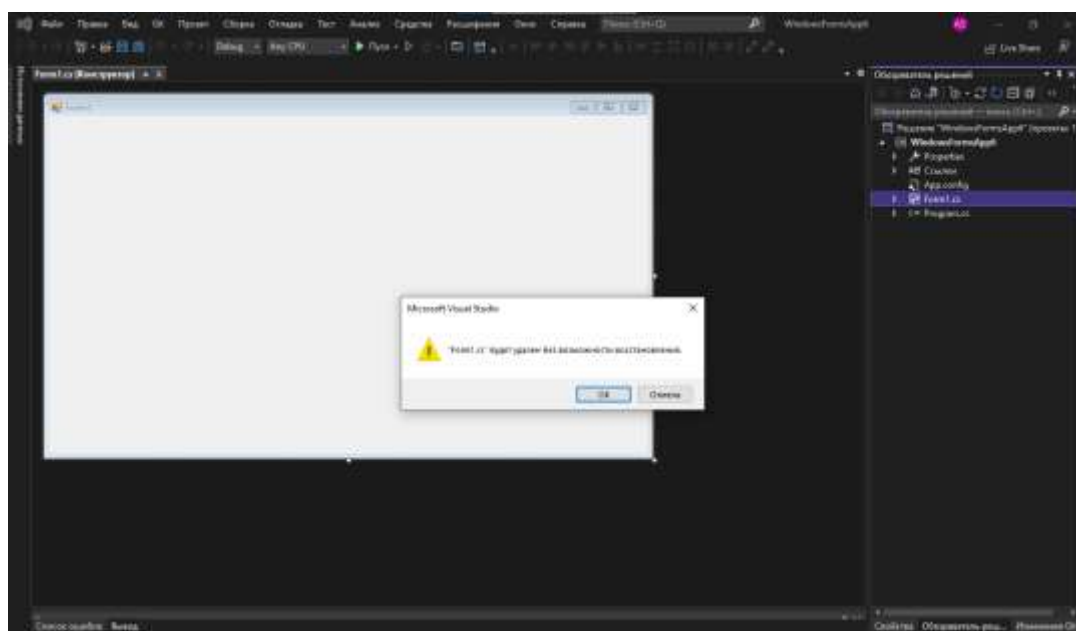


Рисунок 3 - Удаление формы

### 4. Создание LoginForm

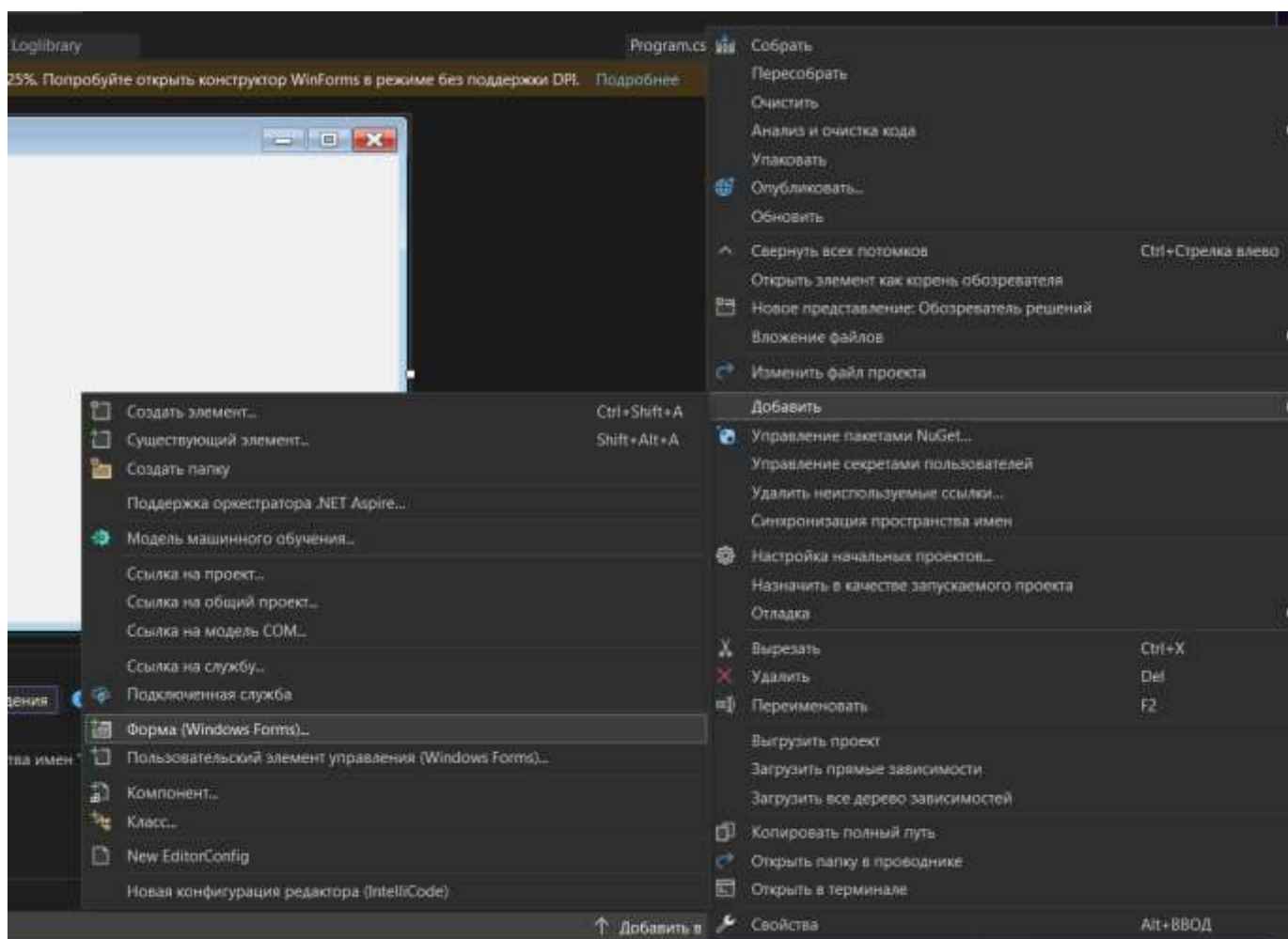


Рисунок 4 - Создание формы

## 5. Открываем панель элементов

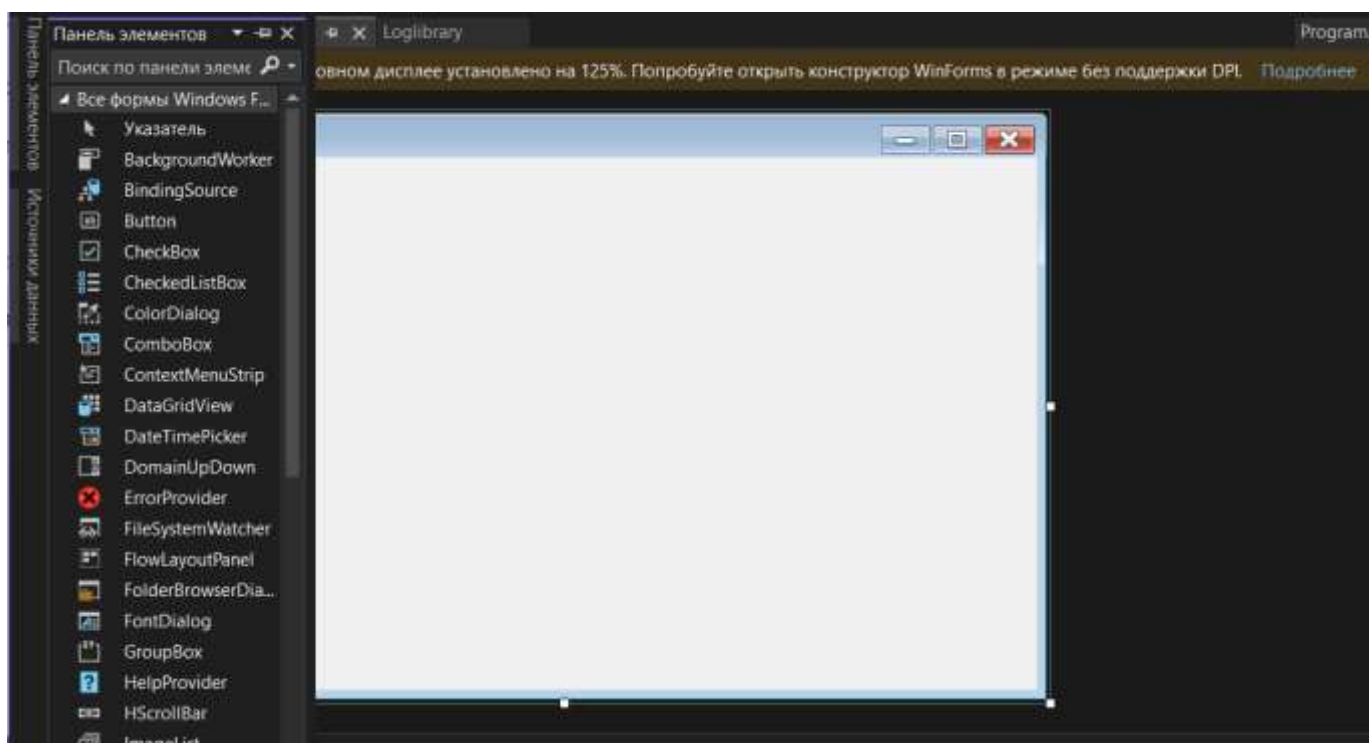


Рисунок 5 - Панель элементов

## 6. Выбираем в панели элементов элемент panel

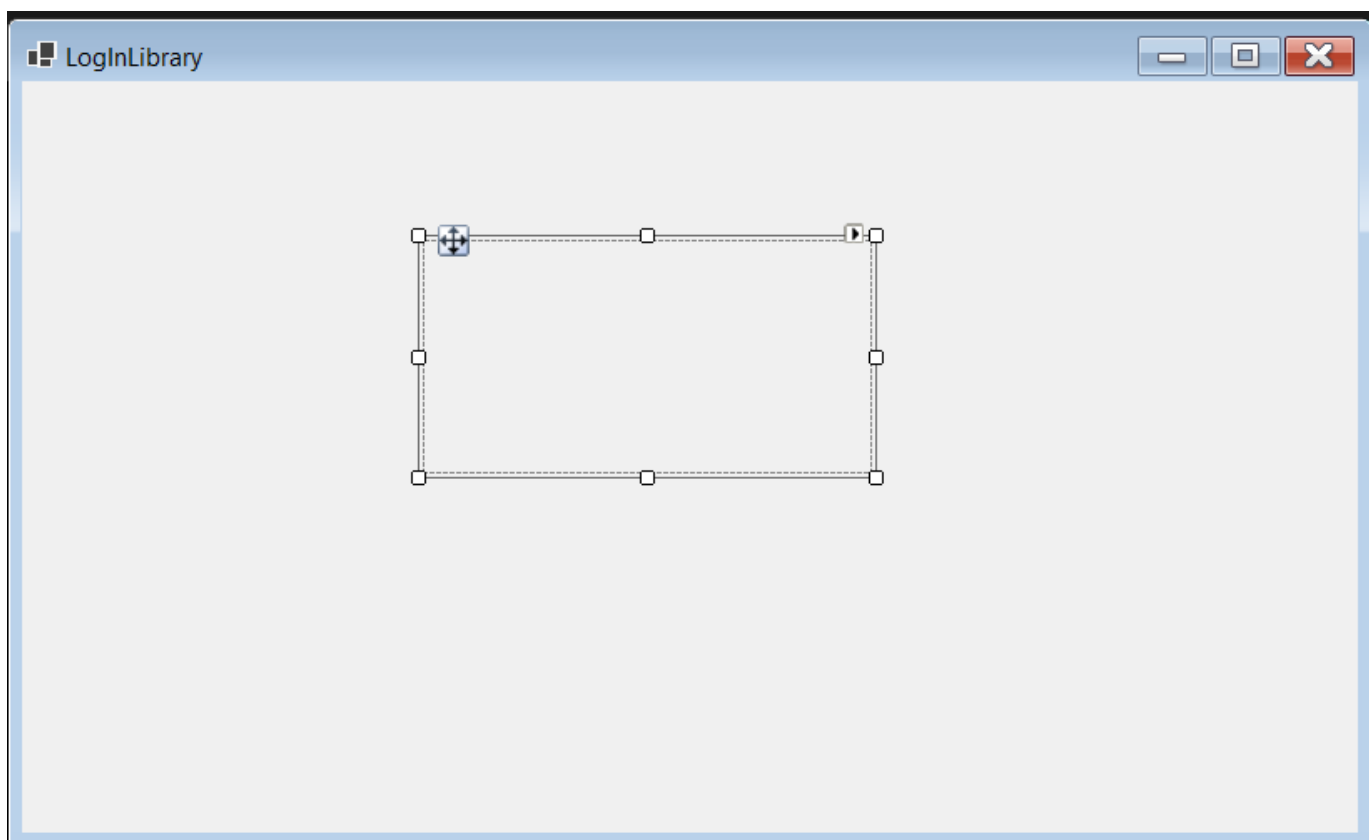


Рисунок 6 - Элемент panel

## 7. Закрепляем panel в родительском контейнере

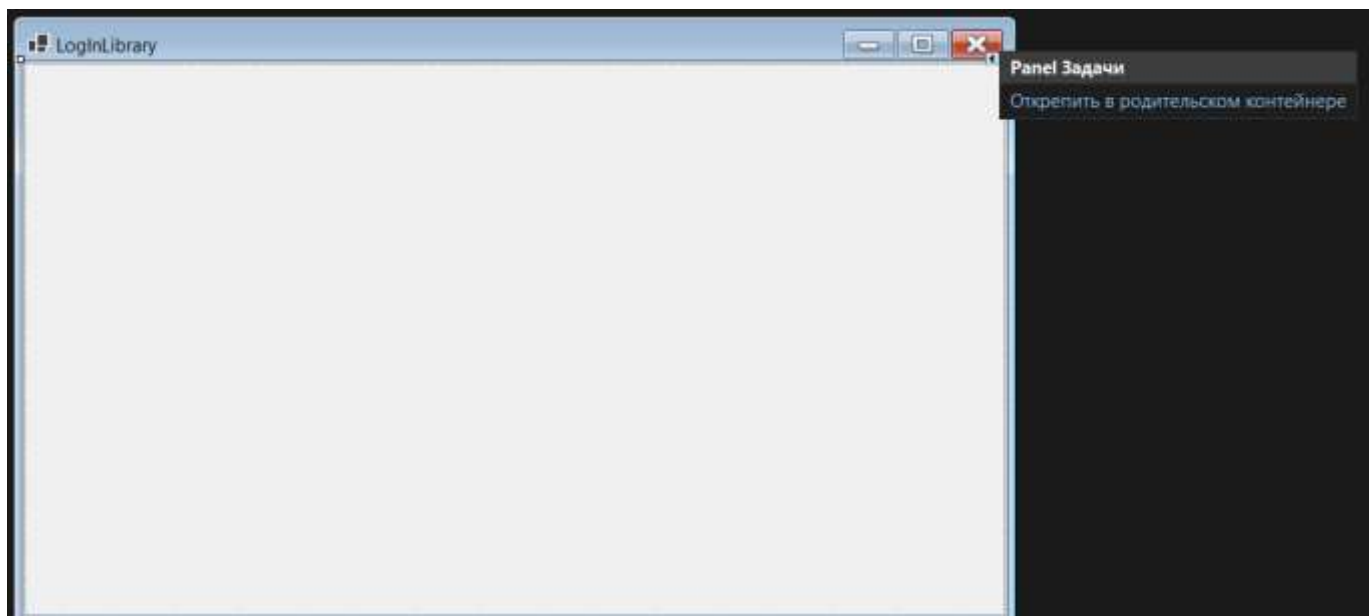


Рисунок 7 – Закрепление panel

## 8. Меняем значения свойства BackColor элемента Panel на 17;24;34

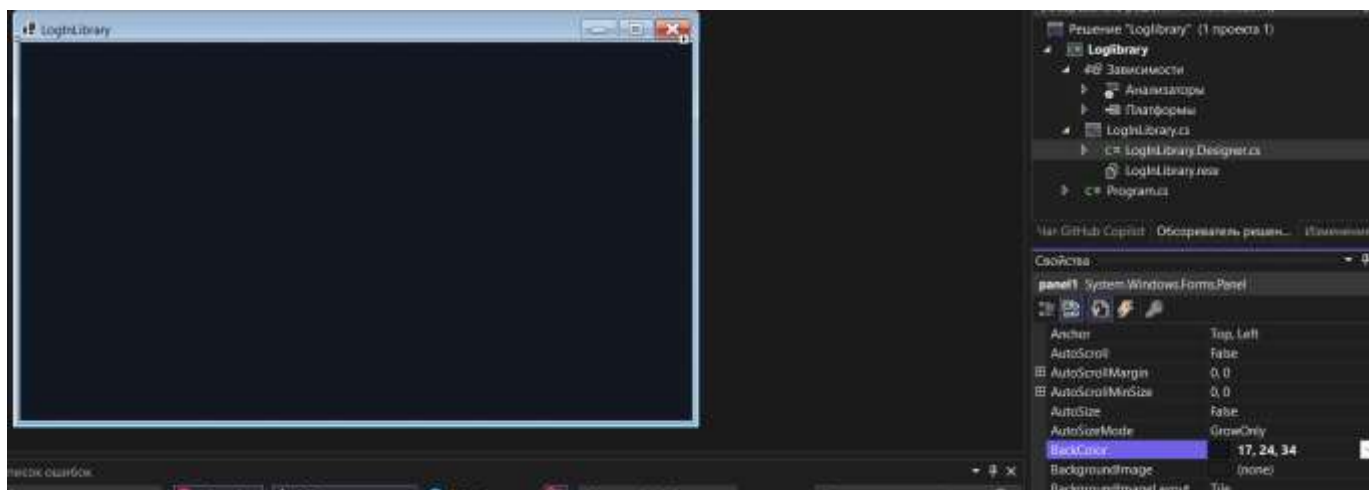


Рисунок 8 - Цвет панели

## 9. Создаем новую панель и располагаем ее сверху формы по ширине

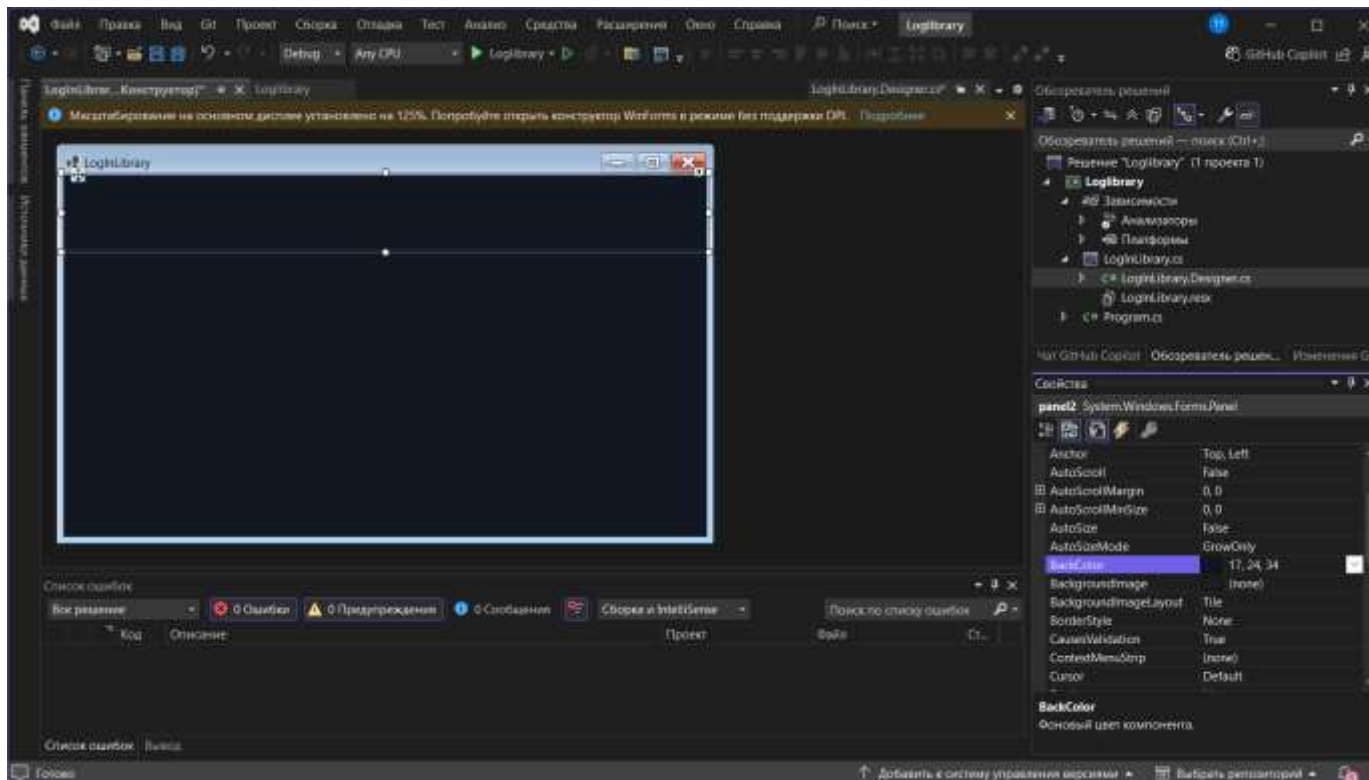


Рисунок 9 - Создание второй панели

## 10. Меняем значения свойства BackColor элемента Panel на 255;183;19

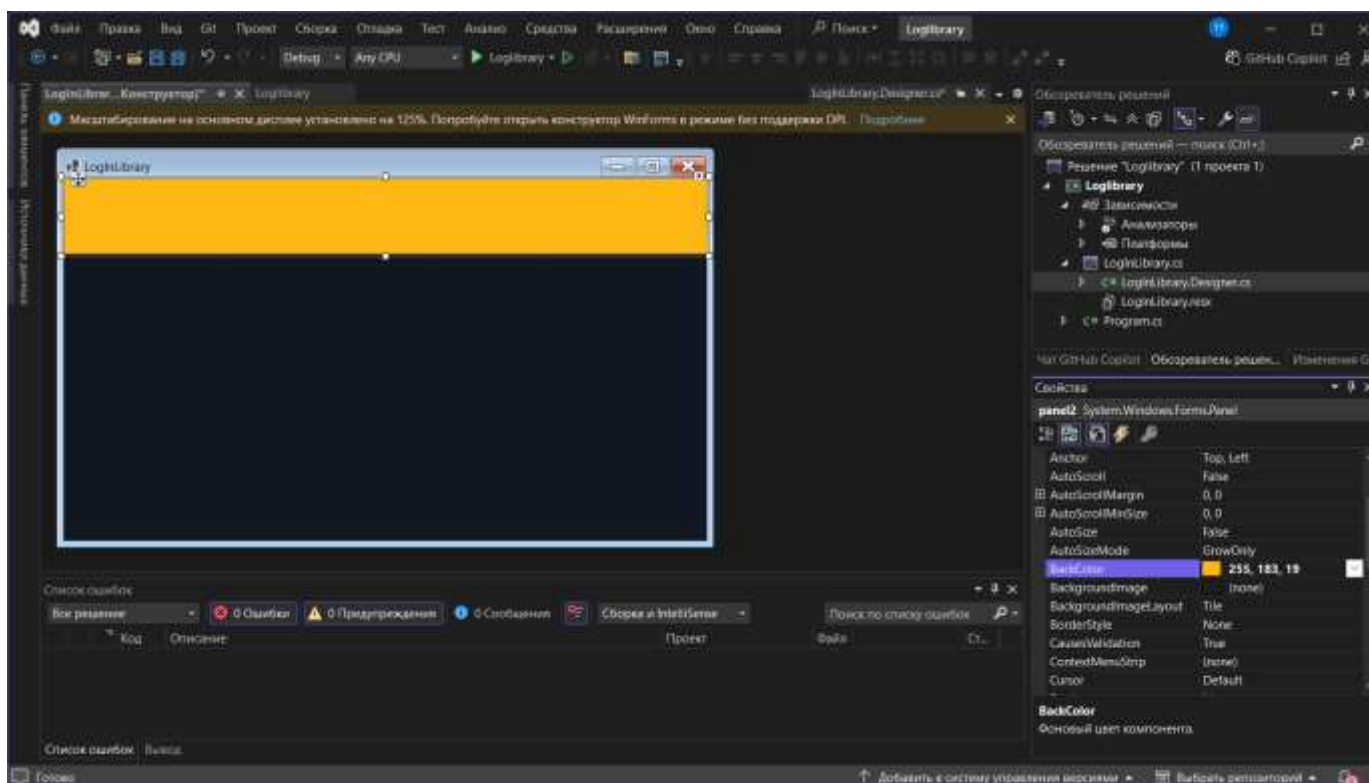


Рисунок 10 - Цвет панели

11. Создаем папку с изображениями под названием image, копируем изображения и вставляем в эту папку.

