

1. Создание проекта

Приложение Windows Forms (.NET Framework) C# Windows Рабочий стол

Имя проекта

Lab1leskina1.S.D.VD50-1-20

Расположение

C:\Users\Admin\source\repos

Имя решения ⓘ

Lab1leskina1.S.D.VD50-1-20

☐ Поместить решение и проект в одном каталоге

Платформа

.NET Framework 4.7.2

Проект будет создан в "C:\Users\Admin\source\repos\Lab1leskina1.S.D.VD50-1-20\Lab1leskina1.S.D.VD50-1-20\"

Рисунок 1 - Создание проекта

2. Созданный проект

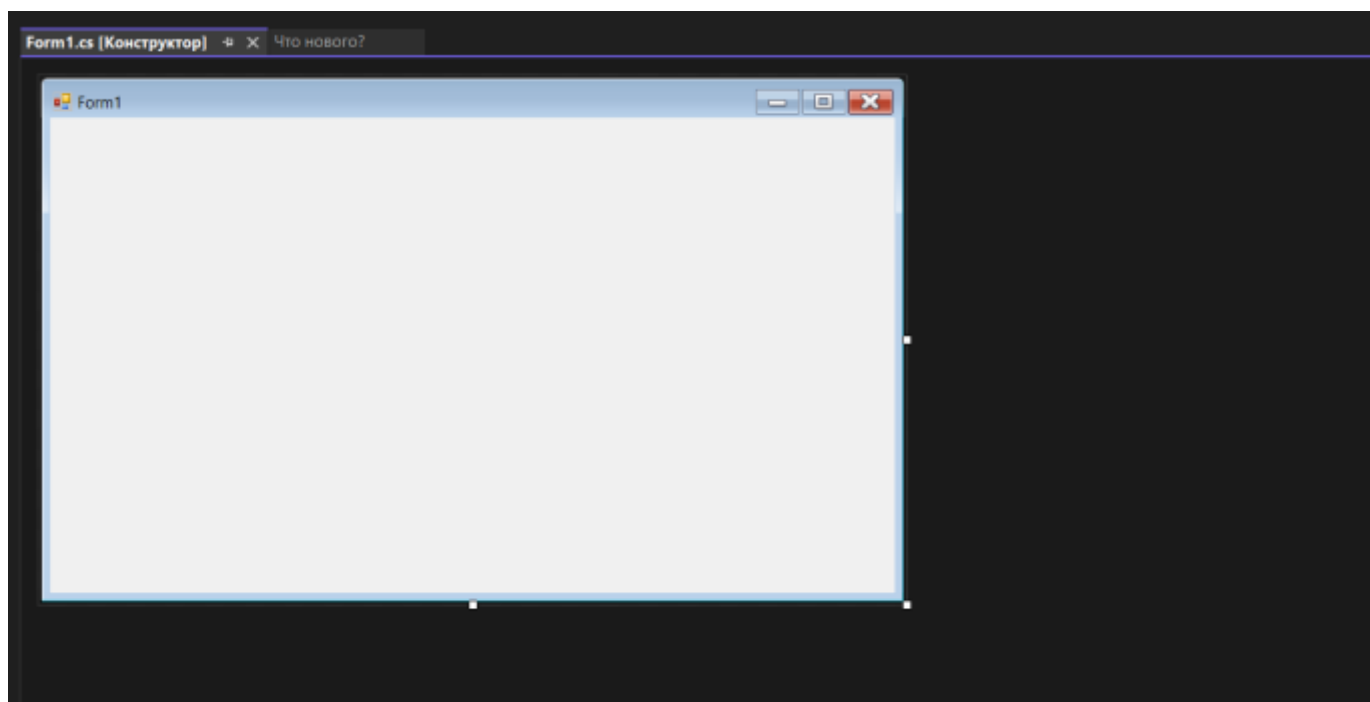


Рисунок 2 - Созданный проект

3. Удаление Form1

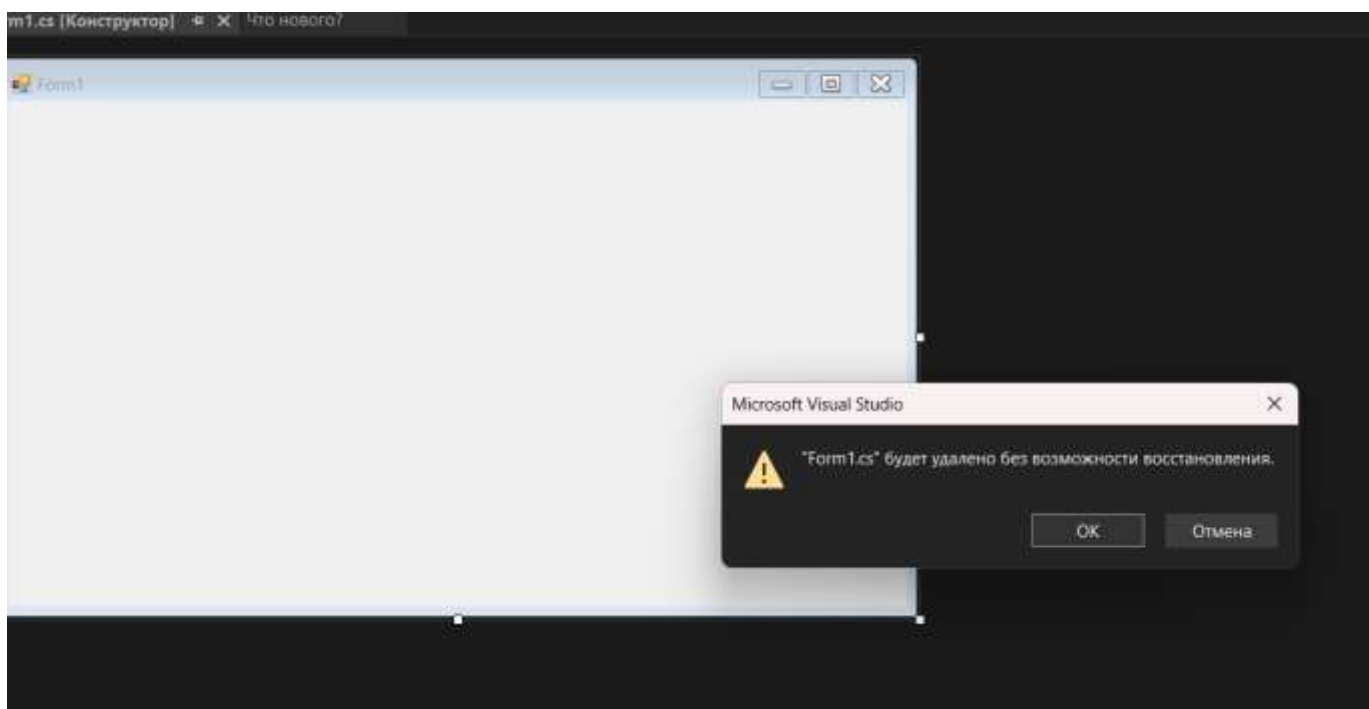


Рисунок 3 - Удаление формы

4. Создание LoginForm

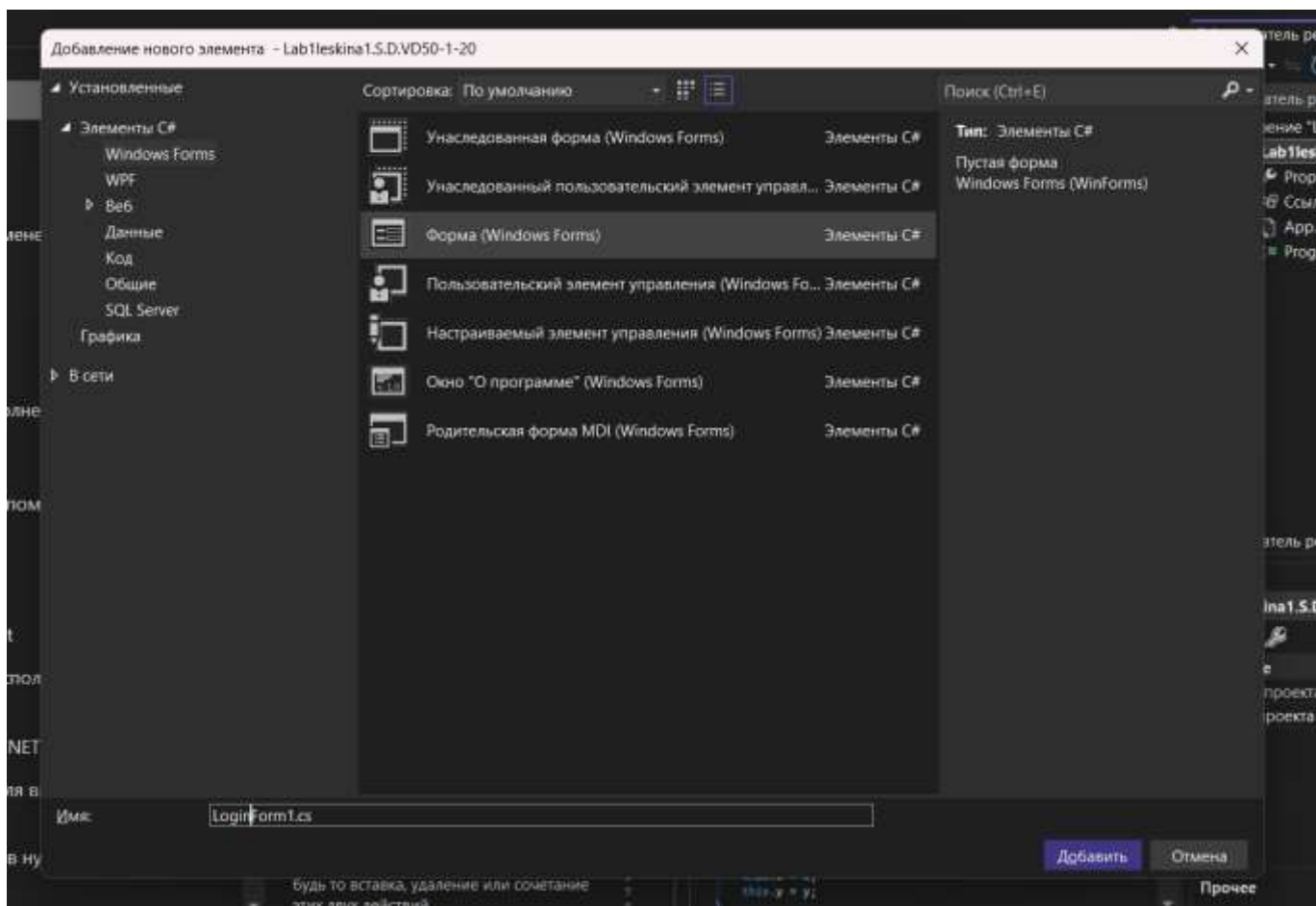