Рисунок 11 - Создание папки

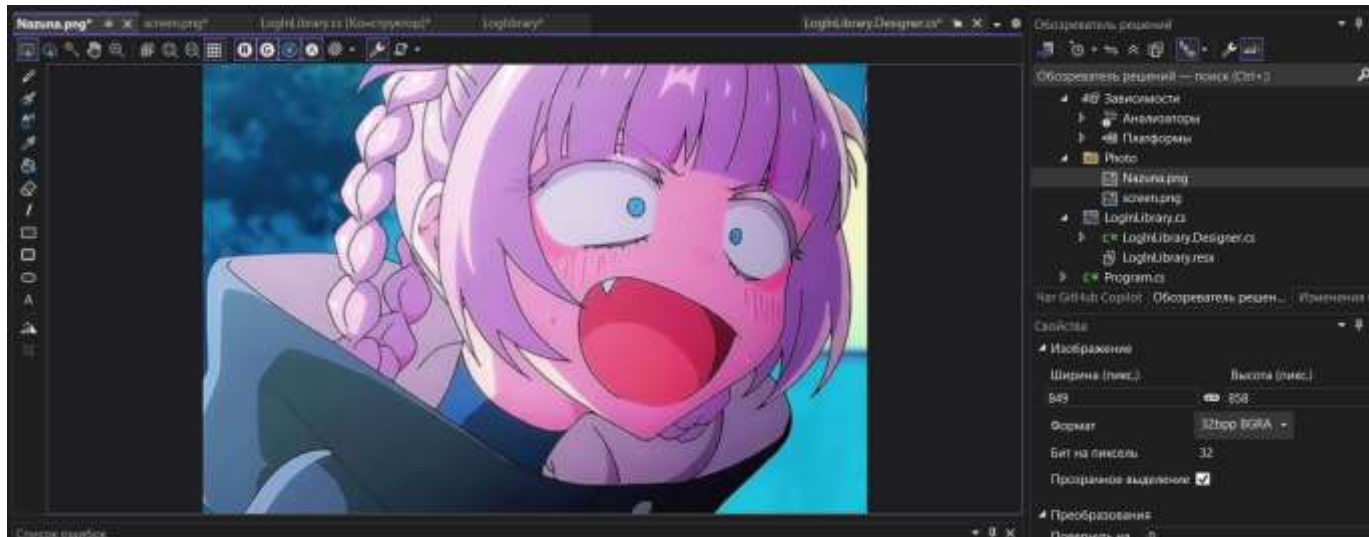


Рисунок 12 - Созданная папка

12. Создаем label и настраиваем шрифт и текст

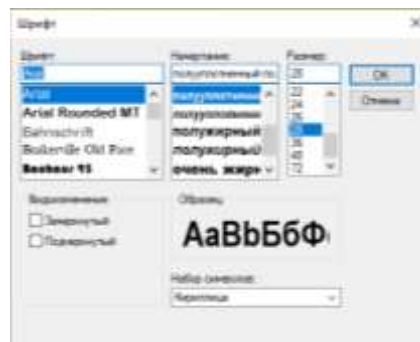


Рисунок 13 - Настройки шрифта

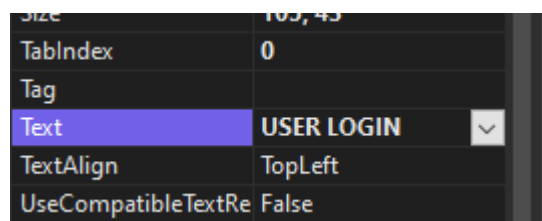


Рисунок 14 - Текст label

13. Меняем свойство dock на значение fill, и textAlign на MiddleCenter

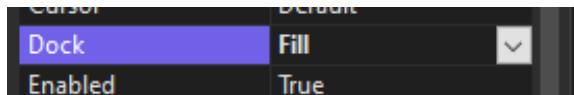


Рисунок 15 - Свойство Dock

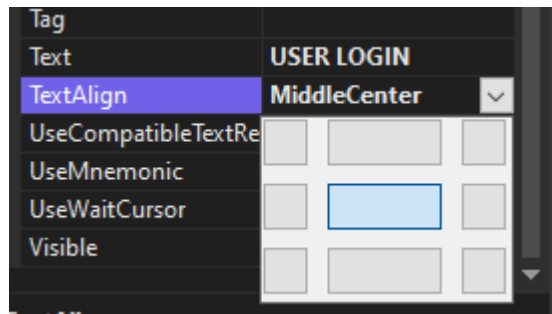


Рисунок 16 - Свойство TextAlign

14. Создаем еще один label, который будет кнопкой закрытия приложения, меняем текст на x и имя на labelClose, а так же настраиваем шрифт

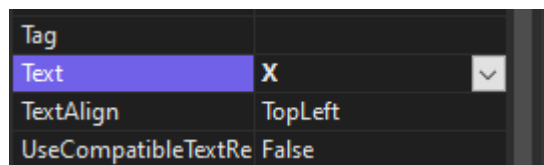


Рисунок 17 - Текст lable

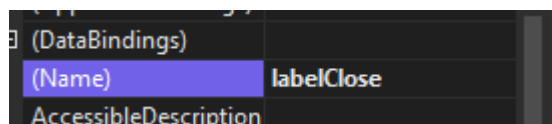


Рисунок 18 - Имя label

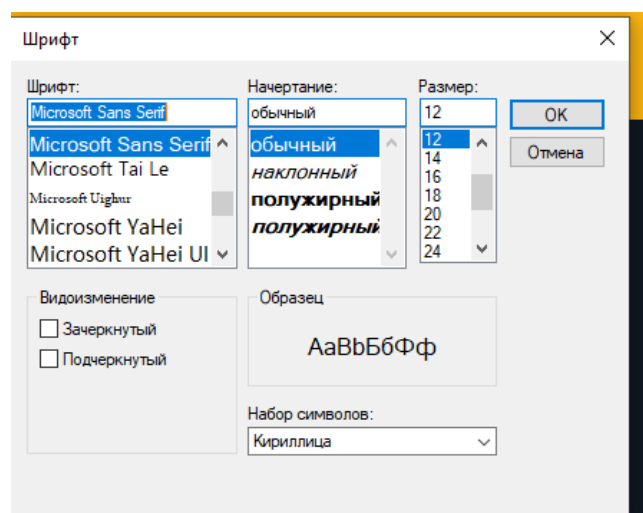


Рисунок 19 - Настройки шрифта

15. Заходим в Program.cs и меняем Form1 на LoginForm, чтобы при включении приложения загружалась LoginForm

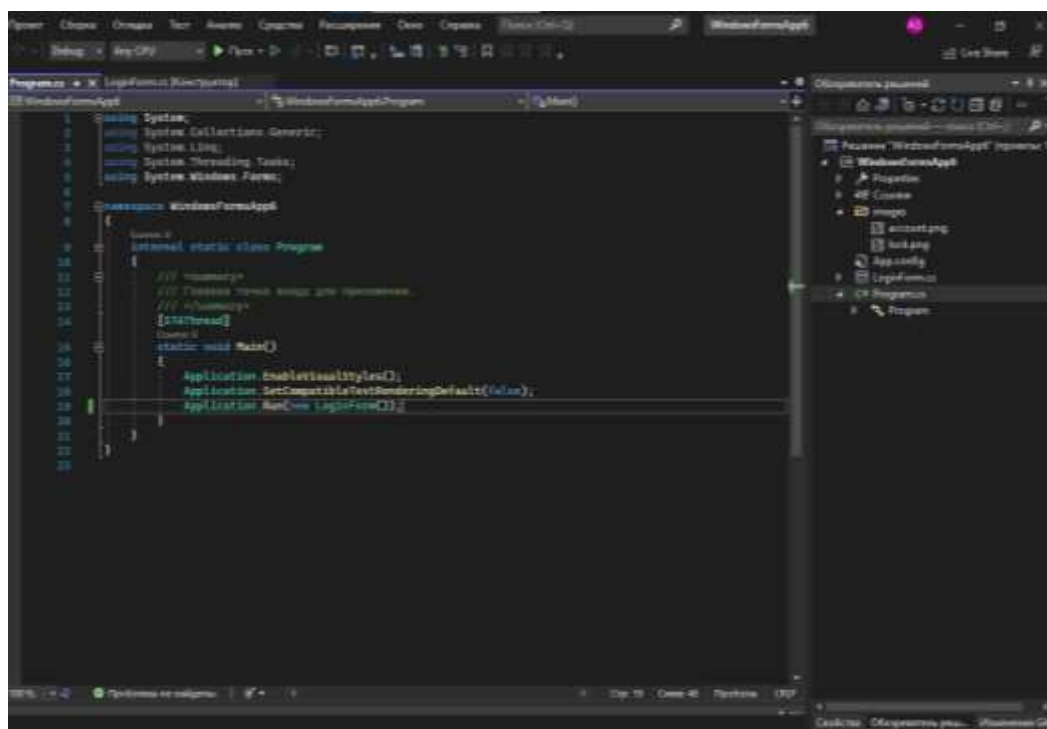


Рисунок 20 - Program.cs

16. Промежуточный результат

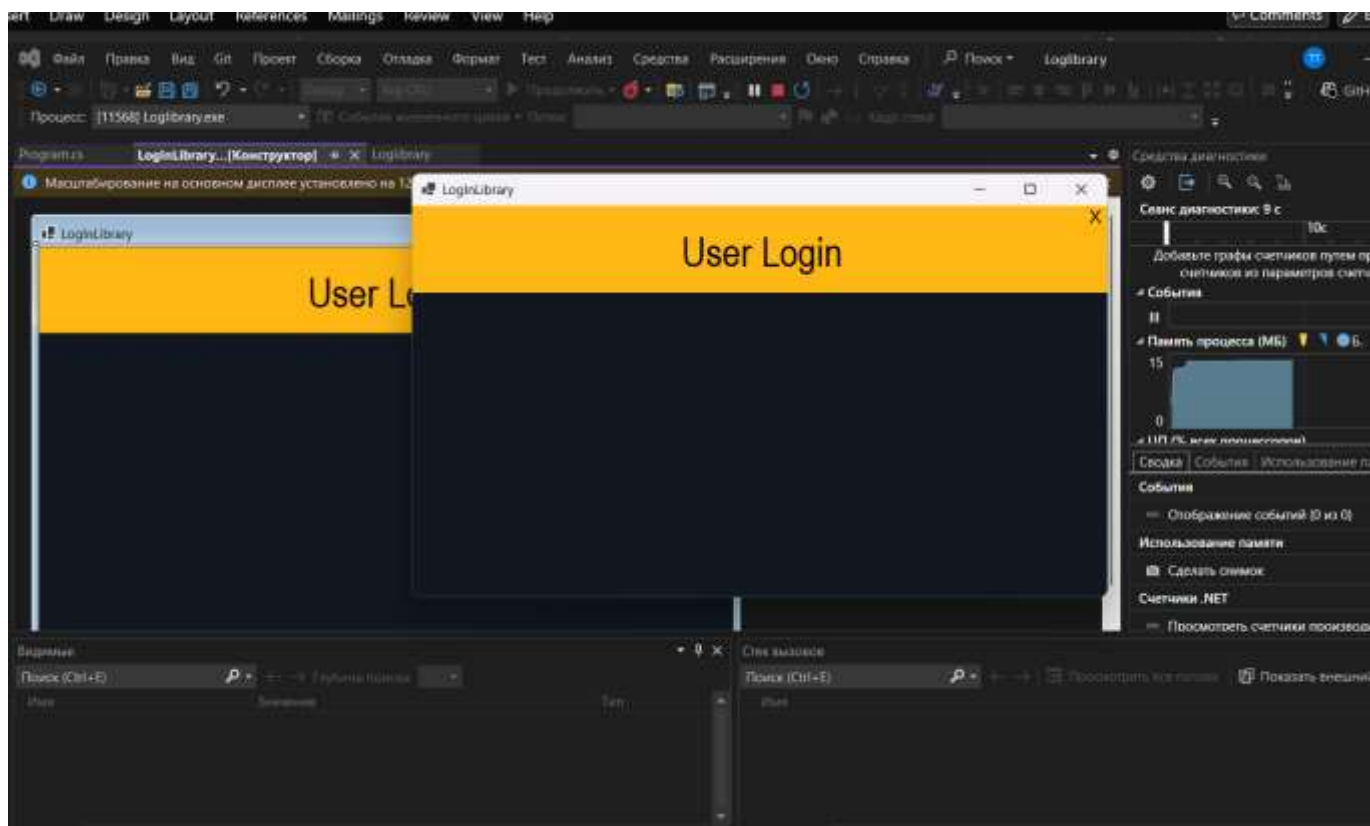


Рисунок 21 - Промежуточный результат



17. Заходим на панель элементов и создаем pictureBox, задаем ему размер и загружаем в него изображение из папки image

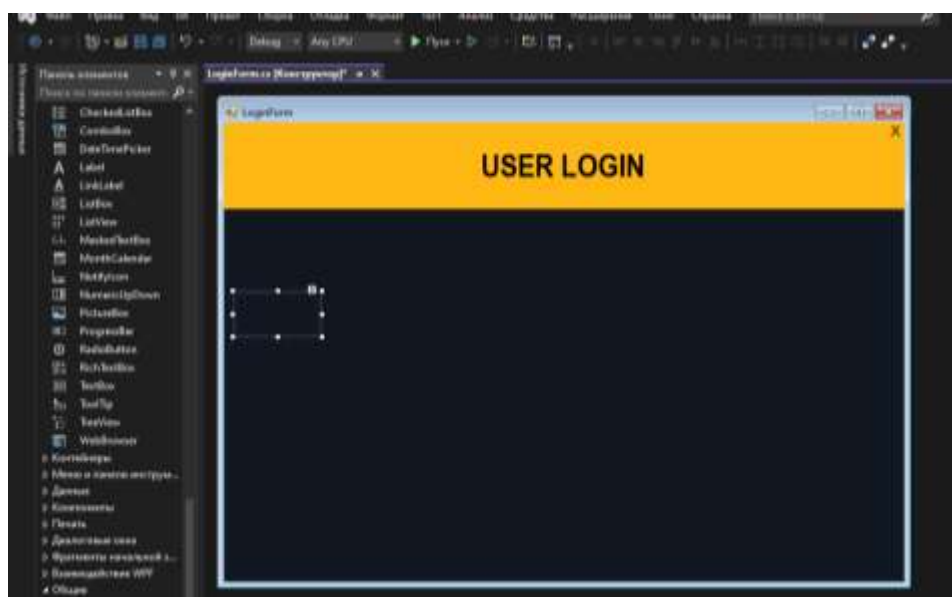


Рисунок 22 - pictureBox

Padding	0; 0; 0; 0
Size	50; 50
SizeMode	Normal

Рисунок 23 - Размер pictureBox

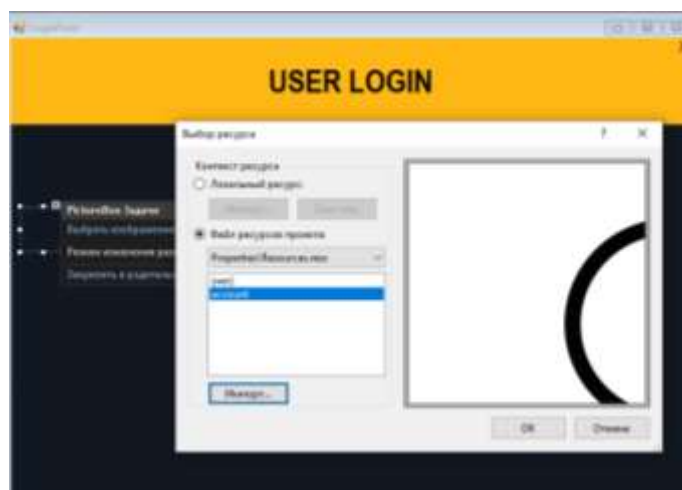


Рисунок 24 - Выбор изображения для pictureBox

## 18. Готовый pictureBox

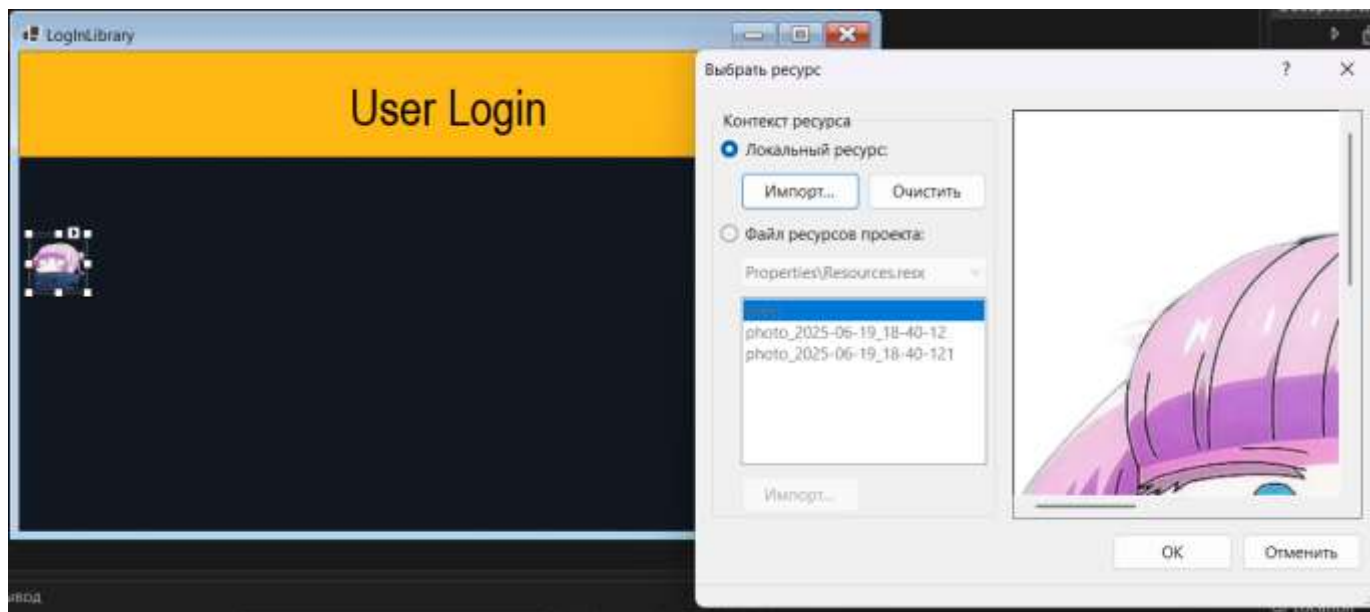


Рисунок 25 - Готовый pictureBox

## 19. Создаем textbox, в свойстве MultiLine ставим галочку и задаем размеры

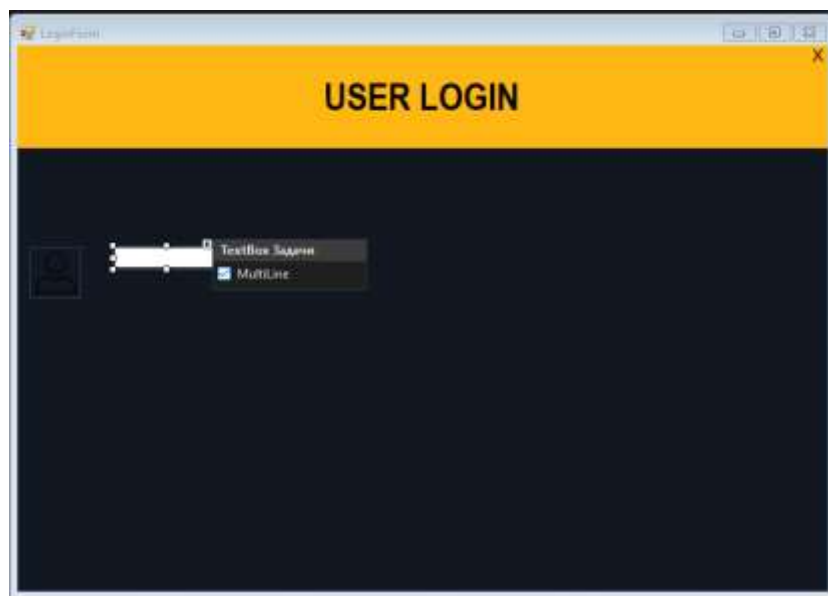


Рисунок 26 - Свойство MultiLine

ShortcutsEnabled	true
Size	350; 50
TabIndex	2

Рисунок 27 - Размеры textbox

20. Копируем сделанный pictureBox и textbox и меняем изображение в pictureBox на замок

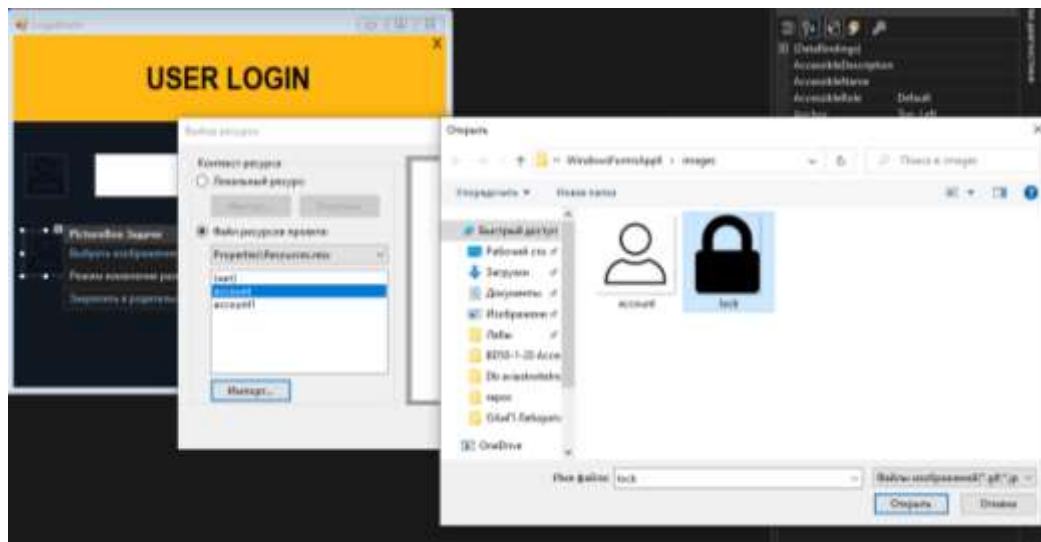


Рисунок 28 - Выбор изображения для pictureBox

21. Промежуточный результат

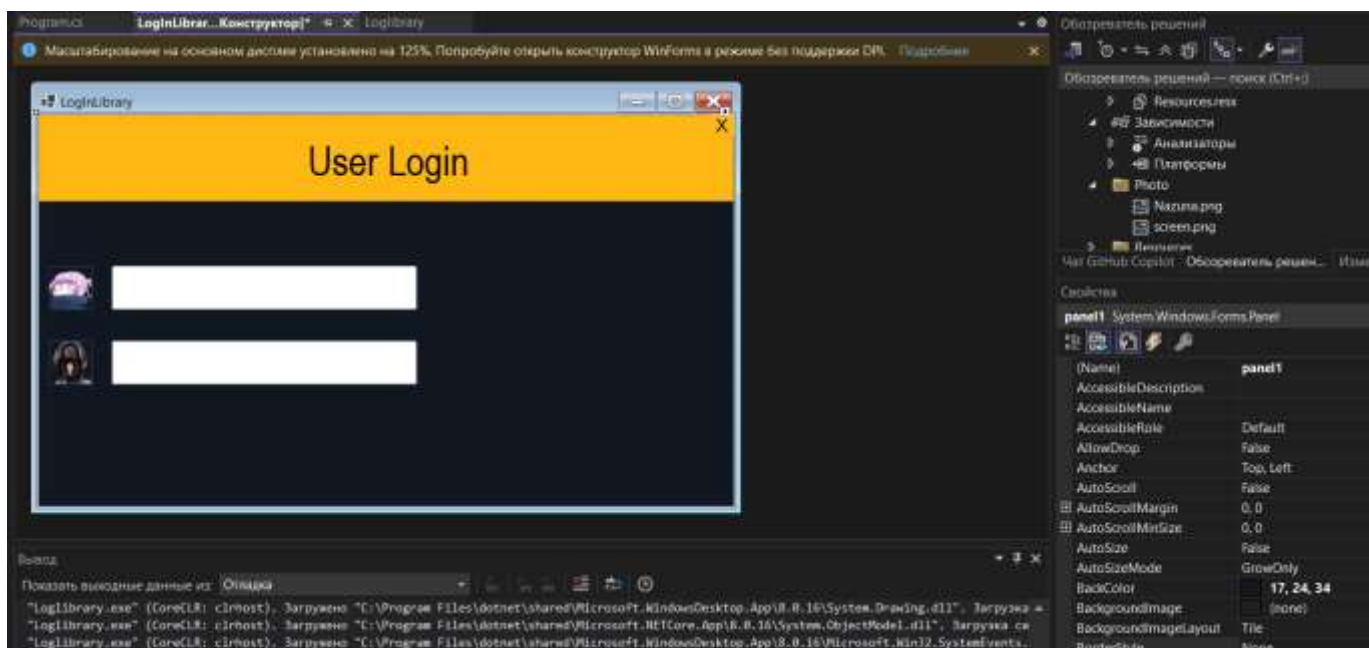


Рисунок 29 - Промежуточный результат

22. Создаем кнопку и настраиваем следующие свойства: BackColor и FlatStyle

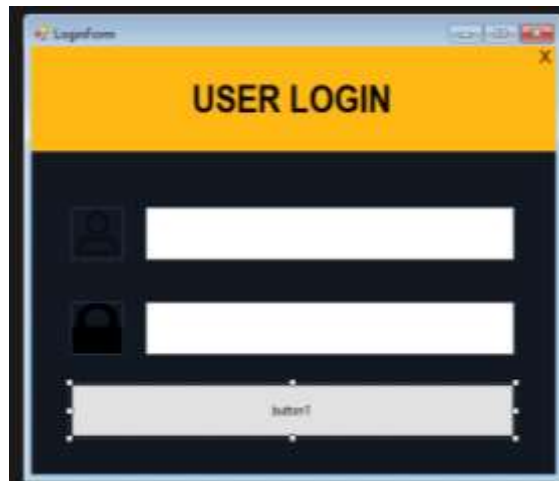


Рисунок 30 - Кнопка



Рисунок 31 - Цвет кнопки



Рисунок 32 - Свойство FlatStyle

## 23. Настраиваем шрифт кнопки и цвет текста

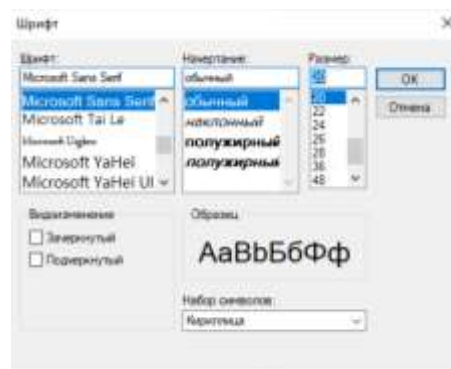


Рисунок 33 - Настройки шрифта

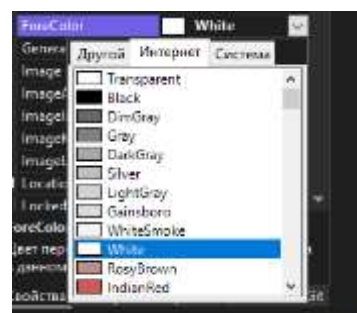


Рисунок 34 - Цвет текста

## 24. Убираем контур у кнопки

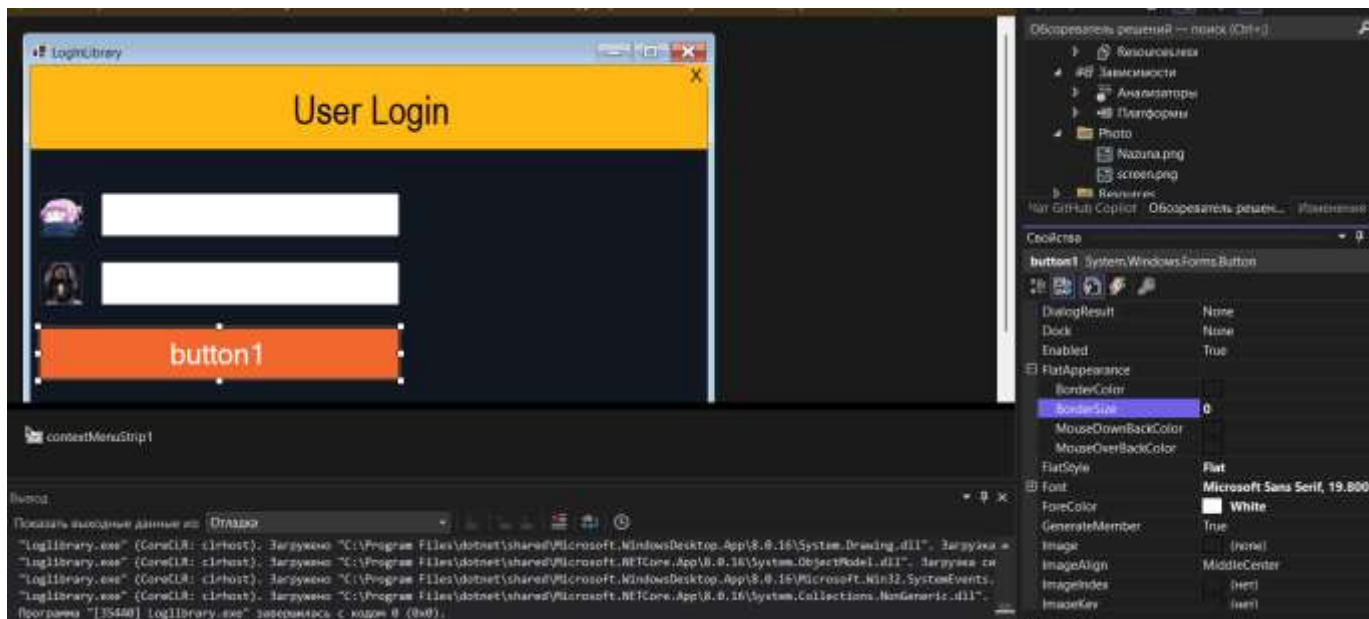


Рисунок 35 - Контур кнопки

## 25. У текстовых полей настраиваем шрифты

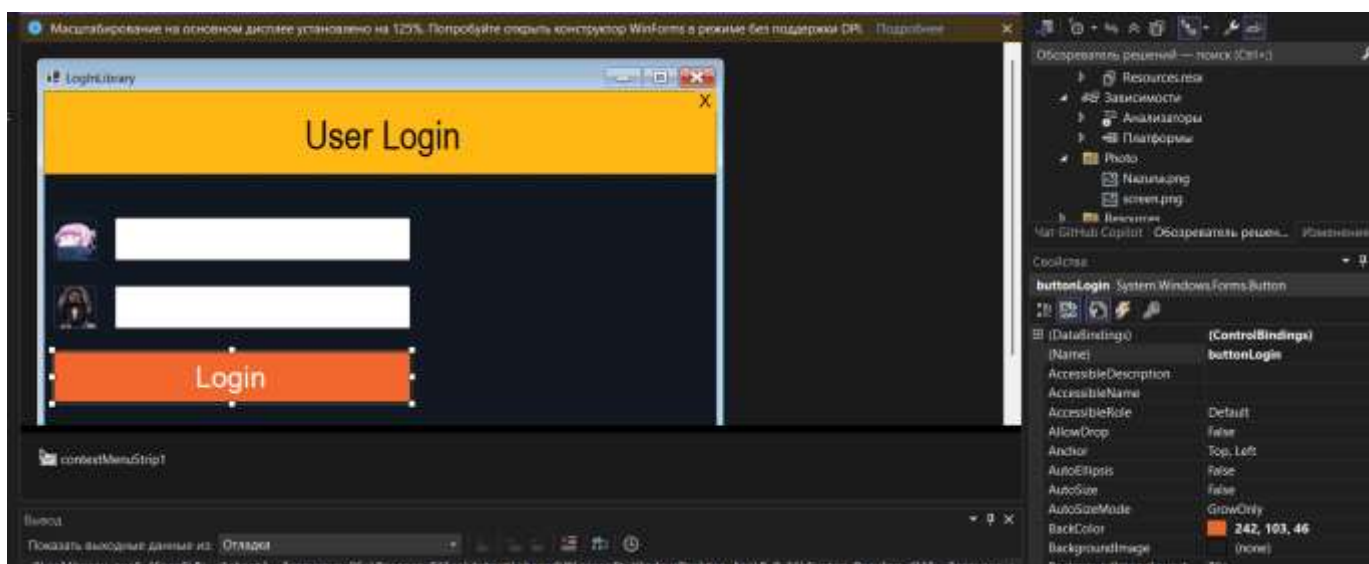


Рисунок 36 - Настройка шрифта

## 26. Далее переименовываем элементы

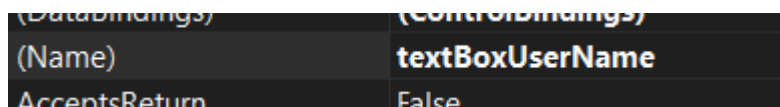


Рисунок 37 - Название textBoxName

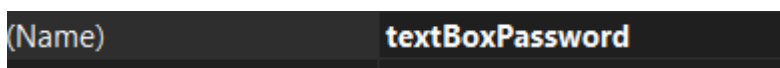


Рисунок 38 - Название textBoxPassword

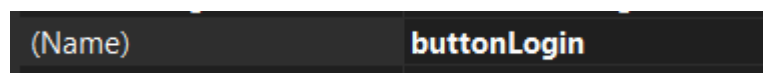


Рисунок 39 - Название buttonLogin

27. Заходим в настройки textBoxPassword и свойство UseSystemPasswordChar  
меняем на true и в свойстве MultiLine убираем галочку

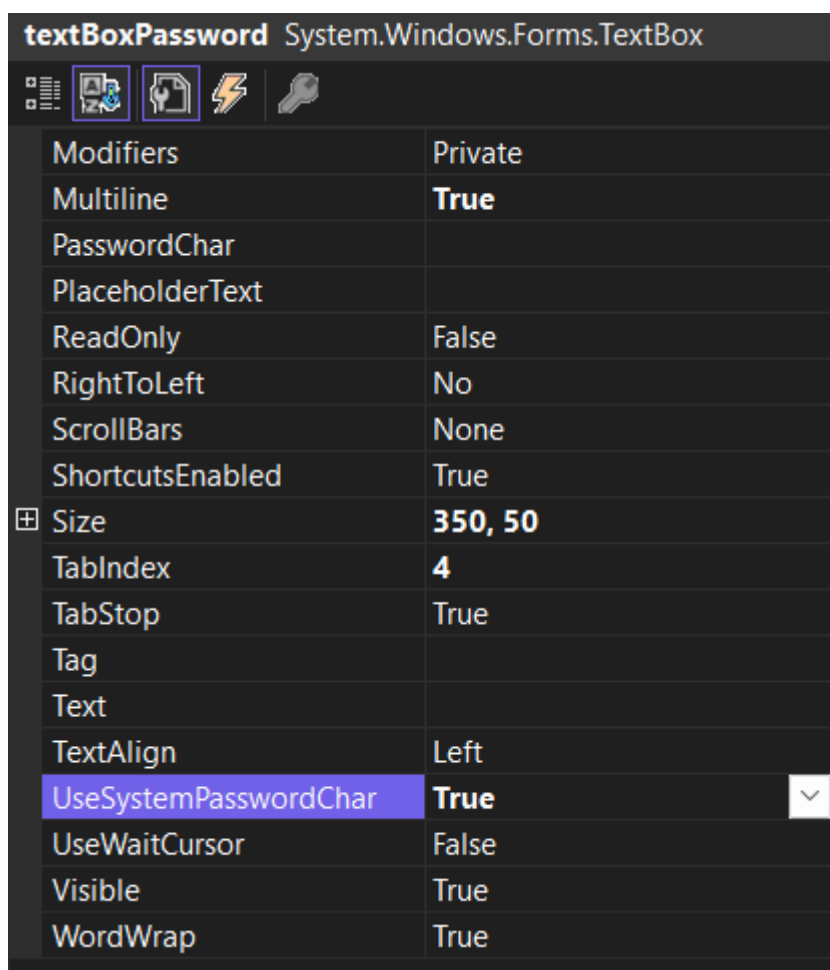


Рисунок 40 - Свойство UseSystemPassword

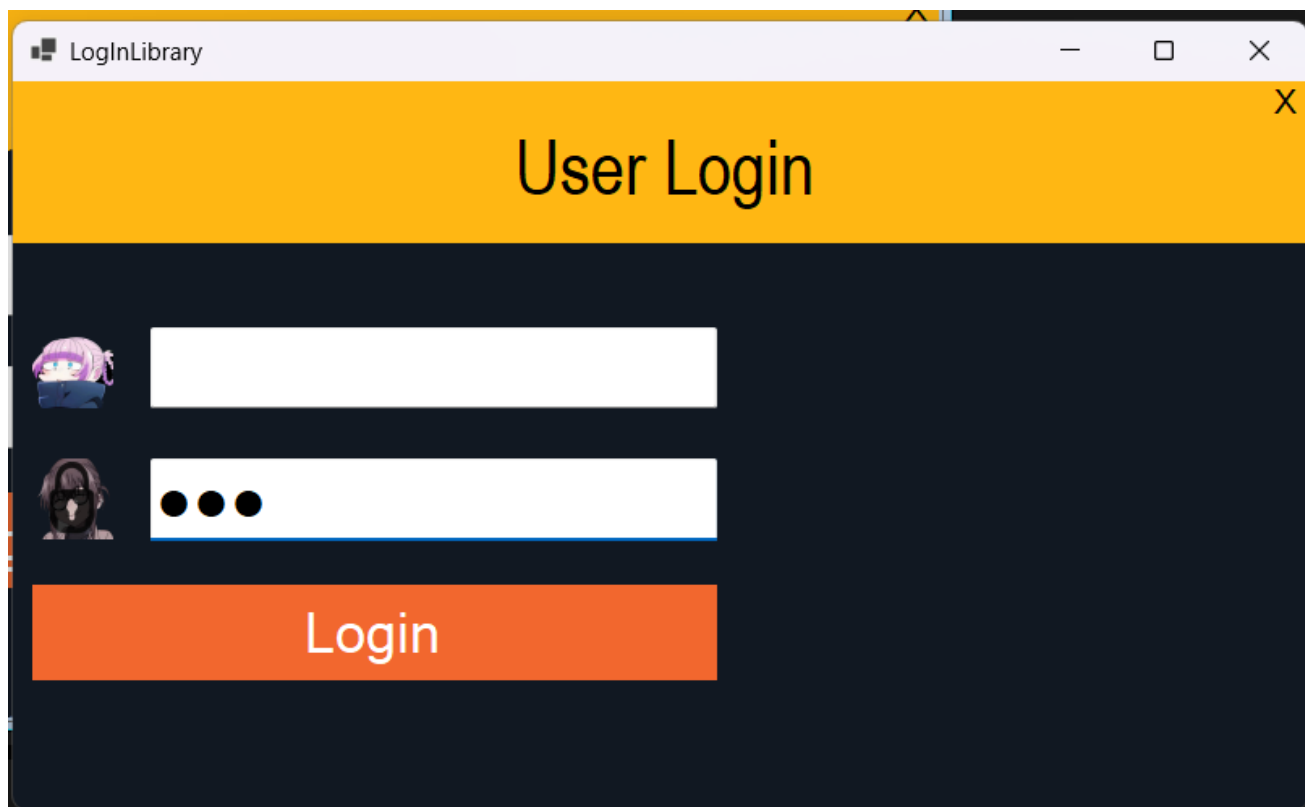
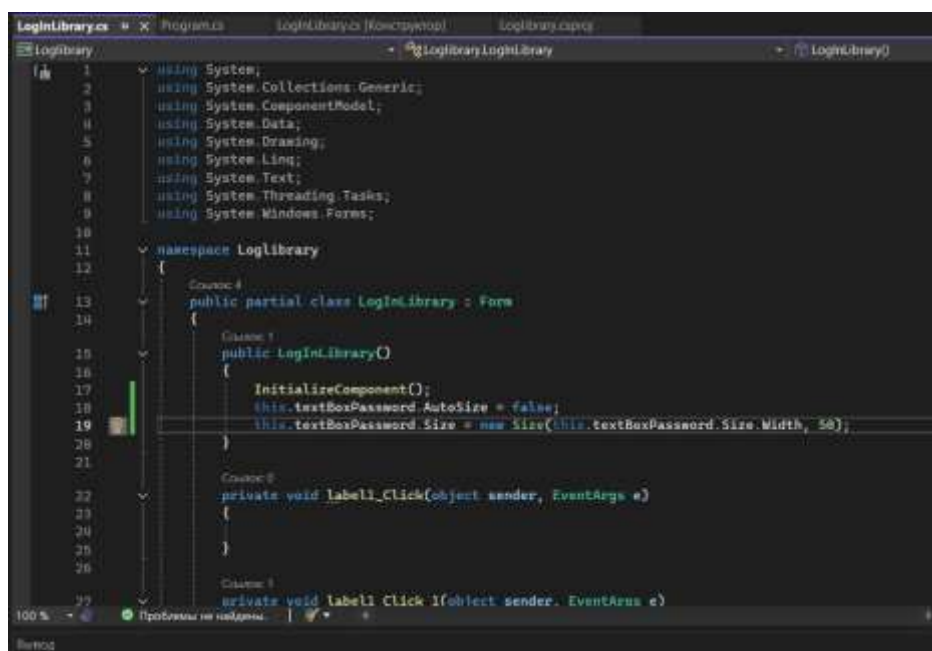


Рисунок 41 - Свойство MultiLine

Рисунок 42 - Промежуточный результат

## 28. Переходим к коду, задаем textBoxPassword размер



```
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Data;
5 using System.Drawing;
6 using System.Linq;
7 using System.Text;
8 using System.Threading.Tasks;
9 using System.Windows.Forms;
10
11 namespace LogLibrary
12 {
13     public partial class LoginLibrary : Form
14     {
15         public LoginLibrary()
16         {
17             InitializeComponent();
18             this.textBoxPassword.AutoSize = false;
19             this.textBoxPassword.Size = new Size(this.textBoxPassword.Size.Width, 50);
20         }
21
22         private void label1_Click(object sender, EventArgs e)
23         {
24         }
25
26         private void label1_Click_1(object sender, EventArgs e)
27         {
28         }
29     }
30 }
```