Рисунок 4 - Создание формы

5. Открываем панель элементов

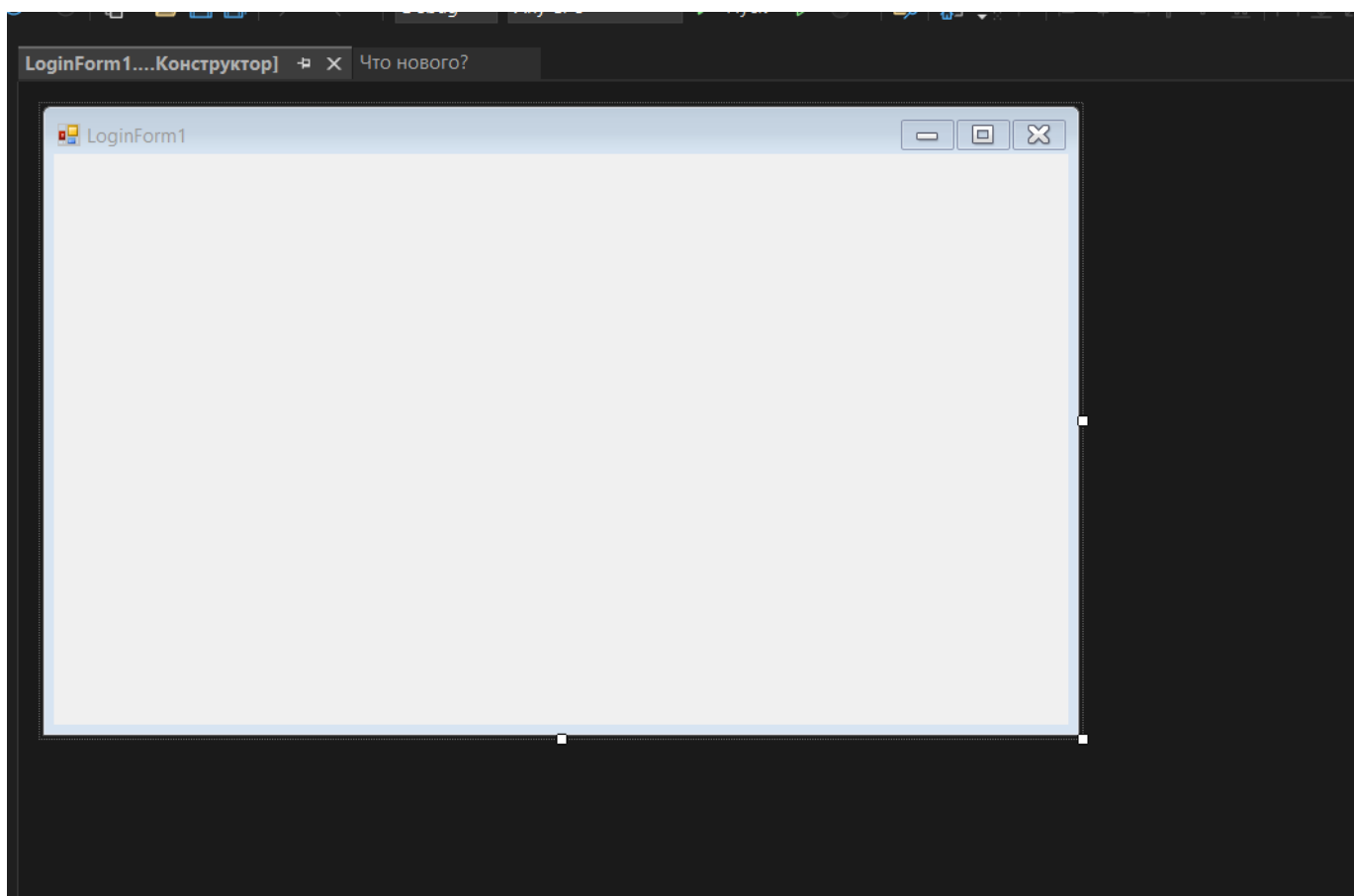


Рисунок 5 - Панель элементов

6. Выбираем в панели элементов элемент panel

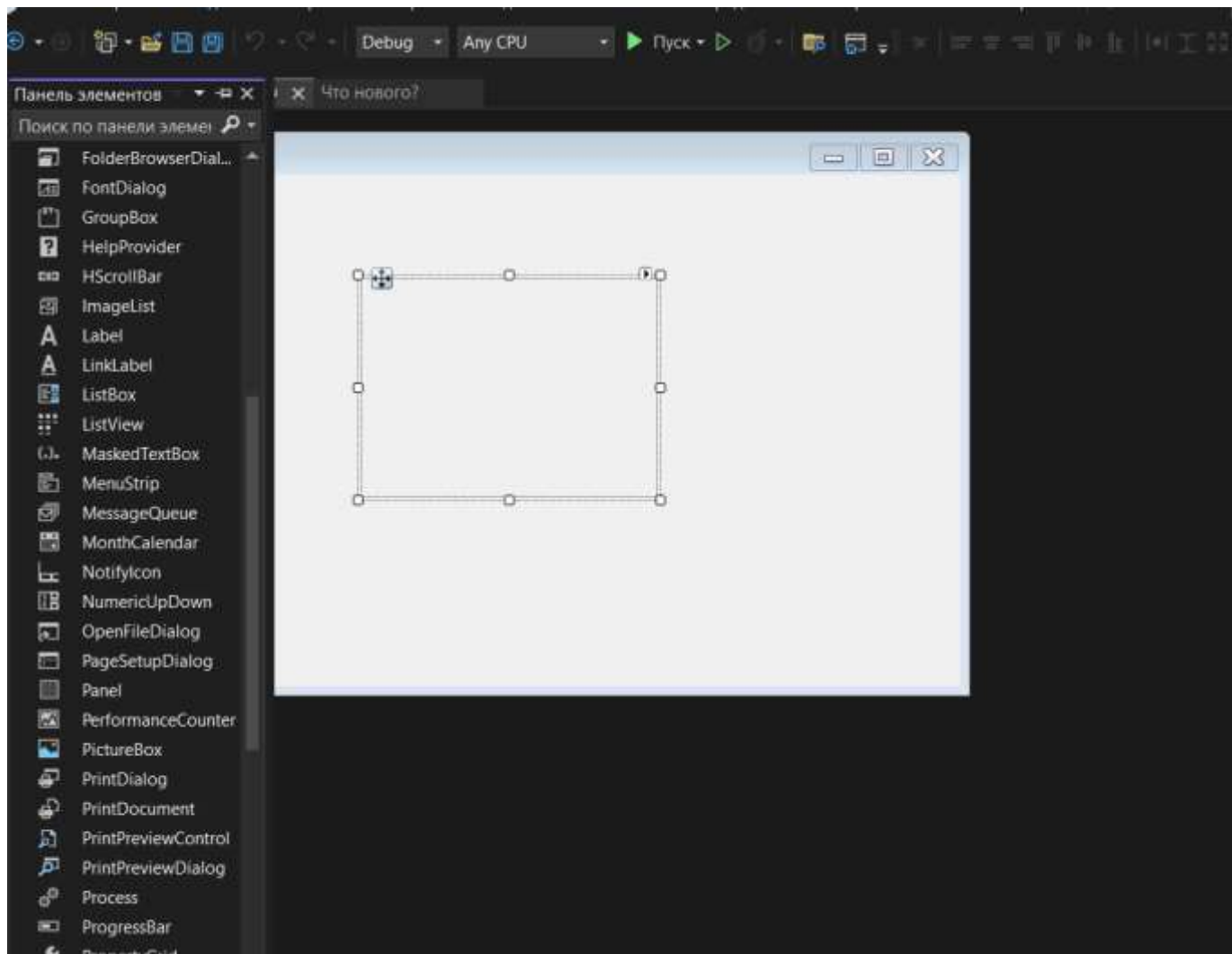