Рисунок 43 - Размер textbox

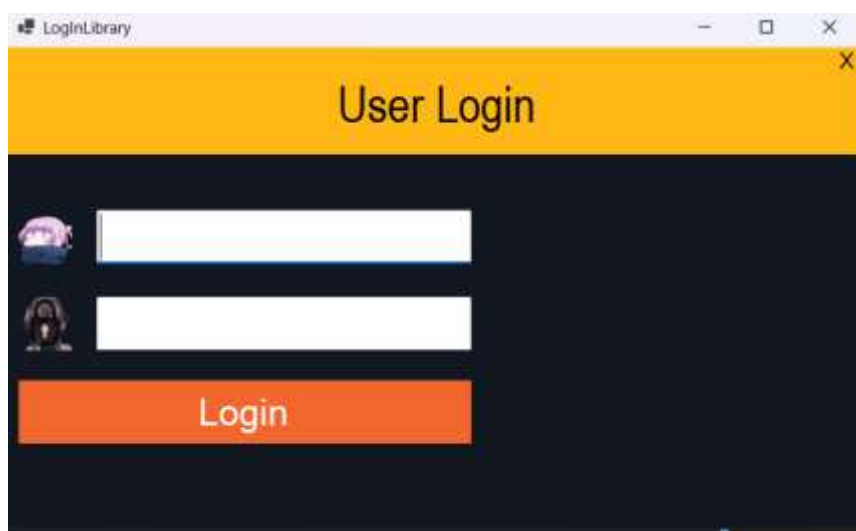


Рисунок 44 - Промежуточный результат

## 29. Заходим в свойства LoginForm и настраиваем, чтобы форма открывалась по середине экрана и у нее не было конутра

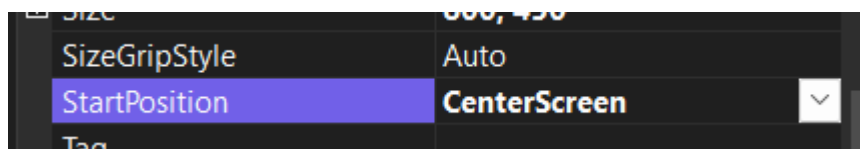


Рисунок 45 - Свойство CenterScreen



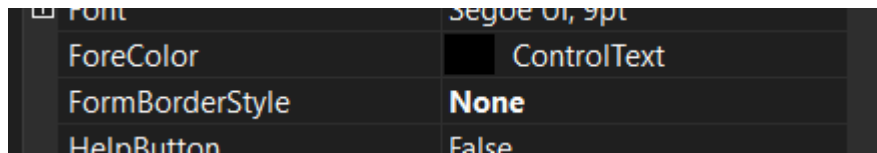


Рисунок 46 - Свойство FormBorderStyle

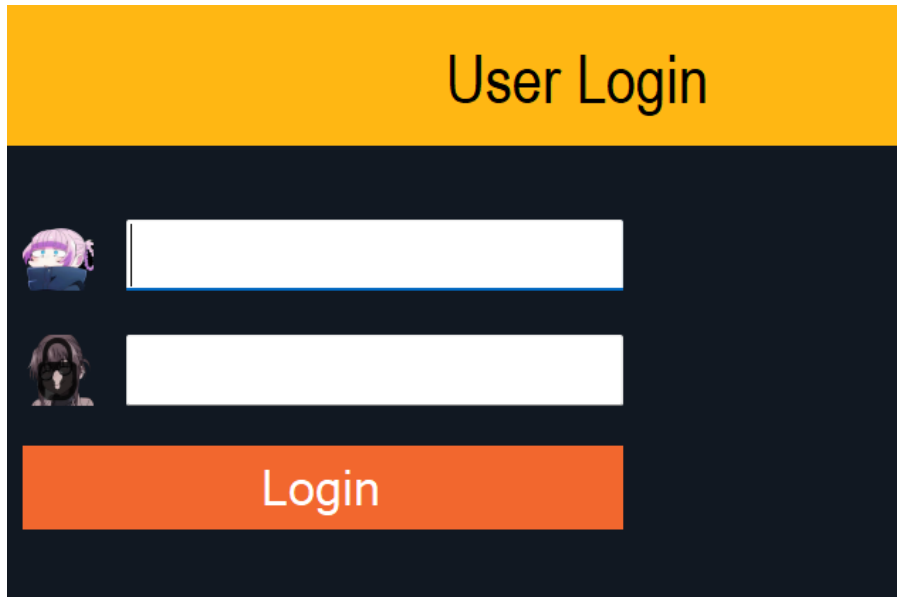


Рисунок 47 - Промежуточный результат

30. Заходим в настройки свойств labelClose и настраиваем цвет и курсор, шрифт



Рисунок 48 - Свойство Cursor



Рисунок 49 - Цвет

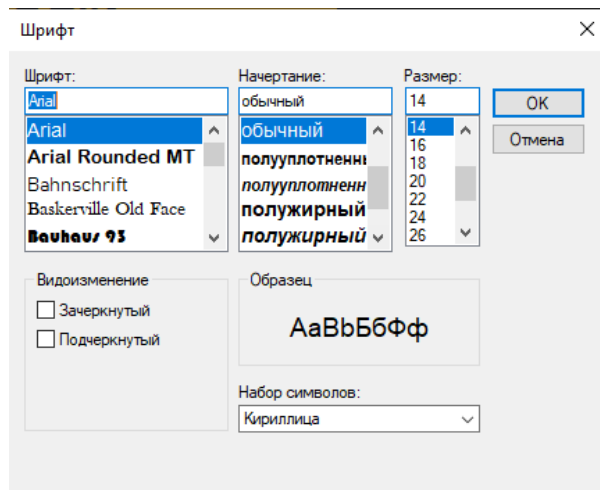


Рисунок 50 - Настройка шрифта

31. Заходим в события `labelClose` и находим `MouseEnter` и `MouseLeave` и задаем цвета, чтобы `label` был белым, а при наведении курсора черным

Рисунок 51 - Событие `MouseEnter`

Рисунок 52 - Код события `MouseEnter`

Рисунок 53 - Событие `MouseLeave`

<code>MouseClick</code>	
<code>MouseDoubleClick</code>	
<code>MouseDown</code>	
<code>MouseEnter</code>	<code>labelClose_MouseEnter</code>
<code>MouseHover</code>	
<code>MouseLeave</code>	<code>labelClose_MouseLeave</code>
<code>MouseMove</code>	

Рисунок 54 – Код события `MouseLeave`

### 32. Готовая форма

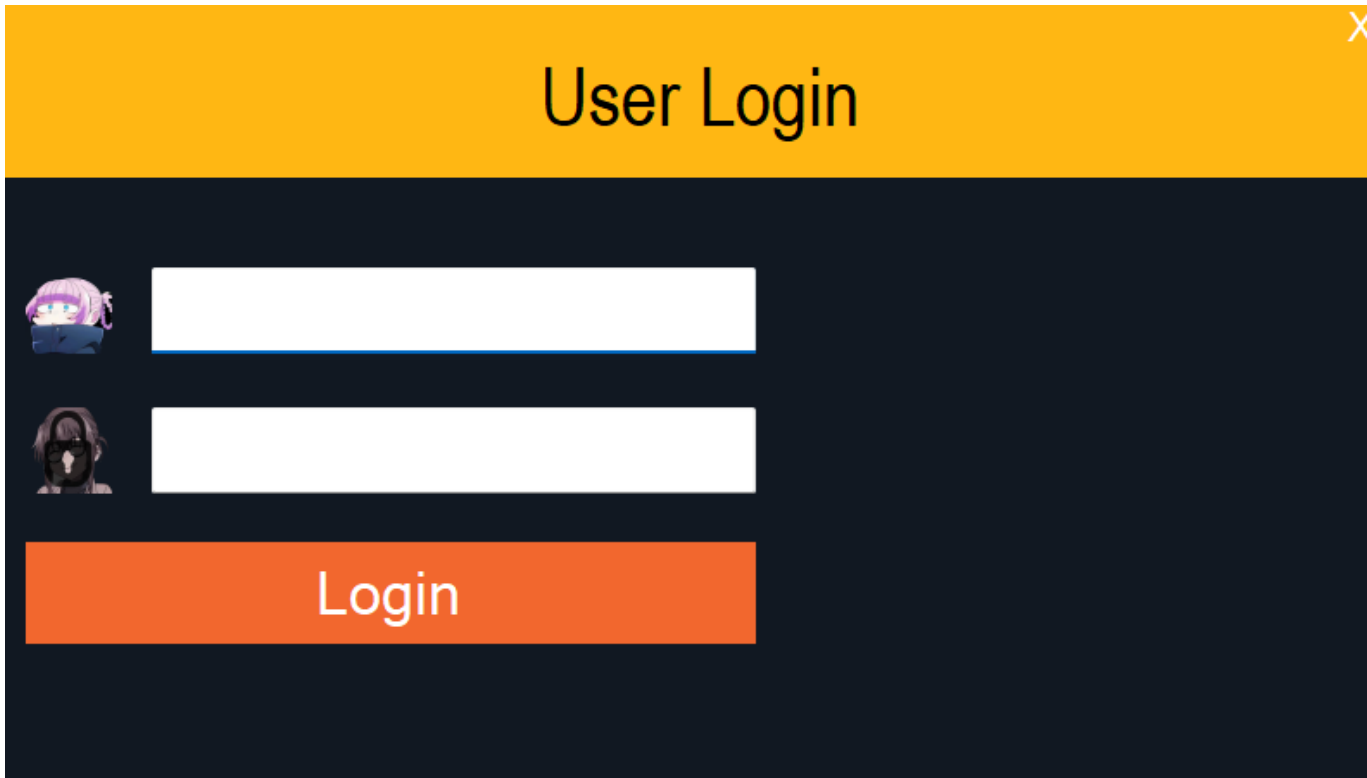
A user login form with a yellow header containing the text "User Login" and a small "X" icon in the top right corner. The form has a dark blue background. It contains two input fields, each preceded by a small avatar icon (one with pink hair, one with brown hair). Below the input fields is an orange button with the text "Login".

Рисунок 55 -Готовая форма

### 33. Форма при наведении курсора на кнопку закрыть

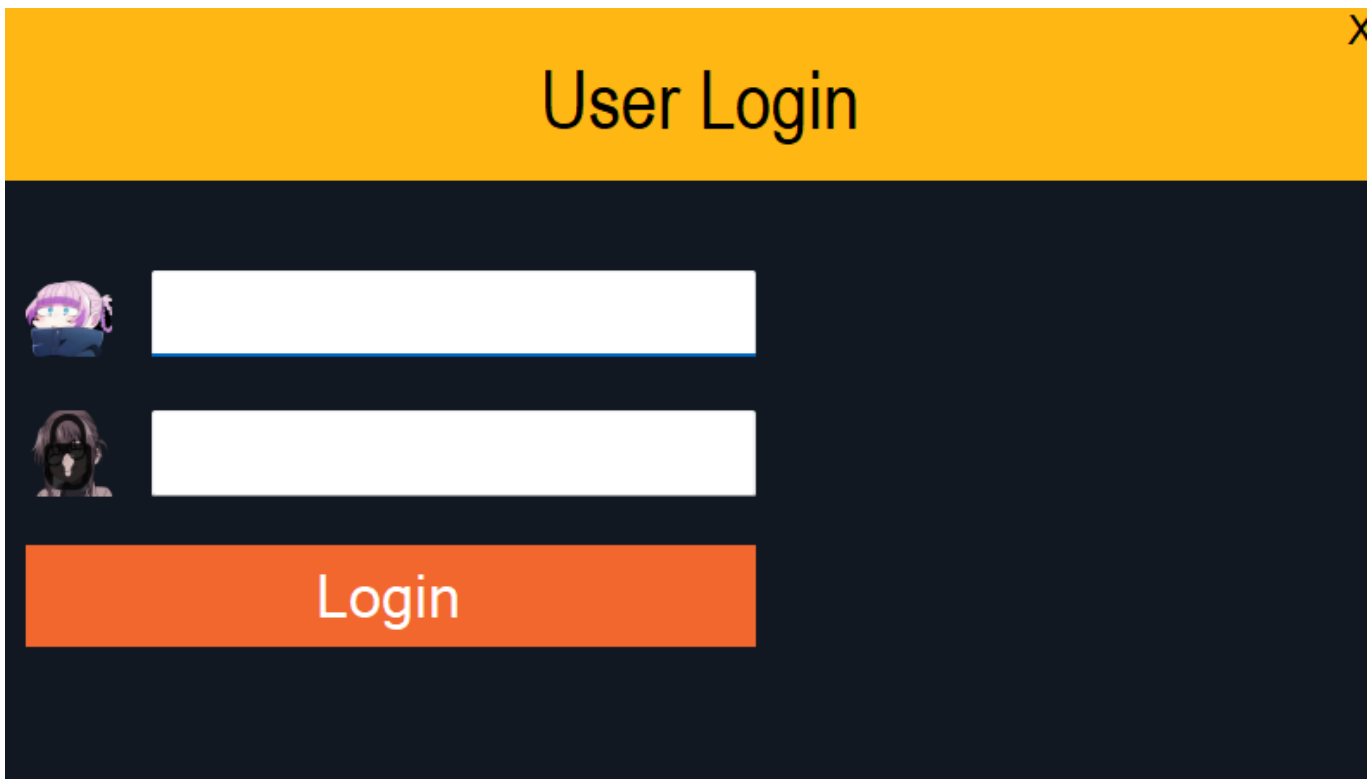
A user login form identical to the one in Figure 55, but with a small "X" icon in the top right corner of the yellow header, indicating a close button.

Рисунок 56 - Форма при наведении курсора на кнопку закрыть

1. Заходим в события label, который у нас выполняет роль выхода из приложения (крестик), и пишем событие, чтобы при нажатии на него, приложение закрылось

```

Ссылка: 1
private void labelClose_Click(object sender, EventArgs e)
{
    this.Close();
}

```

Рисунок 57 - Функция labelClose\_Click

2. Заходим на сайт, представленный на скрине и скачиваем mysql connector

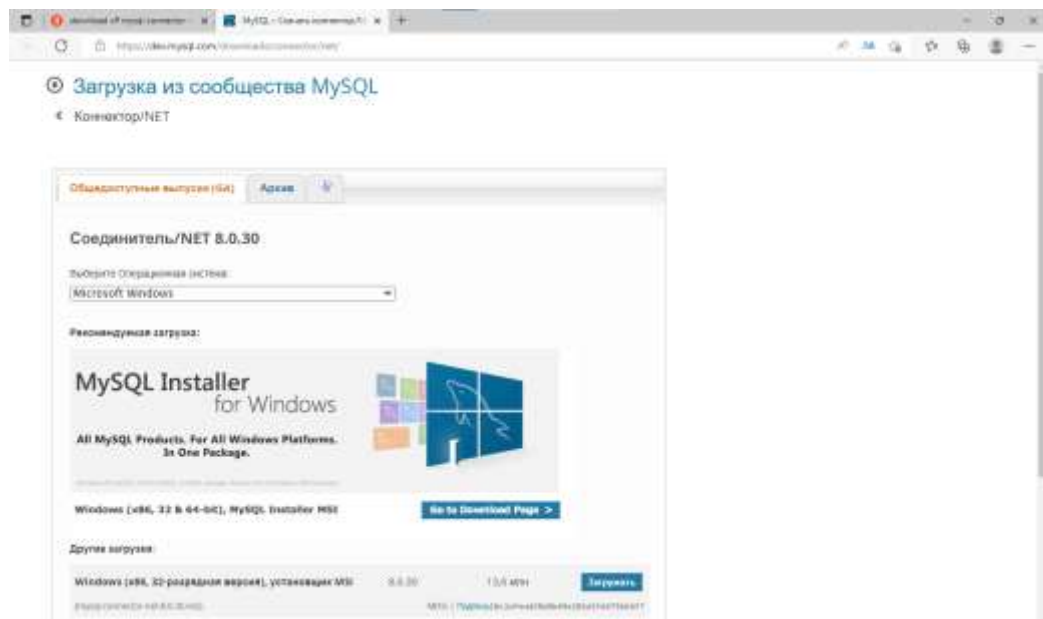


Рисунок 58 - Сайт mysql connector

3. Далее необходимо выполнить установку

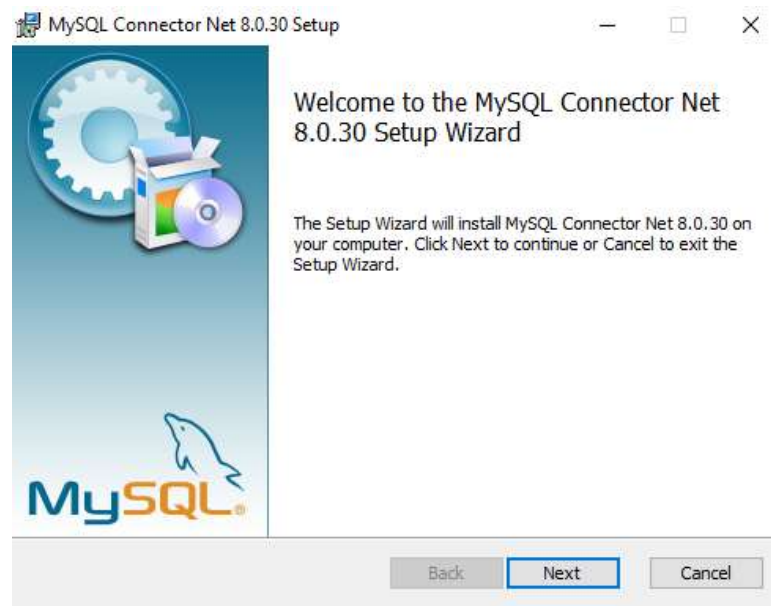


Рисунок 59 - Установка



4. Заходим в менеджер ссылок проекта и добавляем MySql.Data

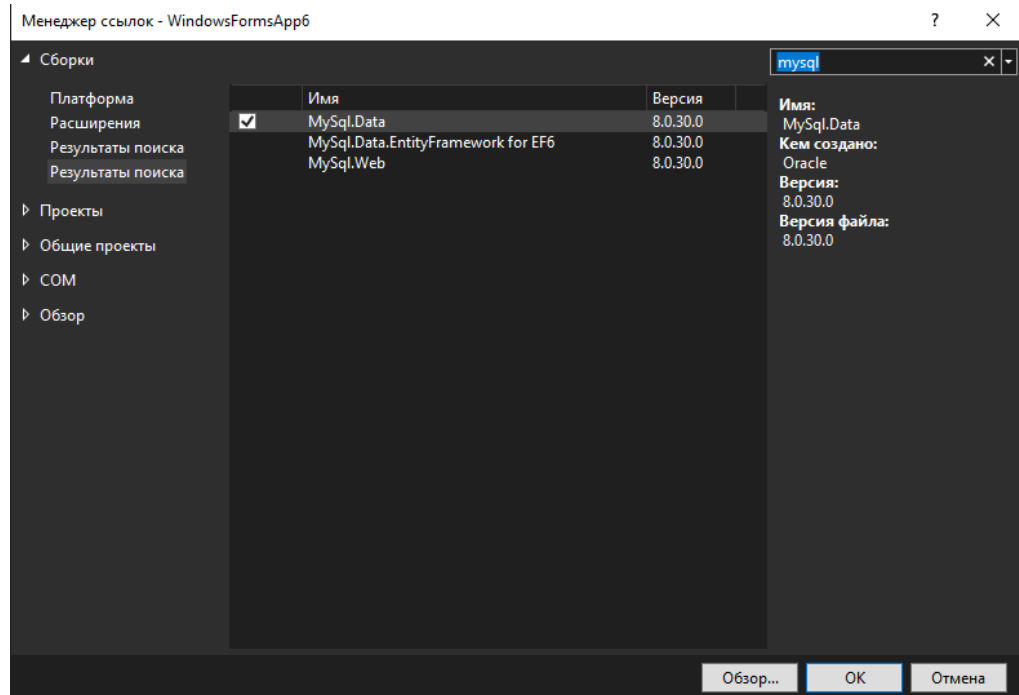


Рисунок 59 - Менеджер ссылок

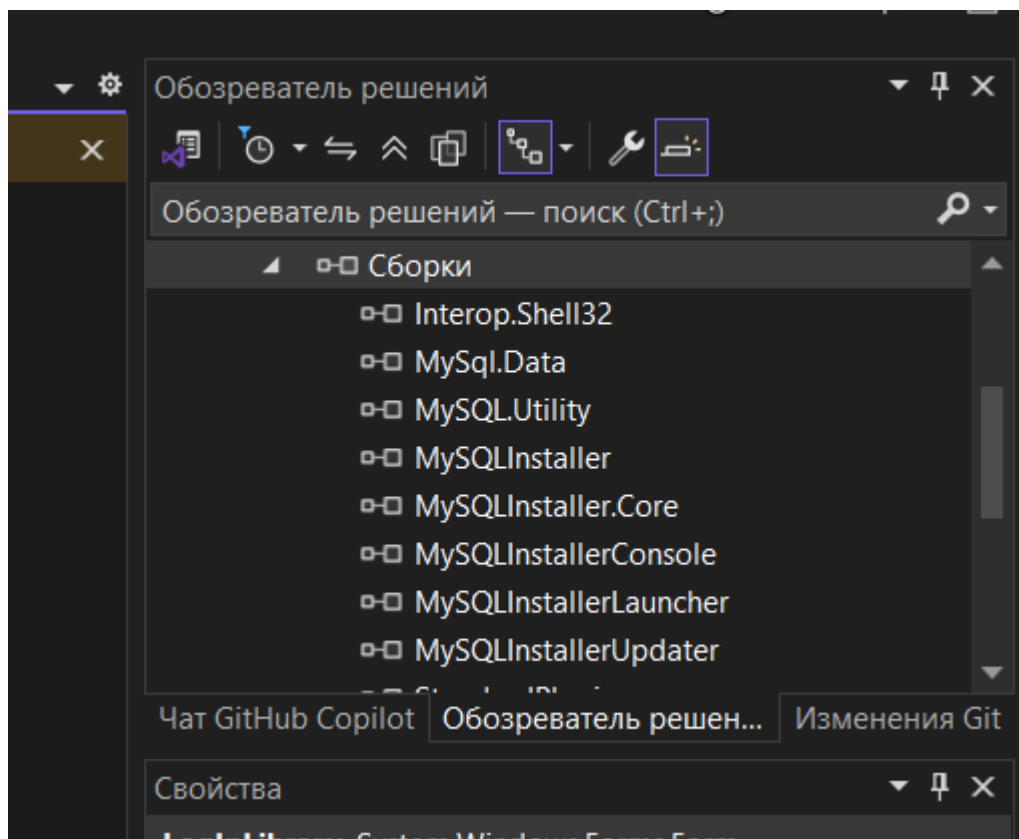


Рисунок 60 - Добавленные ссылки

5. Создаем подключение и в пространство имен подключаем `MySQL.Data.MySqlClient`

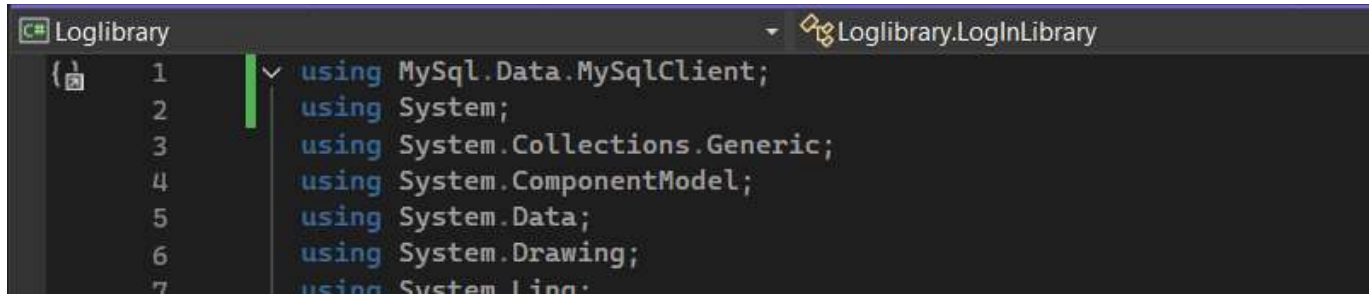


Рисунок 61 - Пространство имен

6. Запускаем сервер и заходим в PhpMyAdmin

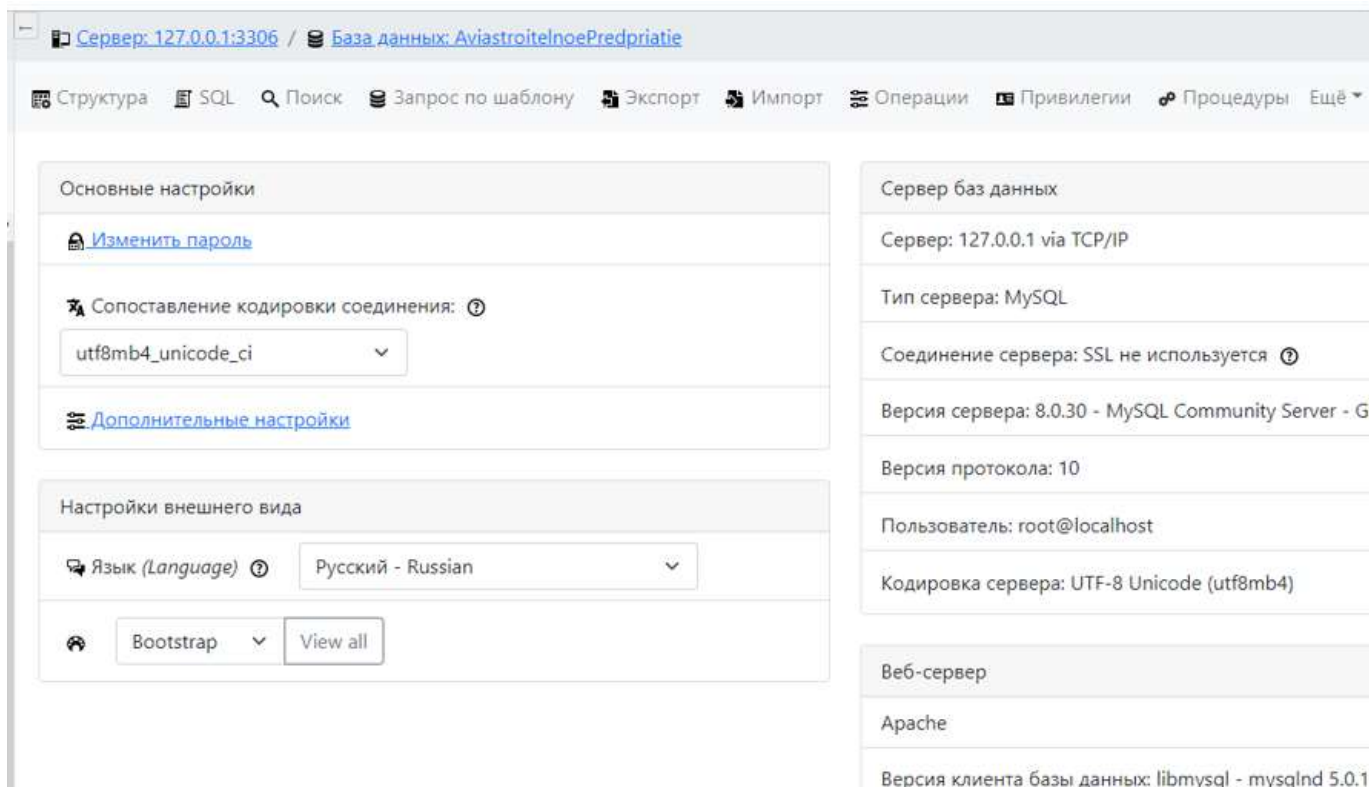


Рисунок 62 - PhpMyAdmin

7. Создаем новую бд под названием `csharp_users_db`

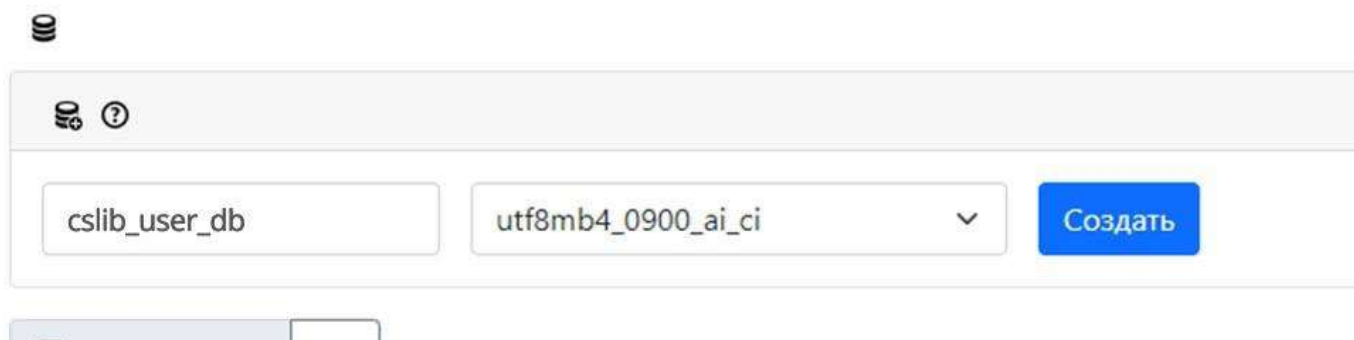
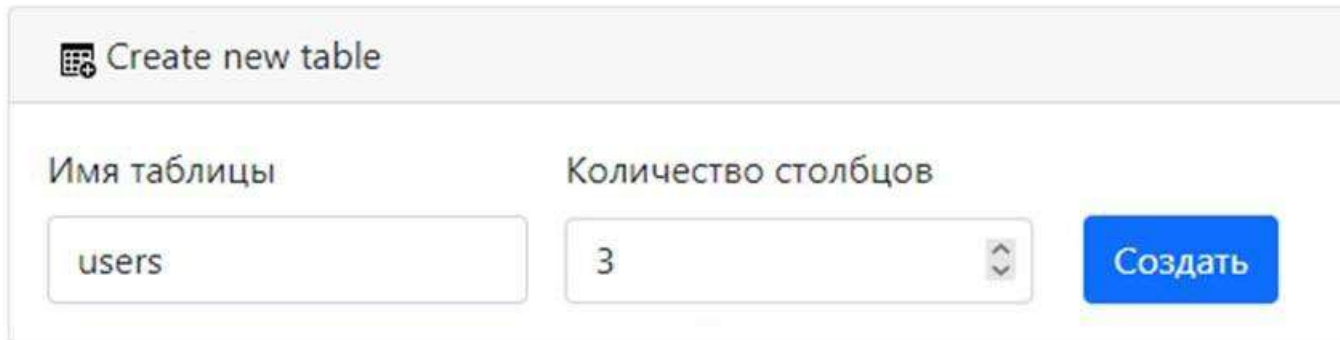


Рисунок 63 - Создание бд

8. Создаем новую таблицу, указывая имя таблицы и количество столбцов



Create new table

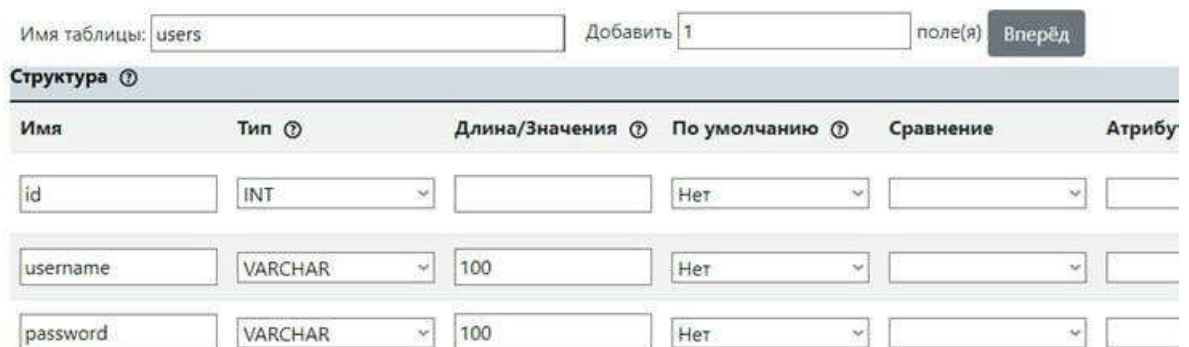
Имя таблицы: users

Количество столбцов: 3

Создать

Рисунок 64 - Создание таблицы

9. Заполняем в таблице атрибуты



Имя таблицы: users    Добавить 1 поле(я)    Вперёд

Структура ⓘ

Имя	Тип ⓘ	Длина/Значения ⓘ	По умолчанию ⓘ	Сравнение	Атрибу
id	INT		Нет		
username	VARCHAR	100	Нет		
password	VARCHAR	100	Нет		

Рисунок 65 - Заполнение таблицы

10. Создаем подключение к бд, указывая сервер, порт, логин, пароль и название, созданной ранее бд



```
private MySqlConnection connection = new MySqlConnection("server=localhost; port=3306; username=root; password=; database=mysqlib_user_db");  
  
Событие: 1  
private void label1_Click_1(object sender, EventArgs e)  
{
```

Рисунок 66 - Подключение бд

11. Создаем новый класс



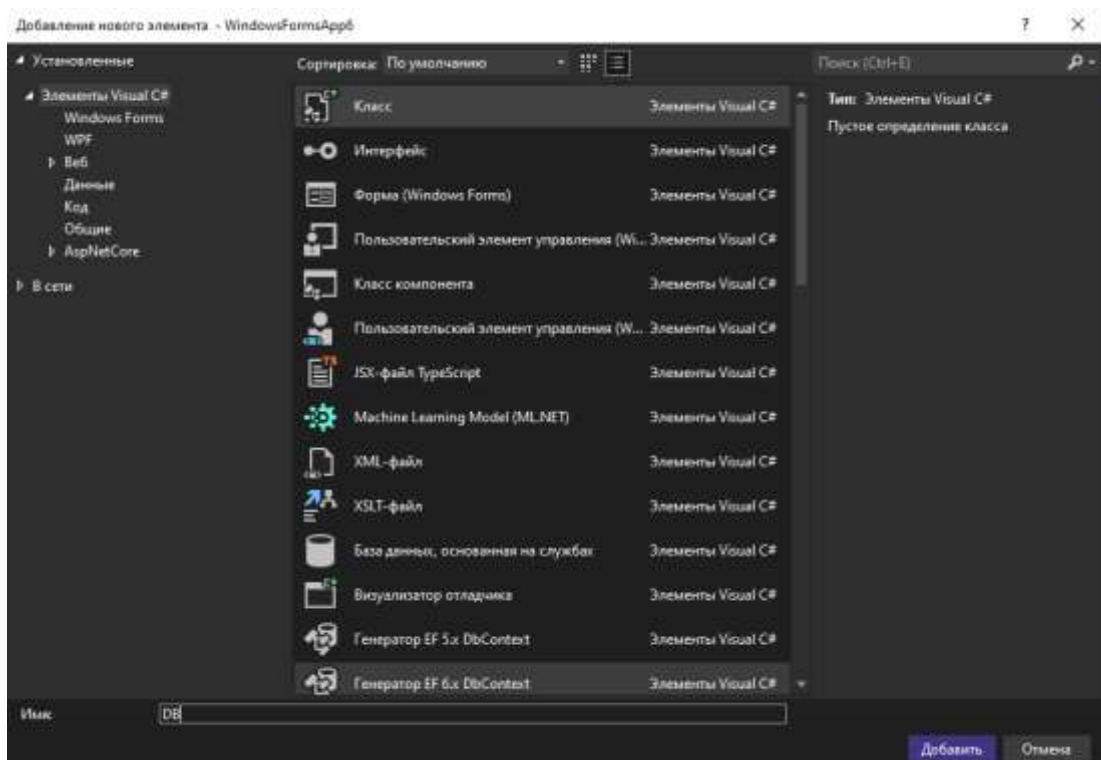


Рисунок 67 - Создание нового класса

## 12. Копируем наше подключение бд в новый созданный класс

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WindowsFormsApp6
{
    internal class DB
    {
        private MySqlConnection connection = new MySqlConnection("server=localhost;port=3306;username=root;password=;database=sharp_users_db");
    }
}
```

Рисунок 68 - Созданный класс

## 13. Пишем функцию для открытия соединений

Рисунок 69 - Открытие соединения

## 14. Создаем функцию для закрытия соединения

Рисунок 70 - Закрыие соединения

## 15. Создаем функцию для возврата соединения

```

Console:0
internal class DB
{
    Console:0
    public void openConnection()
    {
        if(connection.State == System.Data.ConnectionState.Closed)
        {
            connection.Open();
        }
    }
    Console:0
    public void closeConnection()
    {
        if (connection.State == System.Data.ConnectionState.Closed)
        {
            connection.Close();
        }
    }
    Console:0
    public MySqlConnection GetConnection()
    {
        return connection;
    }
    private MySqlConnection connection = new MySqlConnection("server=localhost; port=3306; username=llsaf; password=; database=cslib_user_db");
}

```

Рисунок 71 - Возврат соединения

16. Пишем событие кнопки login, в котором происходит обращение к бд с данными пользователя для получения пароля и логина, для входа и их проверки

```
Ссылка 1
private void buttonLogin_Click(object sender, EventArgs e)
{
    DB dB = new DB();
    String username = textBoxUserName.Text;
    String password = textBoxPassword.Text;

    DataTable table = new DataTable();
    MySqlDataAdapter adapter = new MySqlDataAdapter();
    MySqlCommand command = new MySqlCommand("SELECT * FROM 'users' WHERE 'username' = @usn and 'password' = @pass", dB.GetConnection());

    command.Parameters.Add("@usn", MySqlDbType.VarChar).Value = username;
    command.Parameters.Add("@pass", MySqlDbType.VarChar).Value = password;

    adapter.SelectCommand = command;
    adapter.Fill(table);

    if (table.Rows.Count > 0)
    {
        MessageBox.Show("YES");
    }
    else
    {
        MessageBox.Show("NO");
    }
}
```

Рисунок 72 - Функция кнопки Login

Вносим в нашу бд данные пользователя для проверки авторизации

The screenshot shows a database management interface with a table structure. The table has three columns: 'id' (int), 'username' (varchar(100)), and 'password' (varchar(100)). The 'username' field is populated with 'Nazuna' and the 'password' field is populated with '24Agree'. A 'Вперёд' button is visible at the bottom right.

Столбец	Тип	Функция	Null	Значение
id	int			
username	varchar(100)			Nazuna
password	varchar(100)			24Agree

Рисунок 73 - Внесения данных в бд

Тестируем результат

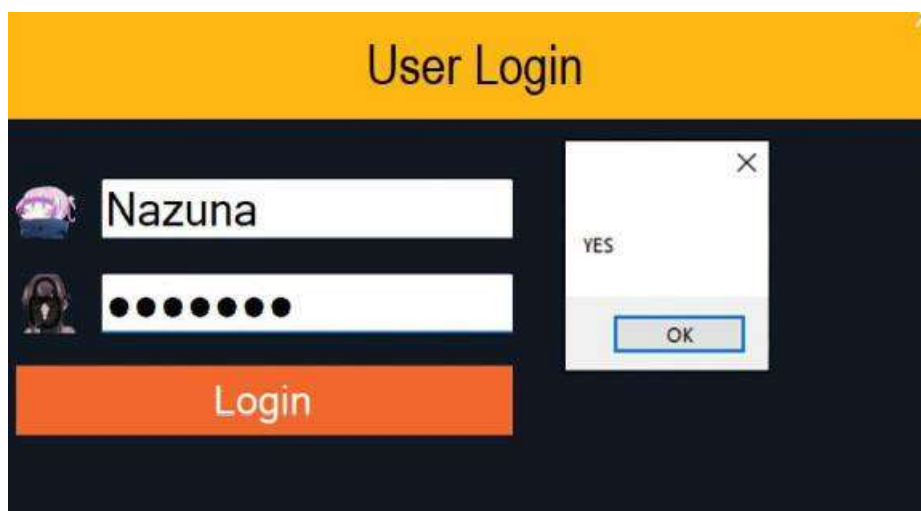


Рисунок 74 - Результат при введении верных логин и пароль

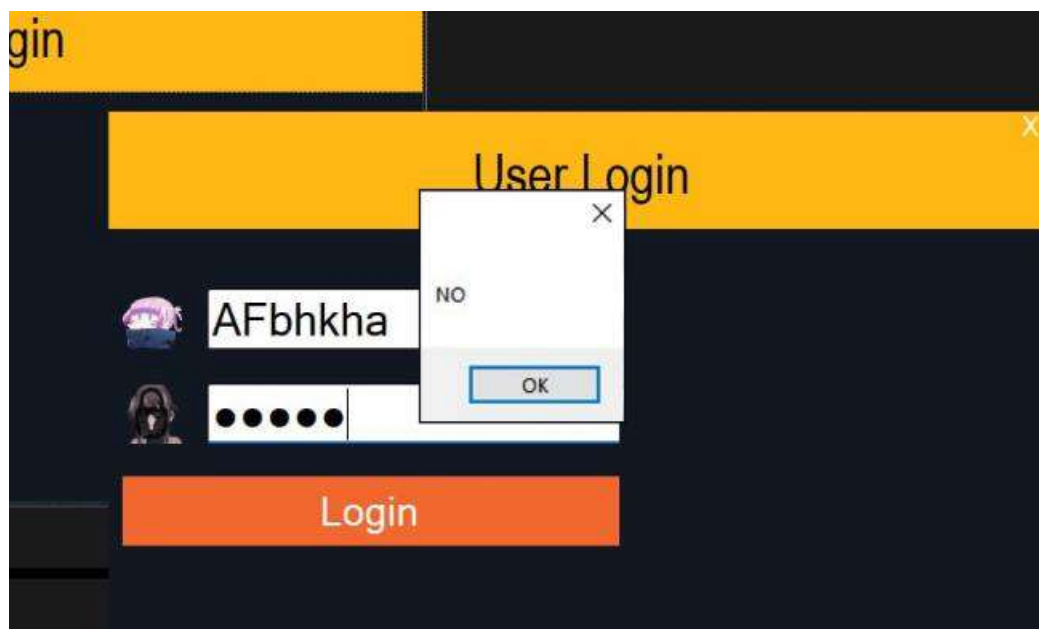


Рисунок 75 - Результат при введении неверных логин и пароль

Добавляем новую форму в проект

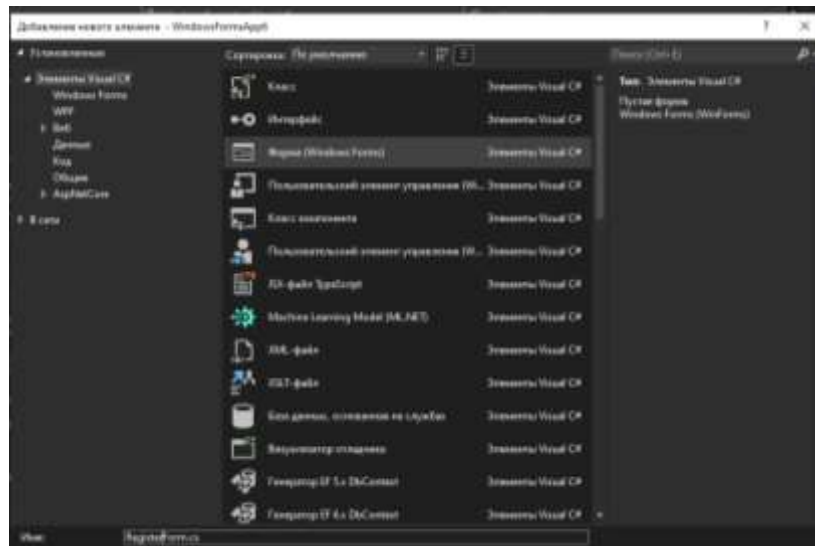


Рисунок 57 - Новая форма

Копируем содержимое LoginForm на новую созданную форму RegisterForm

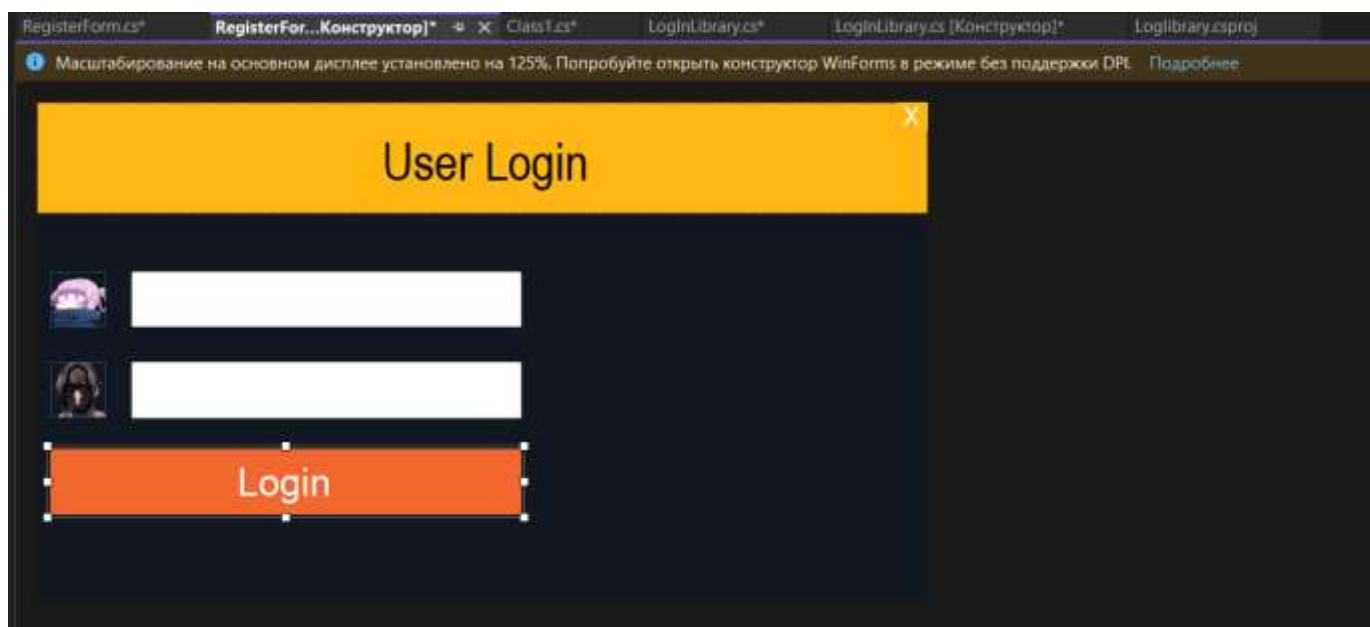


Рисунок 58 - Скопированная LoginForm

Редактируем, добавляем textbox

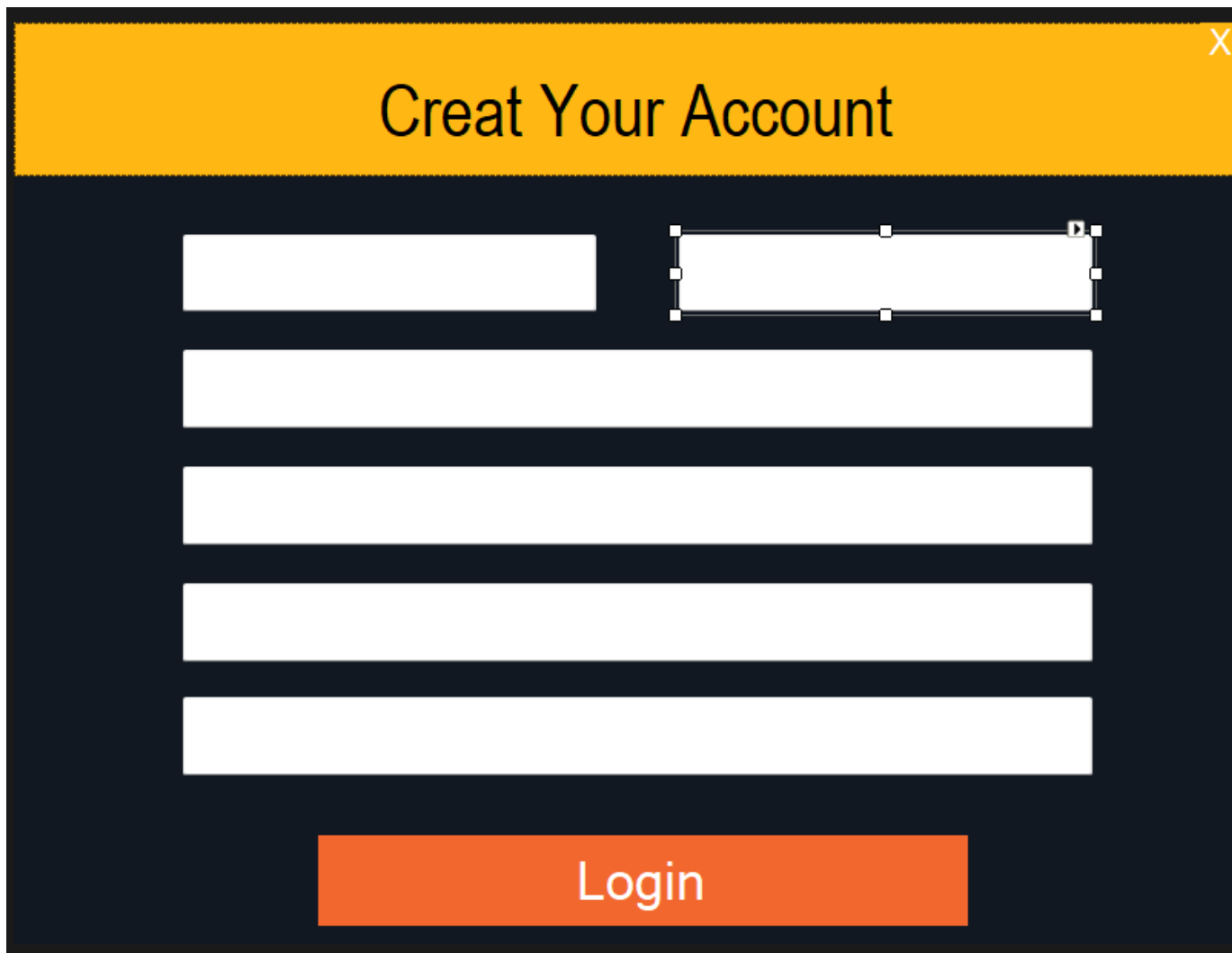


Рисунок 59 - Отредактированная форма RegisterForm

В Program.cs меняем LoginForm на RegisterForm, чтобы при запуске приложения запускалась она

```
{
    // To customize application configuration such as set high DPI settings or default font,
    // see https://aka.ms/applicationconfiguration.
    ApplicationConfiguration.Initialize();
    Application.Run(new RegisterForm());
}
```

Рисунок 60 - Program.cs

Заходим в свойства формы и задаем, чтобы она запускалась по центру экрана



Рисунок 61 – StartPosition

Переименовываем все textbox



Рисунок 62 - textBoxFirstName

(DataBindings)	(ControlBindings)
(Name)	<b>textBoxLastName</b>
AcceptsReturn	False

Рисунок 63 - textBoxLastName

(Name)	<b>TextBoxEmail</b>
--------	---------------------

Рисунок 64 - textBoxEmail

(DataBindings)	(ControlBindings)
(Name)	<b>textBoxUserName</b>
AcceptsReturn	False

Рисунок 65 - textBoxUsername

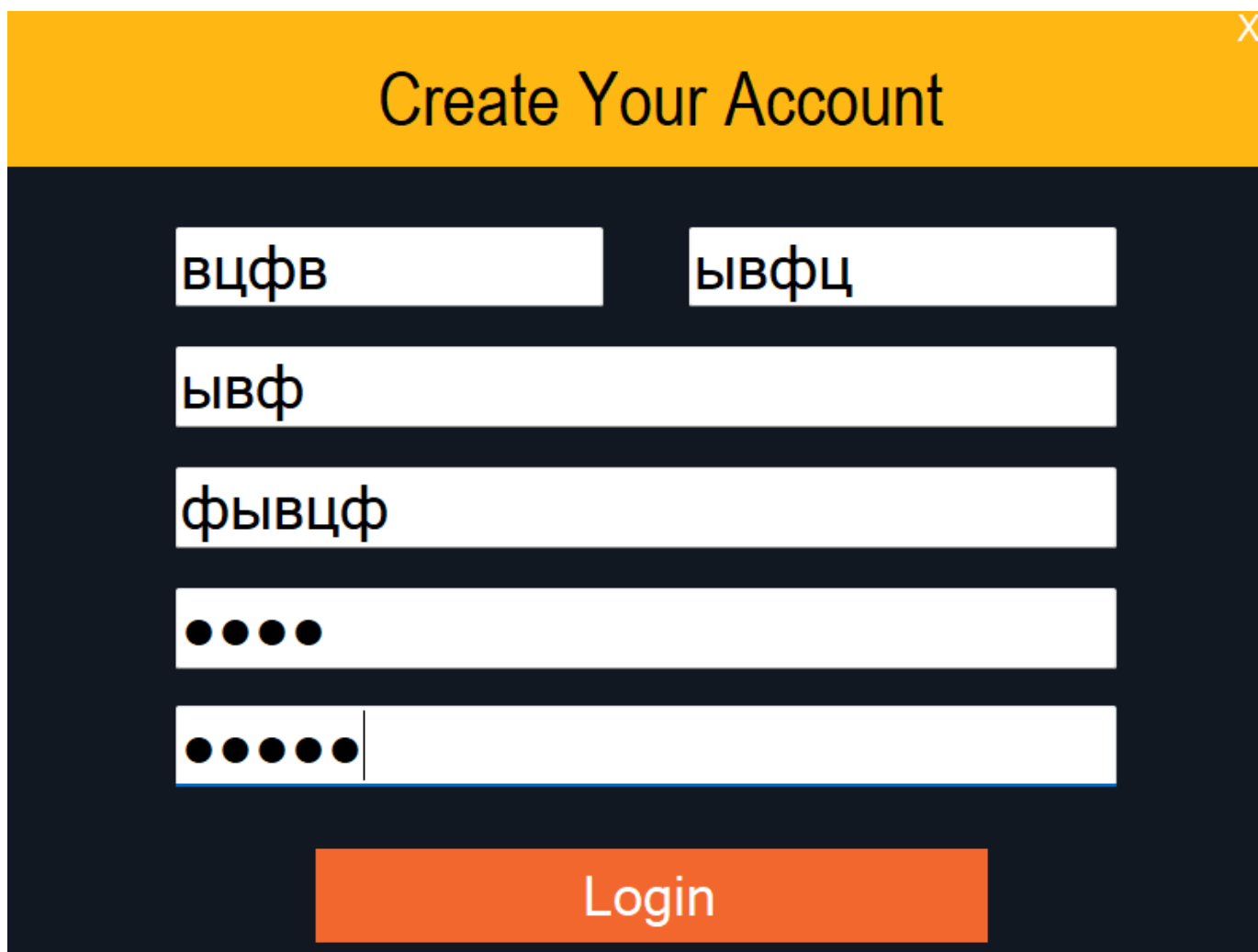
(Name)	<b>textBoxPassword</b>
AcceptsReturn	False

(Name)	<b>textBoxPasswordConfirm</b>
AcceptsReturn	False

Рисунок 66 – textBoxPasswordConfirm

(Name)	<b>buttonCreatAccount</b>
--------	---------------------------

Рисунок 67 – buttonCreateAccountx



The image shows a web form titled "Create Your Account" on a dark blue background. The form contains five input fields with the following text: "вцфв", "ывфц", "ывф", "фывцф", and a password field with five dots. Below the fields is an orange "Login" button. A small "X" icon is in the top right corner of the form area.

Рисунок 68 - Промежуточный результат

Для `textBoxPassword` и `textBoxPasswordConfirm` меняем свойство `UseSystemPassword`, чтобы в полях было видно, что написано, а не скрыто



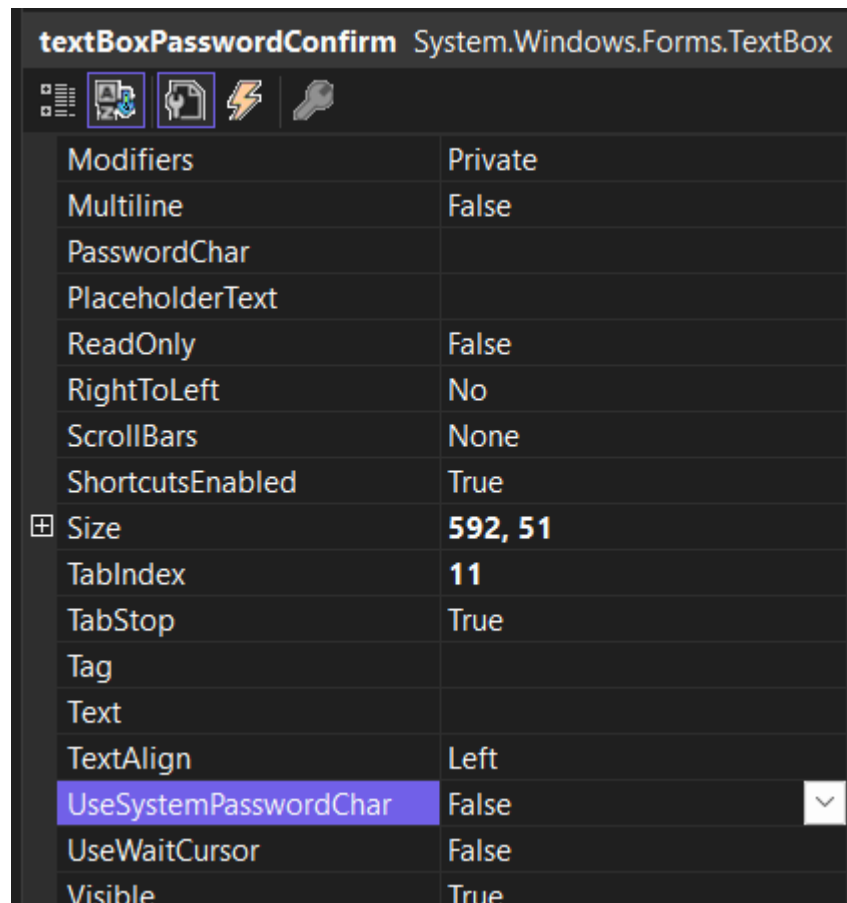


Рисунок 69 - UseSystemPassword

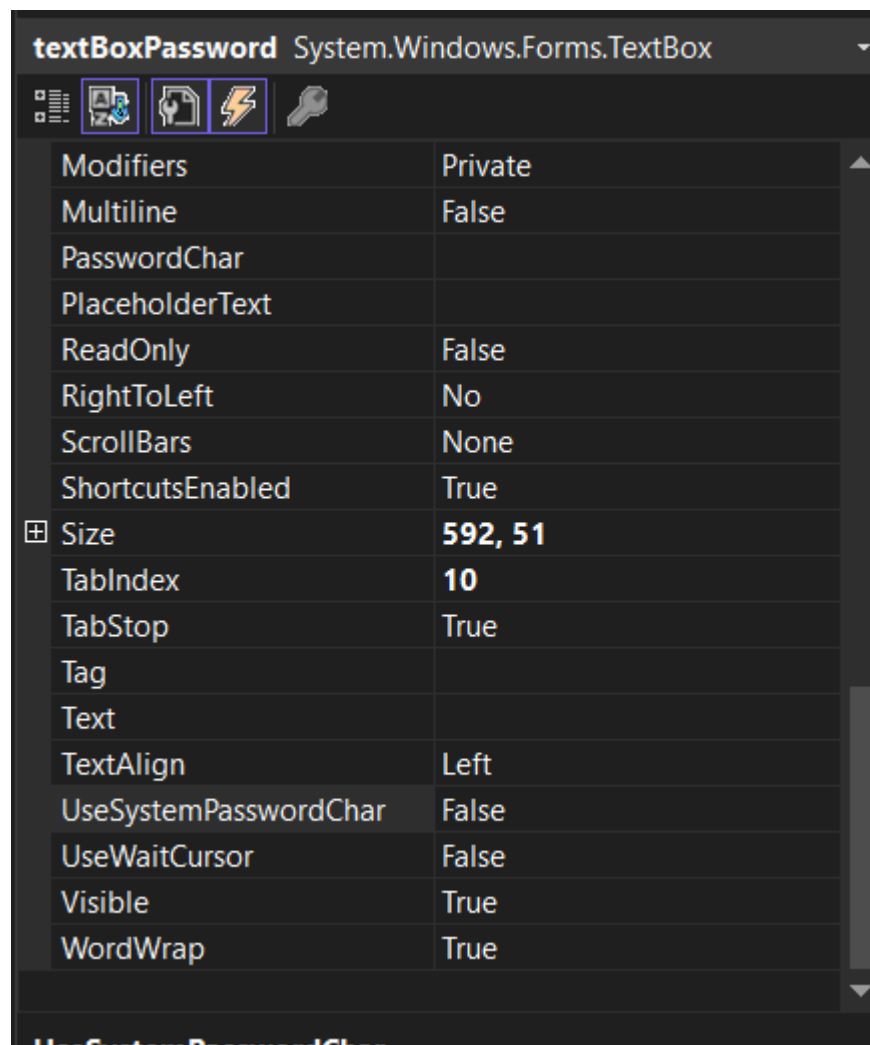


Рисунок 70 – Результат после изменения свойства `UseSystemPassword`

Для textBoxFirstName пишем функции, чтобы при нажатии на данное поле надпись скрывалась, если поле остается пустым, то надпись опять появляется

```
Ссылка: 0
private void textBoxFirstName_Enter(object sender, EventArgs e)
{
    String fname = textBoxFirstName.Text;
    if (fname.ToLower().Trim().Equals("first name"))
    {
        textBoxFirstName.Text = "";
        textBoxFirstName.ForeColor = Color.Black;
    }
}
```

Рисунок 71 - Функция textBoxFirstName\_Enter

```
Ссылка: 0
private void textBoxFirstName_Leave(object sender, EventArgs e)
{
    String fname = textBoxFirstName.Text;
    if (fname.ToLower().Trim().Equals("first name") || fname.Trim().Equals(""))
    {
        textBoxFirstName.Text = "first name";
        textBoxFirstName.ForeColor = Color.Gray;
    }
}
```

Рисунок 72 - Функция textBoxFirstName\_Leave

Чтобы при запуске текстовки не были активными пишем следующую функцию

```
Ссылка: 1
public RegisterForm()
{
    InitializeComponent();
    this.ActiveControl = this;
}
```

Рисунок 73 – Активный элемент

Для textBoxLastName пишем функции, чтобы при нажатии на данное поле надпись скрывалась, если поле остается пустым, то надпись опять появляется

```
Ссылки: 0
private void textBoxLastName_Enter(object sender, EventArgs e)
{
    string lname = textBoxLastName.Text;
    if(lname.ToLower().Trim().Equals("last name"))
    {
        textBoxLastName.Text = "";
        textBoxLastName.ForeColor = Color.Black;
    }
}

Ссылки: 0
private void textBoxLastName_Leav(object sender, EventArgs e)
{
    String lname = textBoxLastName.Text;
    if(lname.ToLower().Trim().Equals("last name") || lname.Trim().Equals(""))
    {
        textBoxLastName.Text = "";
        textBoxLastName.ForeColor = Color.Gray;
    }
}
}
```

Рисунок 74 – textBoxLastName

Для textBoxEmail пишем функции, чтобы при нажатии на данное поле надпись скрывалась, если поле остается пустым, то надпись опять появляется

```
Ссылка: 0
private void textBoxEmail_Enter(object sender, EventArgs e)
{
    string ename = textBoxEmail.Text;
    if (ename.ToLower().Trim().Equals("email address"))
    {
        textBoxEmail.Text = "";
        textBoxEmail.ForeColor = Color.Black;
    }
}

Ссылка: 0
private void textBoxEmail_Leav(object sender, EventArgs e)
{
    String ename = textBoxEmail.Text;
    if (ename.ToLower().Trim().Equals("email address") || ename.Trim().Equals(""))
    {
        textBoxEmail.Text = "email address";
        textBoxEmail.ForeColor = Color.Gray;
    }
}
}
```

Рисунок 75 – textBoxEmail

Для textBoxUsername пишем функции, чтобы при нажатии на данное поле надпись скрывалась, если поле остается пустым, то надпись опять появляется

```
Ссылка: 0
private void textBoxUserName_Enter(object sender, EventArgs e)
{
    string username = textBoxUserName.Text;
    if (username.ToLower().Trim().Equals("username"))
    {
        textBoxUserName.Text = "";
        textBoxUserName.ForeColor = Color.Black;
    }
}

Ссылка: 0
private void textBoxUserName_Leav(object sender, EventArgs e)
{
    String username = textBoxUserName.Text;
    if (username.ToLower().Trim().Equals("username") || username.Trim().Equals(""))
    {
        textBoxUserName.Text = "";
        textBoxUserName.ForeColor = Color.Gray;
    }
}
}
```

Рисунок 76 – textBoxUsername

Для textBoxPassword пишем функции, чтобы при нажатии на данное поле надпись скрывалась, если поле остается пустым, то надпись опять появляется

```

Ссылка: 0
private void textBoxPassword_Enter(object sender, EventArgs e)
{
    string password = textBoxPassword.Text;
    if (password.ToLower().Trim().Equals("password"))
    {
        textBoxPassword.Text = "";
        textBoxPassword.UseSystemPasswordChar = true;
        textBoxPassword.ForeColor = Color.Black;
    }
}

Ссылка: 0
private void textBoxPassword_Leav(object sender, EventArgs e)
{
    String password = textBoxPassword.Text;
    if (password.ToLower().Trim().Equals("password") || password.Trim().Equals(""))
    {
        textBoxPassword.Text = "password";
        textBoxPassword.UseSystemPasswordChar = false;
        textBoxPassword.ForeColor = Color.Gray;
    }
}

```

Рисунок 77 – textBoxPassword

Для textBoxPasswordConfirm пишем функции, чтобы при нажатии на данное поле надпись скрывалась, если поле остается пустым, то надпись опять появляется

```

Ссылка: 0
private void textBoxPasswordConfirm_Enter(object sender, EventArgs e)
{
    string cpassword = textBoxPasswordConfirm.Text;
    if (cpassword.ToLower().Trim().Equals("confirm password"))
    {
        textBoxPasswordConfirm.Text = "";
        textBoxPasswordConfirm.UseSystemPasswordChar = true;
        textBoxPasswordConfirm.ForeColor = Color.Black;
    }
}

Ссылка: 0
private void textBoxPasswordConfirm_Leav(object sender, EventArgs e)
{
    String cpassword = textBoxPasswordConfirm.Text;
    if (cpassword.ToLower().Trim().Equals("confirm password") ||
        cpassword.ToLower().Trim().Equals("password") ||
        cpassword.Trim().Equals(""))
    {
        textBoxPasswordConfirm.Text = "password";
        textBoxPasswordConfirm.UseSystemPasswordChar = false;
        textBoxPasswordConfirm.ForeColor = Color.Gray;
    }
}

```

Рисунок 78 – textBoxPasswordConfirm

Для label, который выполняет функцию закрытия приложения пишем следующие события, чтобы при нажатии на него оно закрывалось, а при наведении label менял цвет на черный

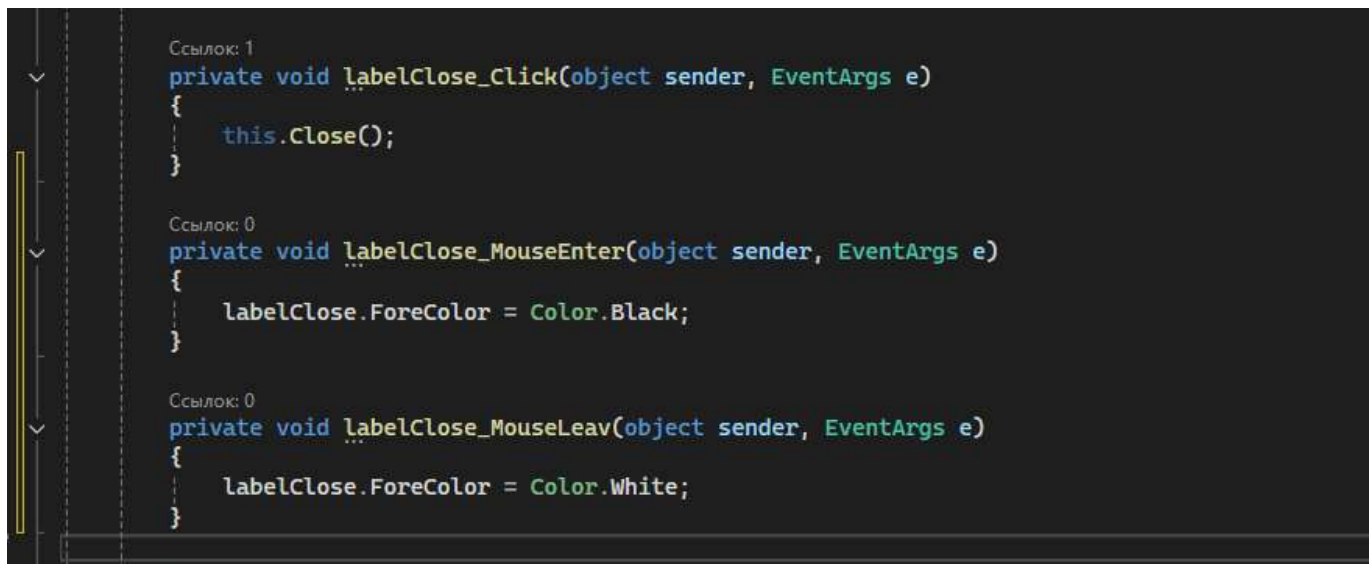


Рисунок 79 - События кнопки закрытия приложения

Убираем у формы рамки меняя свойство FormBorderStyle

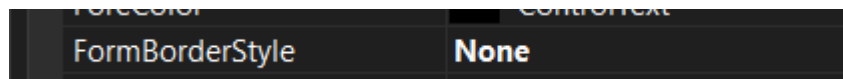


Рисунок 80 - FormBorderStyle



После выполнения всех действий получаем данный результат

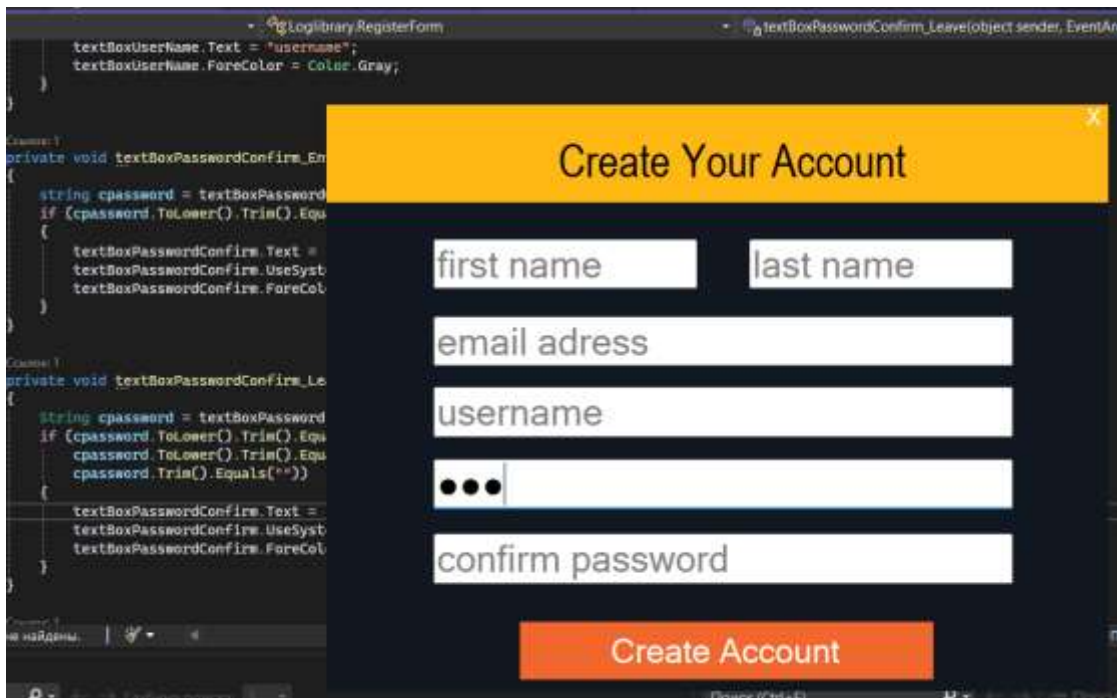


Рисунок 81 - Результат

Заходим в PhpMyAdmin и добавляем в таблице столбцы с атрибутами firstname, lastname и emailaddress

Рисунок 82 - Добавление столбцов в таблицу

Имя	Тип	Длина/Значения	По умолчанию	Сравнение
firstname	VARCHAR	100	Нет	
lastname	VARCHAR	100	Нет	
emailaddress	TEXT	100	Нет	

Рисунок 83 - Структура таблицы

В пространство имен подключаем MySQL

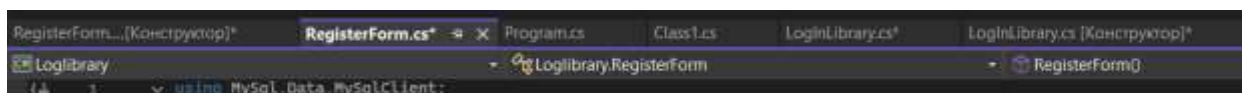


Рисунок 84 - Пространство имен

Для кнопки создание аккаунт пишем функцию, в которой данные с текстовых считываются и заносятся в бд и выводится сообщение «ACCOUNT CREATED»



```

COMMAND:
private void buttonLogin_Click(object sender, EventArgs e)
{
    DB db = new DB();
    MySqlCommand = new MySqlCommand("INSERT INTO 'users'('firstname', 'lastname', 'emailaddress', 'username', 'password') VALUES (@fn, @ln, @email, @un, @pass)", db.connection);
    command.Parameters.Add("@fn", MySqlDbType.VarChar).Value = textBoxFirstName.Text;
    command.Parameters.Add("@ln", MySqlDbType.VarChar).Value = textBoxLastName.Text;
    command.Parameters.Add("@email", MySqlDbType.VarChar).Value = textBoxEmail.Text;
    command.Parameters.Add("@un", MySqlDbType.VarChar).Value = textBoxUserName.Text;
    command.Parameters.Add("@pass", MySqlDbType.VarChar).Value = textBoxPassword.Text;

    db.openConnection();

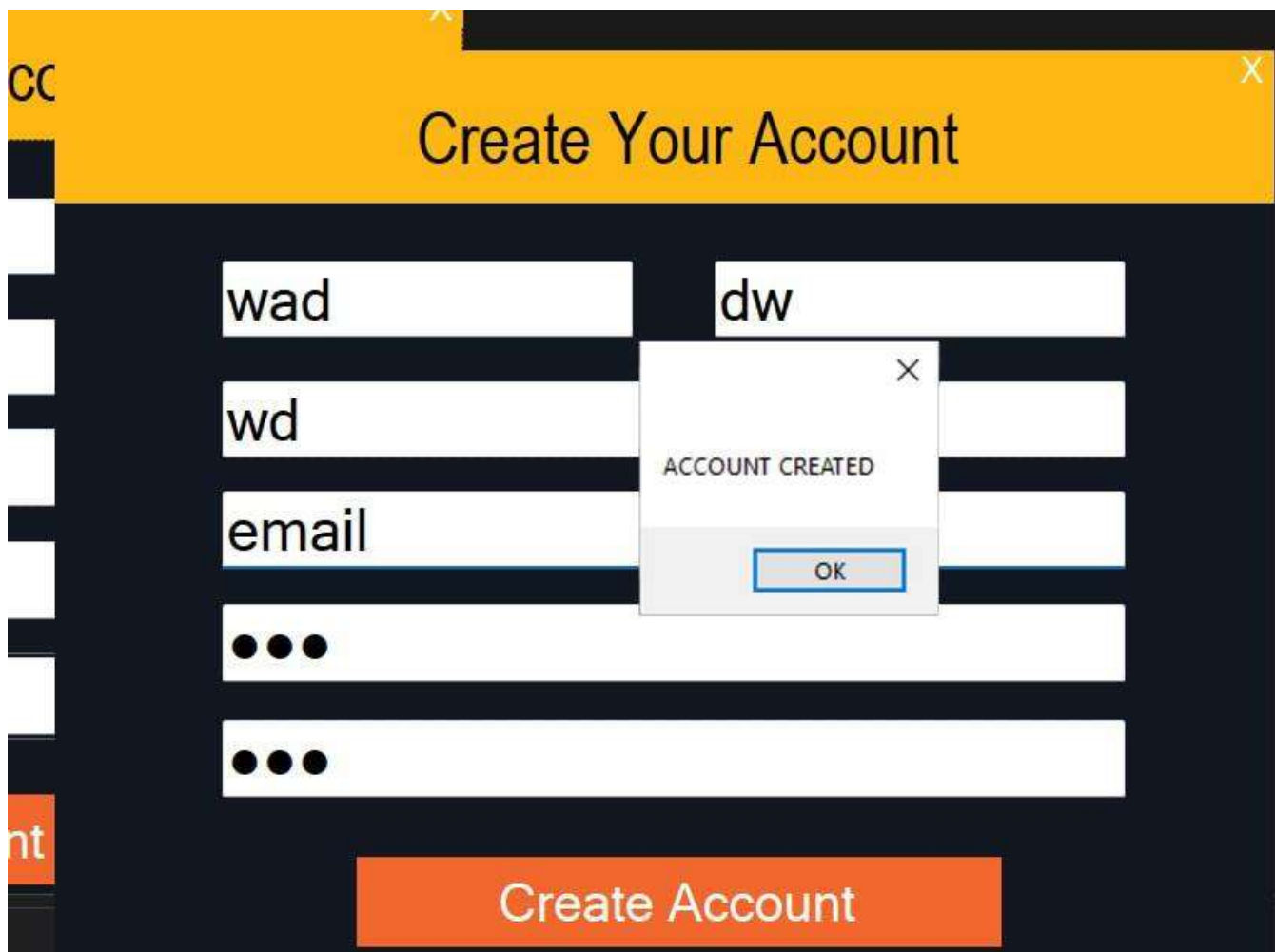
    if (command.ExecuteNonQuery() == 1)
    {
        MessageBox.Show("ACCOUNT CREATED");
    }
    else
    {
        MessageBox.Show("ERROR");
    }

    db.closeConnection();
}

```

Рисунок 85 - Функция кнопки создания аккаунта

Промежуточный результат



The screenshot shows a web form titled "Create Your Account" with a yellow header. The form has several input fields: "wad" (containing "wad"), "dw" (containing "dw"), "wd" (containing "wd"), "email" (containing "email"), and two password fields (both containing three dots). A modal dialog box is displayed in the center, titled "ACCOUNT CREATED" with an "OK" button. At the bottom of the form is a large orange button labeled "Create Account".

Рисунок 86 - Промежуточный результат



The screenshot shows a database table with the following columns: id, firstname, lastname, emailaddress, username, and password. The table contains two rows of data.

id	firstname	lastname	emailaddress	username	password
2	wad	dw	user	email	pas

Рисунок 87 - Данные в бд

Пишем событие, в котором будет проверяться username, чтобы невозможно было создать пользователя с таким username, который уже используется

```
Ссылка: 0
public Boolean checkUsername()
{
    DB db = new DB();
    String username = textBoxUserName.Text;
    DataTable table = new DataTable();
    MySqlDataAdapter adapter = new MySqlDataAdapter();
    MySqlCommand command = new MySqlCommand("SELECT * FROM 'users' WHERE 'username' = @usn", db.getConnection());
    command.Parameters.Add("@usn", MySqlDbType.VarChar).Value = username;
    adapter.SelectCommand = command;
    adapter.Fill(table);
    if (table.Rows.Count > 0)
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

Рисунок 88 - Функция checkUsername

Пишем условие, в котором определяется создан такой пользователь уже или нет, если не создан то выводим messagebox, с надписью «account created»

```
if (checkUsername())
{
    MessageBox.Show("This Username already exists, Select a different One");
}
else
{
    if (command.ExecuteNonQuery() = 1)
    {
        MessageBox.Show ("ACCOUNT CREATED");
    }
    else
    {
        MessageBox.Show ("ERROR");
    }
}
```

Рисунок 89 - Проверка, существует такой аккаунт или нет

Пишем функцию, в которой проверяются поля, пустые они или заполненные

```
Ссылка: 0
public Boolean checkTextBoxesValues()
{
    string fname = textBoxFirstName.Text;
    string lname = textBoxLastName.Text;
    string email = textBoxEmail.Text;
    string uname = textBoxUsername.Text;
    string pass = textBoxPassword.Text;

    if(fname.Equals("first name") || lname.Equals("last name") ||
        email.Equals("email address") || uname.Equals("username")
        || pass.Equals("password"))
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

Рисунок 90 - Проверка, пустые поля регистрации или нет

Проверка создания аккаунта с данными уже существующего пользователя

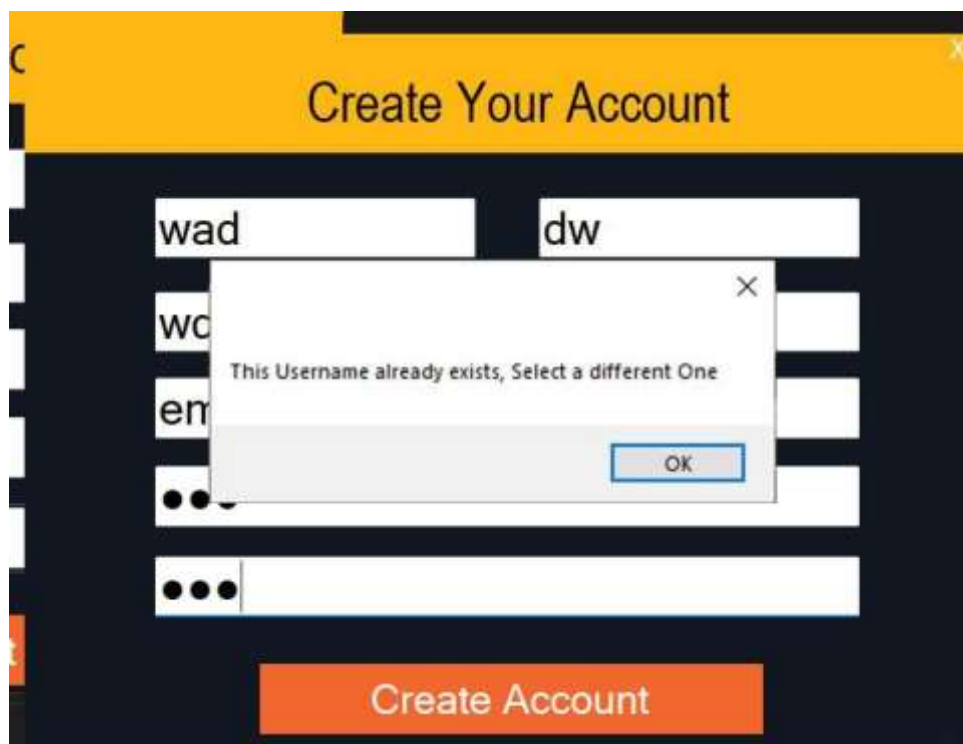


Рисунок 91 - Создание уже существующего пользователя

Проверка создания аккаунта с данными еще существующего пользователя

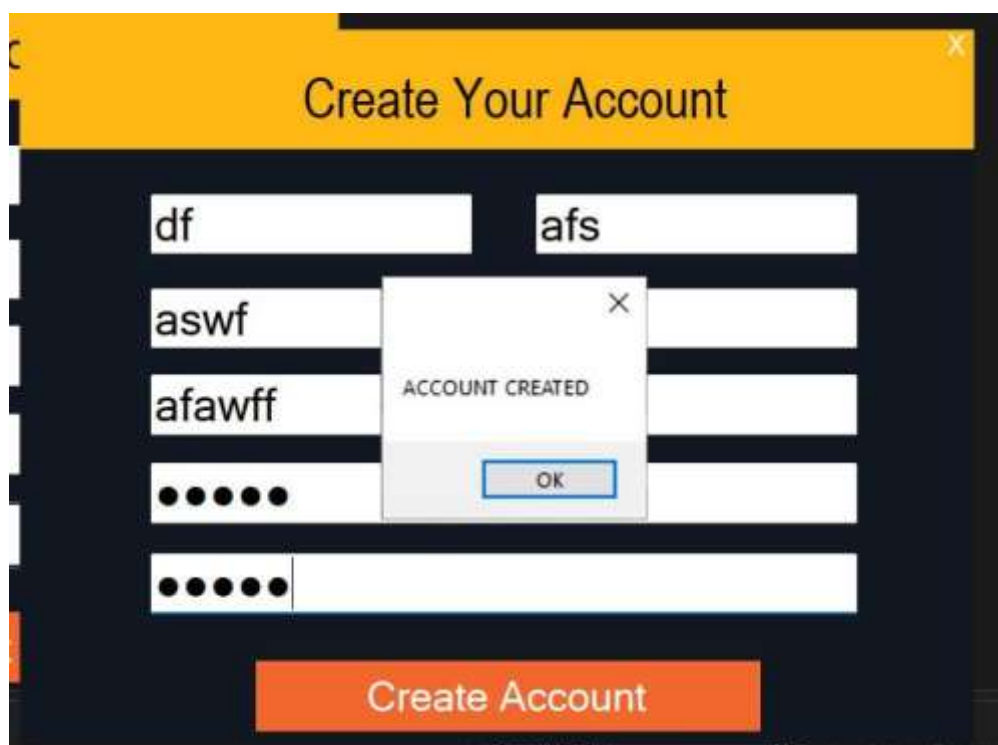


Рисунок 92 - Создание аккаунта

Добавление нового пользователя в базе данных

	id	firstname	lastname	emailaddress	username	password
<input type="checkbox"/>  	2	wad	dw	user	email	pass
<input type="checkbox"/>  	3	df	afs	aswf	afawff	pre12

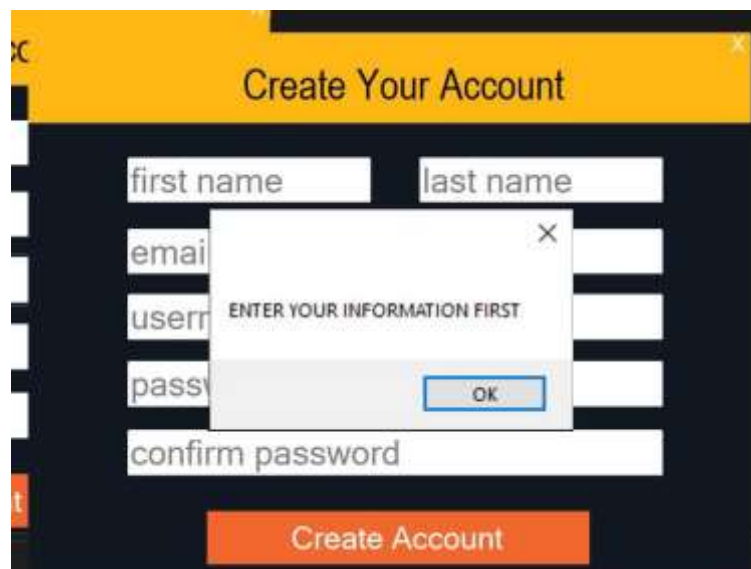
Рисунок 93 - Новый пользователь в бд

Пишем функцию, в которой проверяется введены какие-то данные пользователем или нет, если нет, то выводится мееседжбокс с сообщением «Enter your information first»

```
if(!checkTextBoxesValues())
{
    if (checkUsername())
    {
        MessageBox.Show("This Username already exists, Select a different One");
    }
    else
    {
        if (command.ExecuteNonQuery() == 1)
        {
            MessageBox.Show("ACCOUNT CREATED");
        }
        else
        {
            MessageBox.Show("ERROR");
        }
    }
}
else
{
    MessageBox.Show("ENTER YOUR INFORMATION FIRST");
}
```

Рисунок 94 - Вывод меседжбокса если поля пустые

Проверка регистрации аккаунта без введения данных



The image shows a web form titled "Create Your Account" with a yellow header. The form contains several input fields: "first name", "last name", "email", "username", "password", and "confirm password". At the bottom is a large orange "Create Account" button. A white modal dialog box is overlaid on the form, displaying the message "ENTER YOUR INFORMATION FIRST" and an "OK" button. This indicates that the form validation failed because some fields were empty.

Рисунок 95 - Создание аккаунта с пустыми полями

Пишем условие проверки паролей, чтобы при вводе подтверждения пароля оно совпадало с введенным паролем.

```

if (!checkTextBoxValues())
{
    if (textBoxPassword.Text.Equals(textBoxPasswordConfirm.Text))
    {
        if (checkUsername())
        {
            MessageBox.Show("This Username already exists, select a different one");
        }
        else
        {
            if (command.ExecuteNonQuery() == 1)
            {
                MessageBox.Show("ACCOUNT CREATED");
            }
            else
            {
                MessageBox.Show("ERROR");
            }
        }
    }
    else
    {
        MessageBox.Show("WRONG CONFIRMPASSWORD");
    }
}

```

Рисунок 96 - Проверка пароля

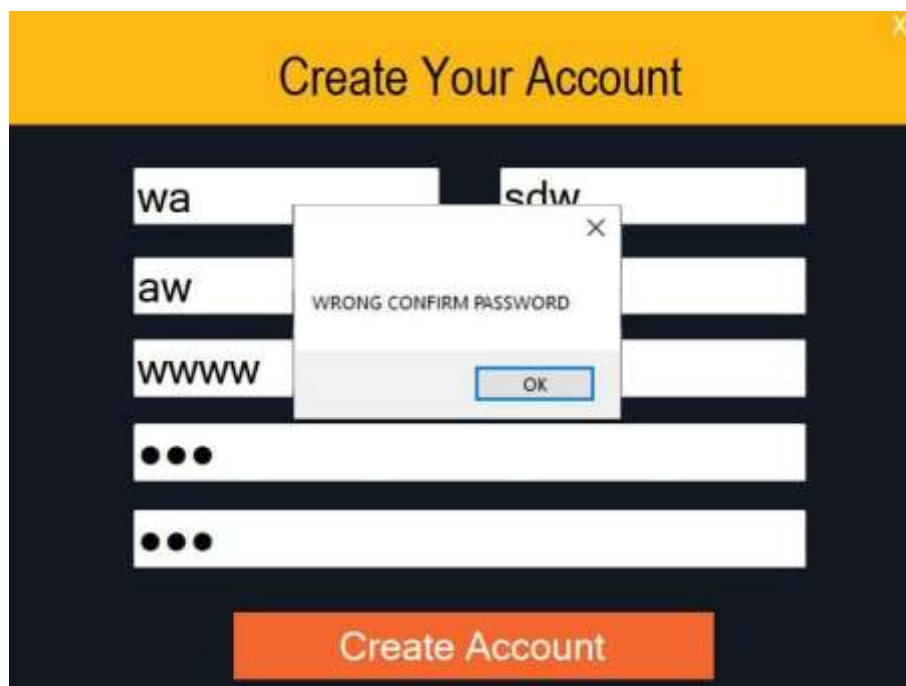


Рисунок 97 – Отрицательный результат проверки пароля



Рисунок 98 – Положительный результат проверки пароля

Добавляем в меседжбокс, который показывается при использовании существующего username кнопки «ОК», «Отмена» и изображение ошибки, а так же в тот, который показывается при создании аккаунта с изображением информации и кнопкой «ОК».

```

if(!checkTextBoxesValues())
{
    if(textBoxPassword.Text.Equals(textBoxPasswordConfirm.Text))
    {
        if (checkUsername())
        {
            MessageBox.Show("This Username already exists, Select a different One", "Duplicate Username", MessageBoxButtons.OKCancel, MessageBoxIcon.Error);
        }
        else
        {
            if (command.ExecuteNonQuery() == 1)
            {
                MessageBox.Show("Your Account Has Been Created", "Account", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
            else
            {
                MessageBox.Show("ERROR");
            }
        }
    }
    else
    {
        MessageBox.Show("WRONG CONFIRM PASSWORD");
    }
}

```

Рисунок 99 - Доработка MessageBox



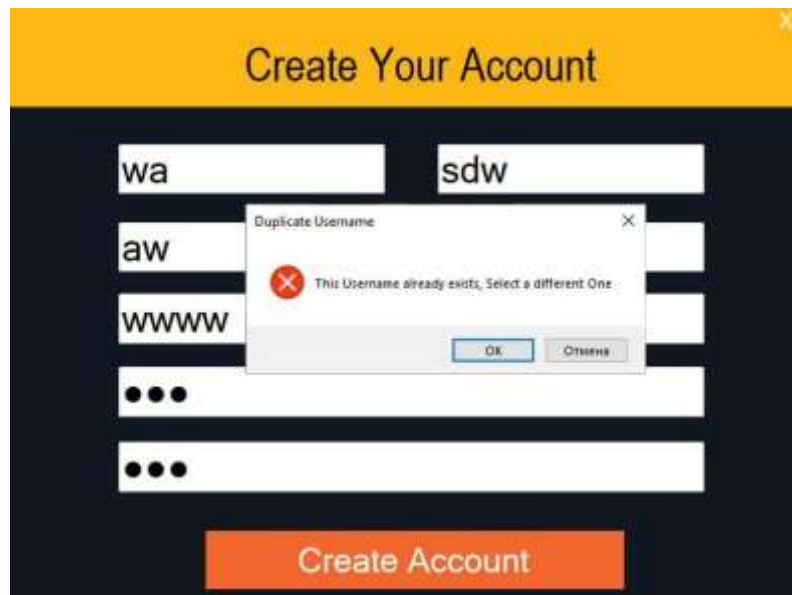


Рисунок 100 - Готовые MessageBox с ошибкой

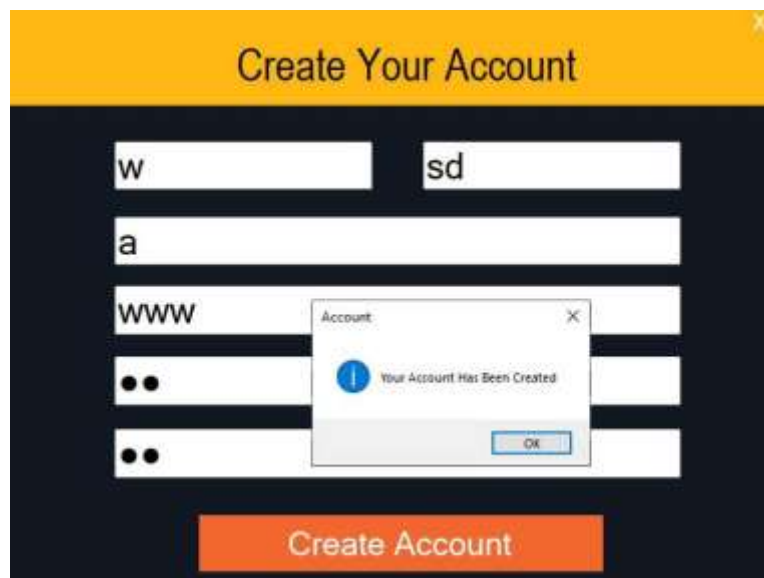


Рисунок 101 - Готовые MessageBox без ошибки

Добавляем в месседжбокс, который показывается при вводе в пароль и подтверждения пароля разные данные кнопки «ОК», «Отмена» и изображение ошибки.

```
else
{
    MessageBox.Show("Wrong Confirm Password", "Password Error", MessageBoxButtons.OKCancel, MessageBoxIcon.Error);
}
```

Рисунок 102 - Доработка MessageBox



Рисунок 103 – Готовый MessageBox с ошибкой

Добавляем в месседжбокс, который показывается при оставлении всех полей воода пустыми кнопки «ОК», «Отмена» и изображение ошибки.

```
else
{
    MessageBox.Show("Enter Your Informations First", "Empty Data", MessageBoxButtons.OKCancel, MessageBoxIcon.Error);
}
```

Рисунок 104 - Доработка MessageBox



Рисунок 105 - Готовый MessageBox с ошибкой

Так же добавляем в мессджбоксы на LoginForm кнопки «OK» и изображение ошибки.

```
DB db = new DB();
String username = textBoxUserName.Text;
String password = textBoxPassword.Text;

DataTable table = new DataTable();
MySQLDataAdapter adapter = new MySQLDataAdapter();
MySQLCommand command = new MySQLCommand("SELECT * FROM 'users' WHERE 'username' = @usn and 'password' = @pass", db.GetConnection());

command.Parameters.Add("@usn", MySQLDbType.VarChar).Value = username;
command.Parameters.Add("@pass", MySQLDbType.VarChar).Value = password;

adapter.SelectCommand = command;
adapter.Fill(table);

if (table.Rows.Count > 0)
{
    MessageBox.Show("YES");
}
else
{
    if (username.Trim().Equals(""))
    {
        MessageBox.Show("Enter Your Username To Login", "Empty Username", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    else if (password.Trim().Equals(""))
    {
        MessageBox.Show("Enter Your Password To Login", "Empty Password", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    else
    {
        MessageBox.Show("Wrong Username Or Password", "Wrong Data", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
}
```

Рисунок 106 - Доработка MessageBox

Создаем новую форму – MainForm и копируем в нее все элементы с RegisterForm.

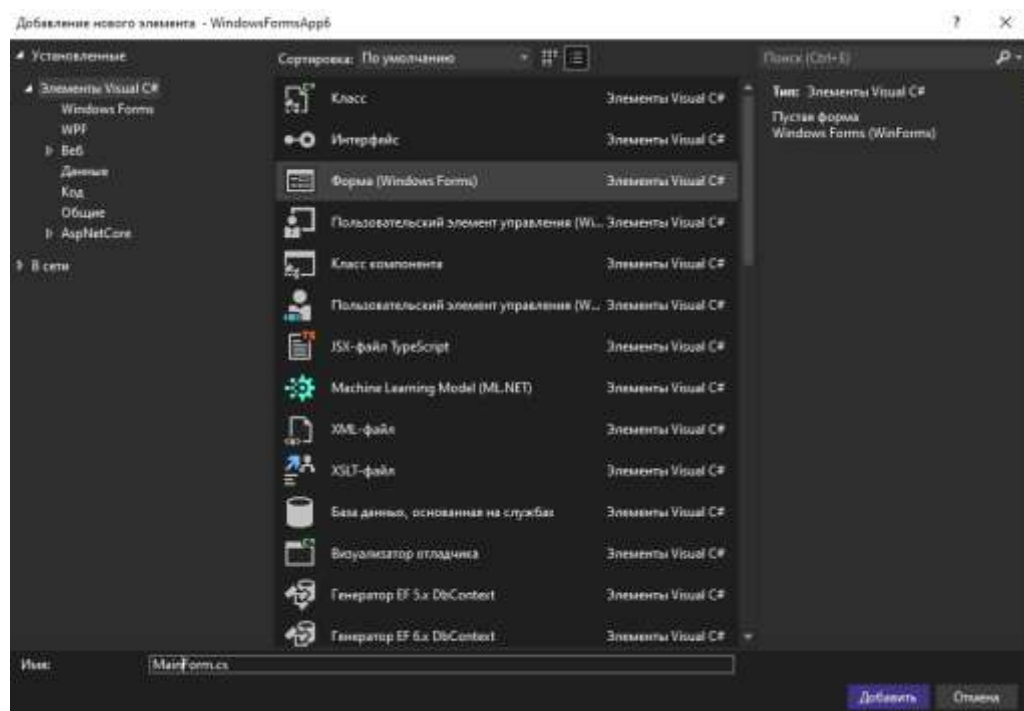


Рисунок 107 - Создание новой формы

Рисунок 108 - Новая форма

Удаляем с нее некоторые элементы и меняем текст в label1, задаем форме свойства, чтобы она открывалась по середине экрана.

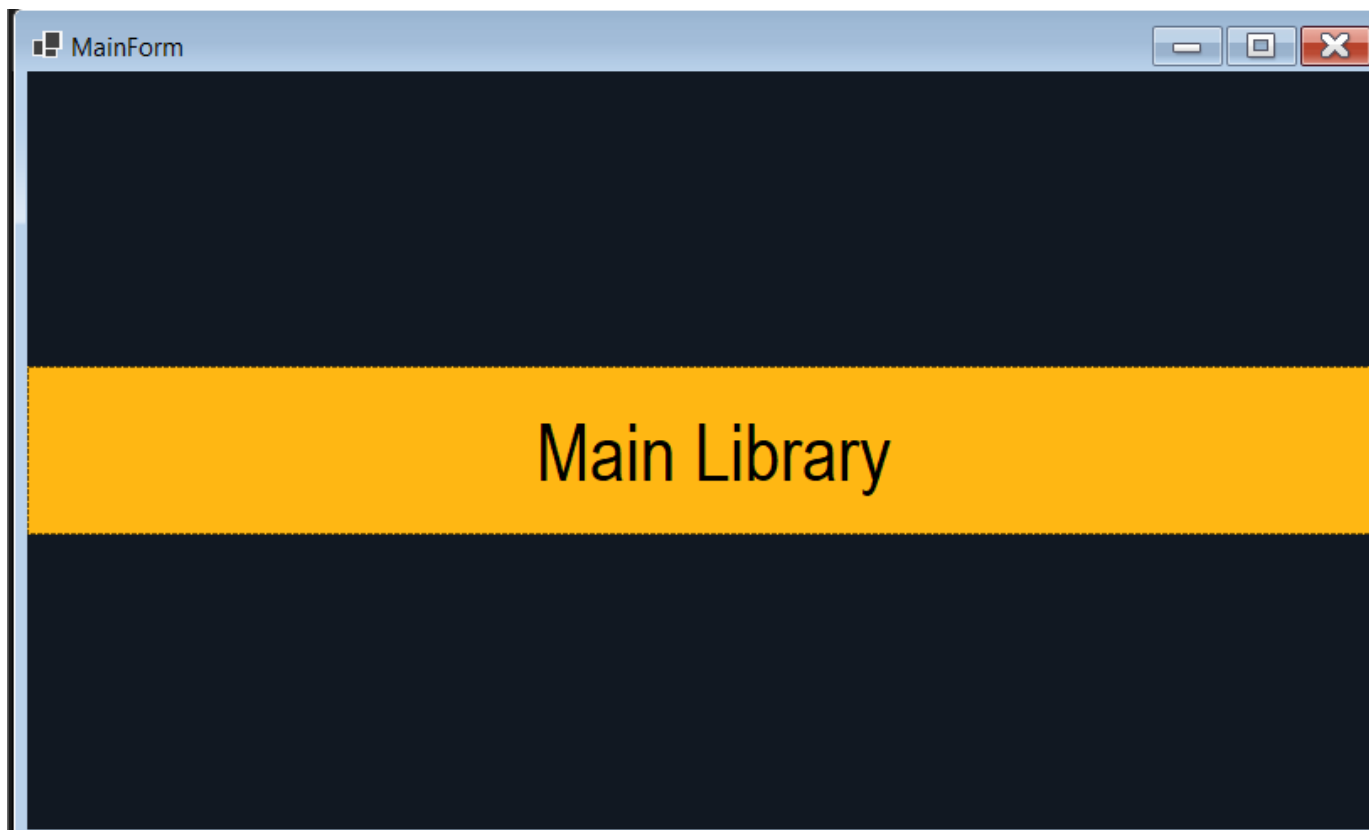


Рисунок 109 - Отредактированная форма

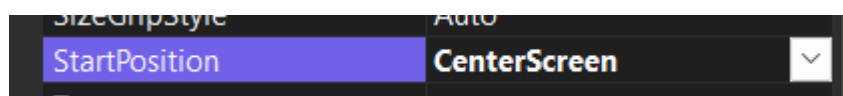


Рисунок 110 - Свойство StartPosition

В форму LoginForm добавляем label с текстом Don't have account? Sign up.

Text Don't Have an Account

Рисунок 111 - Текст label

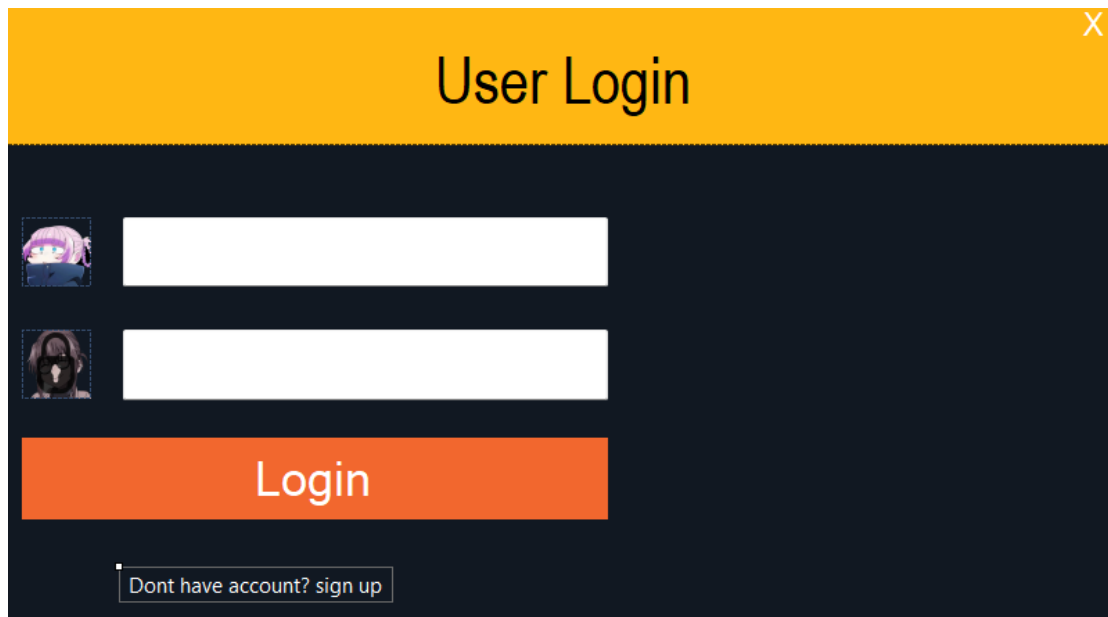
A screenshot of a 'User Login' form. The title 'User Login' is centered at the top in a large, bold, black font. Below the title are two input fields for username and password, each preceded by a small profile picture icon. A large, orange 'Login' button is positioned below the input fields. At the bottom, there is a link that says 'Don't have account? sign up'.

Рисунок 112 - Отредактированная LoginForm

Для созданного label пишем событие на клик, чтобы открывалась форма RegisterForm.

```
private void label2_Click(object sender, EventArgs e)
{
    this.Hide();
    RegisterForm registerForm = new RegisterForm();
    registerForm.Show();
}
```

Рисунок 113 - Событие открытия RegisterForm

Заходим в Program.cs и меняем открывающуюся форму на LoginForm.

```
// To customize application configuration such as set high DPI settings or default font,
// see https://aka.ms/applicationconfiguration.
ApplicationConfiguration.Initialize();
Application.Run(new LoginLibrary());
```

Рисунок 114 - Program.cs

В Register form на label, который отвечает за закрытие приложения пишем Application.Exit(), чтобы оно действительно закрывалось полностью, а не просто скрывалась форма.

```

Ссылка 1
private void labelClose_Click(object sender, EventArgs e)
{
    //this.Close();
    Application.Exit();
}

```

Рисунок 115 - Событие закрытия приложения

На всех формах у кнопок и label, которые выполняют роль кнопки, меняем свойство cursor на hand, чтобы при наведении на них курсор менялся на руку.



Рисунок 116 - Свойство Cursor

Пишем события для labelGoSingUp, чтобы в спокойном состоянии тест был белого цвета, а при наведении желтого

```

Ссылка 1
private void label2_MouseEnter(object sender, EventArgs e)
{
    label2.ForeColor = Color.Yellow;
}

Ссылка 1
private void label2_MouseLeave(object sender, EventArgs e)
{
    label2.ForeColor = Color.White;
}

```

Рисунок 117 - Событие labelGoSingUp

В настройках шрифта у labelGoSingUp ставим галочку напротив «Подчеркнутый»

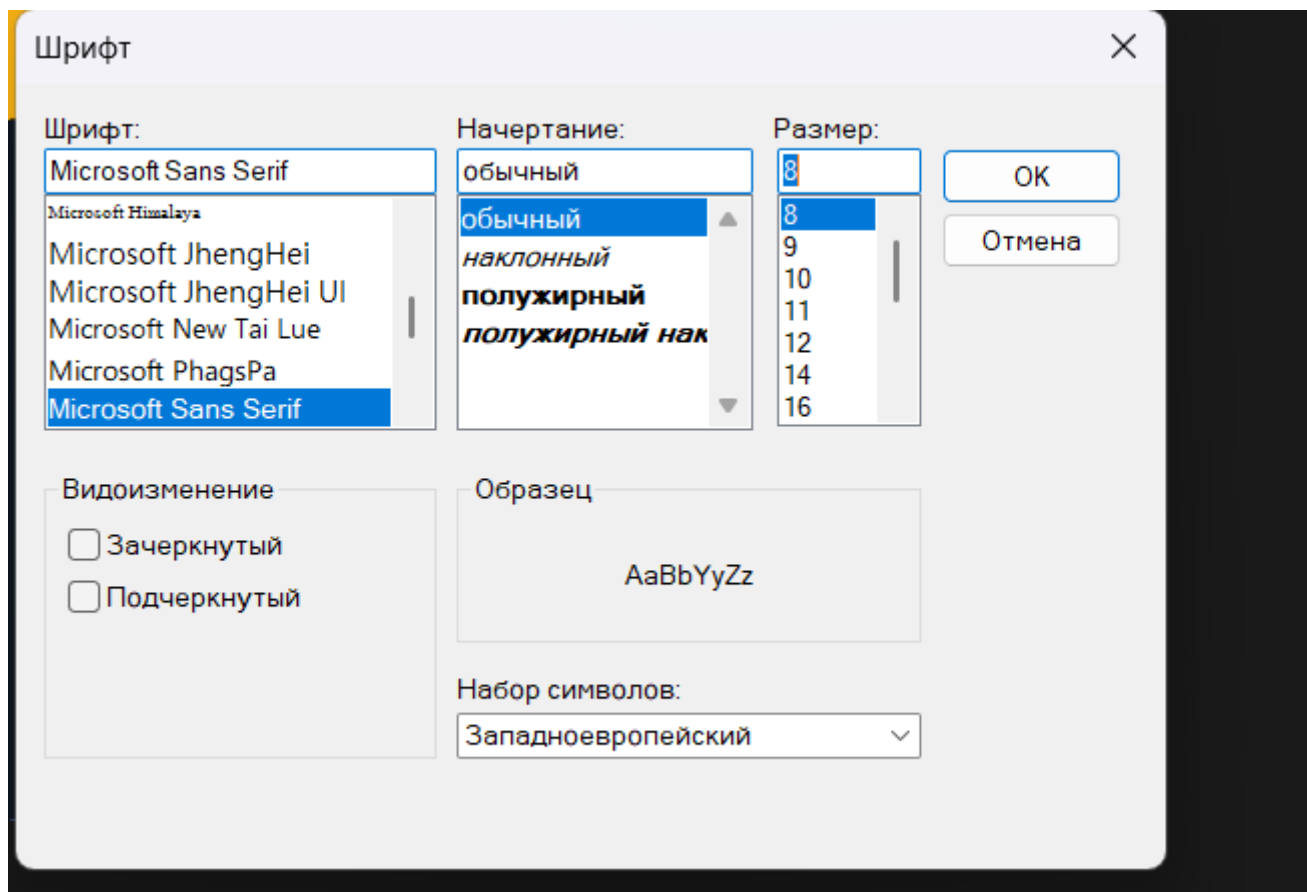


Рисунок 118 - Настройки шрифта

Копируем этот label на форму RegisterForm, меняем у него название и текст

Рисунок 119 - RegisterForm

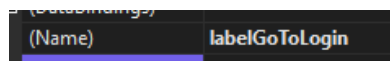


Рисунок 120 - Название label

Пишем события для него, чтобы в спокойном состоянии тест был белого цвета, а при наведении желтого, а также чтобы при клике на него открывалась форма LoginForm.

Рисунок 121 - Событие смены цвета при наведении на label

```
Ссылка: 1
private void label2_MouseEnter(object sender, EventArgs e)
{
    label2.ForeColor = Color.Yellow;
}

Ссылка: 1
private void label2_MouseLeave(object sender, EventArgs e)
{
    label2.ForeColor = Color.White;
}

Ссылка: 1
private void label2_Click(object sender, EventArgs e)
{
    this.Hide();
    LoginLibrary loginLibrary = new LoginLibrary();
    loginLibrary.ShowDialog();
}
```

Рисунок 122 - Событие открытия LoginForm

В LoginForm на label, который отвечает за закрытие, приложение пишем Application.Exit(), чтобы оно действительно закрывалось полностью, а не просто скрывалась форма.

```
Ссылка: 1
private void labelClose_Click(object sender, EventArgs e)
{
    //this.Close();
    Application.Exit();
}
```

Рисунок 123 - Событие закрытия приложения

В событие проверки правильности ввода username и password пишем, открытие MainForm.



```

        if (table.Rows.Count > 0)
        {
            this.Hide();
            LogInLibrary logInLibrary = new LogInLibrary();
            logInLibrary.Show();
        }
    }

```

Рисунок 124 - Открытие MainForm

В MainForm пишем событие закрытия приложения.

```

Ссылка 1
private void MainForm_FormClosing(object sender, FormClosingEventArgs e)
{
    Application.Exit();
}

```

Рисунок 125 - Событие закрытия приложения

В Login form при проверке введения данных в текстовые поля меняем username на password

```

else if (password.Trim().Equals(""))
{
    MessageBox.Show("Enter Your Password to Login", "Empty Password", MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}

```

Рисунок 126 - Проверка введения данных в текстовые поля

## Определение:

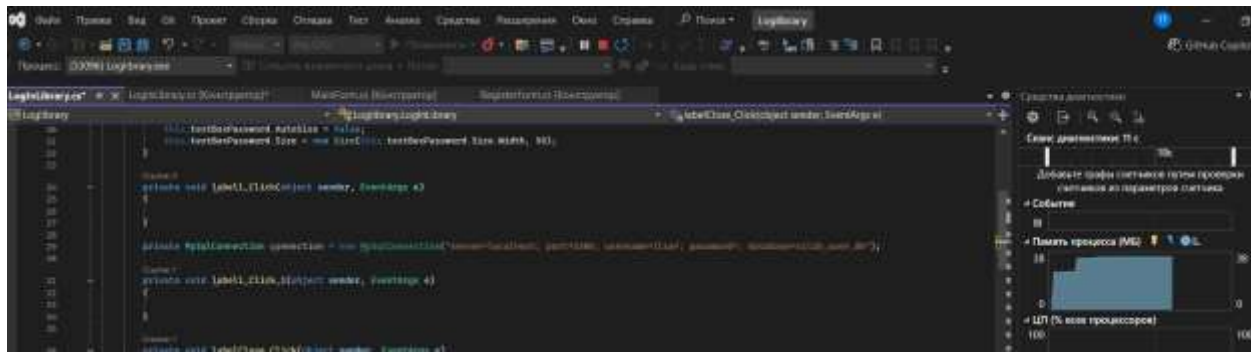
Отладка программных модулей - это процесс поиска и устранения ошибок в коде, чтобы программа работала корректно.

## Цель:

Главная цель отладки - сделать программу надежной и безошибочной.

## Задание:

1. **Воспроизведение ошибки:** Запустите программу и создайте условия, при которых возникает ошибка.



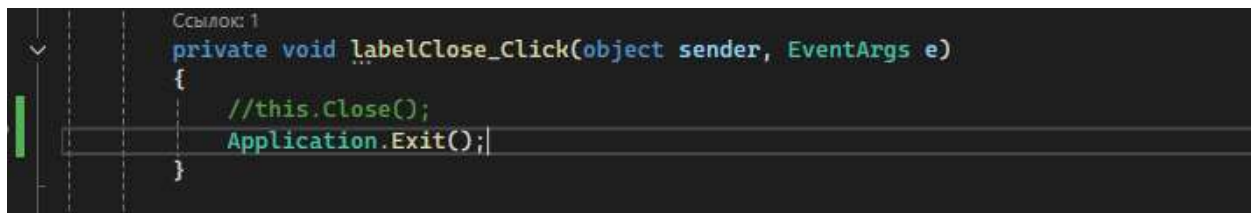
Программа не закрывается полностью при нажатии на крестик

2. **Анализ ошибки:** Определите тип ошибки (синтаксическая, логическая, ошибка времени выполнения) и место ее возникновения.

Это синтаксическая ошибка в коде

3. **Исправление ошибки:** Внесите необходимые изменения в код, чтобы устранить ошибку.

Исправление при помощи добавления `Application.Exit()`:



4. **Тестирование:** Проверьте, решена ли проблема, и не появились ли новые ошибки.

Данная ошибка была исправлена

5. **Повтор:** Повторяйте шаги 1-4, пока не будут найдены и исправлены все ошибки.

Все ошибки исправлены

## Инструменты:

Существует множество инструментов для отладки, как встроенных в среды разработки, так и сторонних.

## Распространенные инструменты:

- **Отладчики:** Позволяют пошагово выполнять код, просматривать значения переменных и останавливаться в определенных точках.
- **Анализаторы статического кода:** Выявляют потенциальные проблемы в коде до его компиляции.
- **Тестовые фреймворки:** Автоматизируют процесс тестирования и помогают найти ошибки.

#### **Методы:**

- **Пошаговая отладка:** Выполнение программы пошагово с остановками в определенных точках.
- **Установка точек останова:** Остановка программы в определенных местах для проверки значений переменных и состояния программы.
- **Использование трассировки:** Запись информации о выполнении программы для последующего анализа.
- **Профилирование:** Измерение производительности программы для выявления узких мест.

#### **Навыки:**

- **Чтение кода:** Умение понимать код, написанный на языке программирования.
- **Логическое мышление:** Умение анализировать код и находить причины ошибок.
- **Внимательность:** Умение концентрироваться на деталях и не упускать из виду важные моменты.
- **Настойчивость:** Умение не сдаваться и продолжать поиск ошибок, даже если это сложно.