Рисунок 6 - Элемент panel

7. Закрепляем panel в родительском контейнере

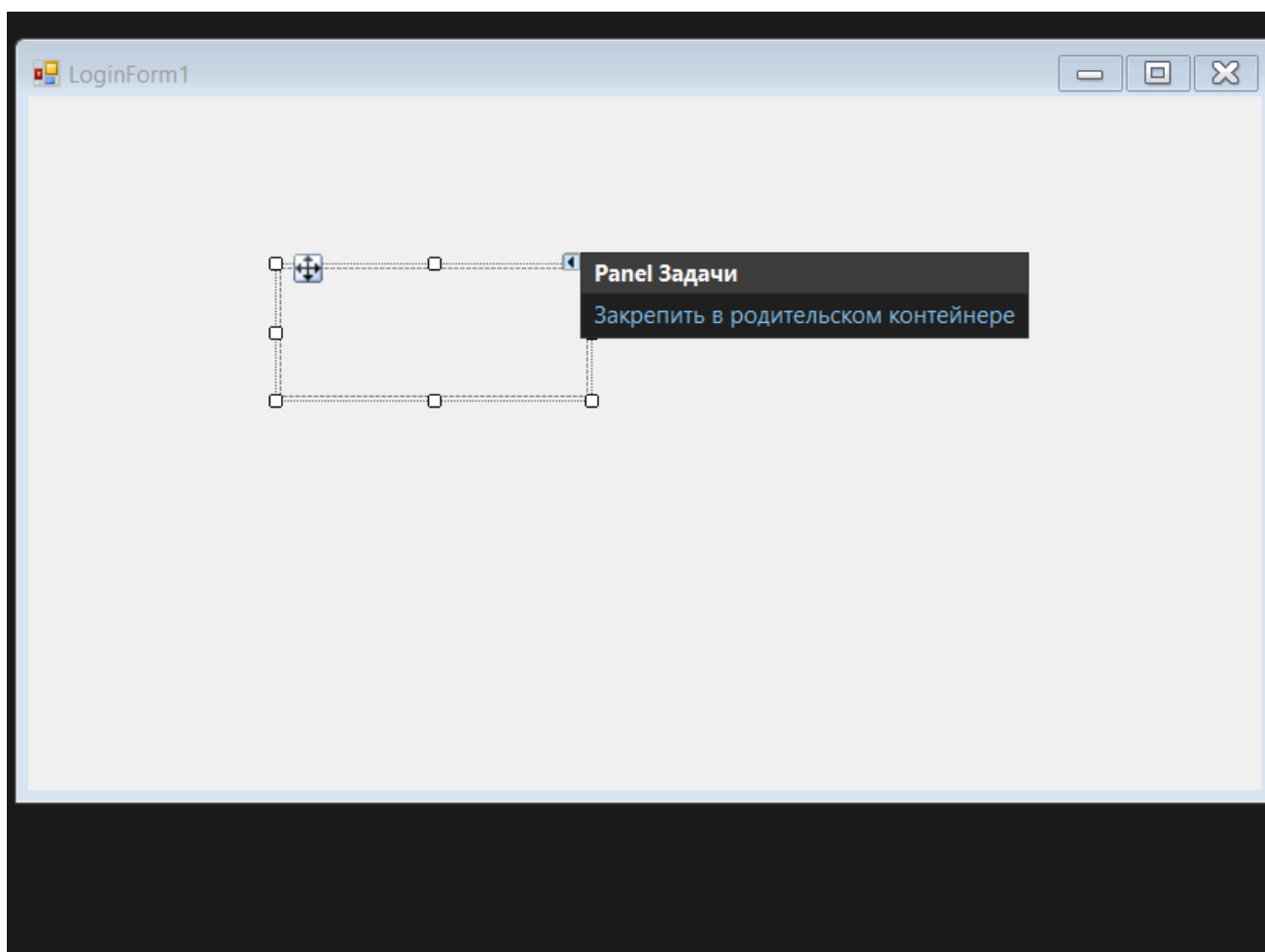


Рисунок 7 – Закрепление panel

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

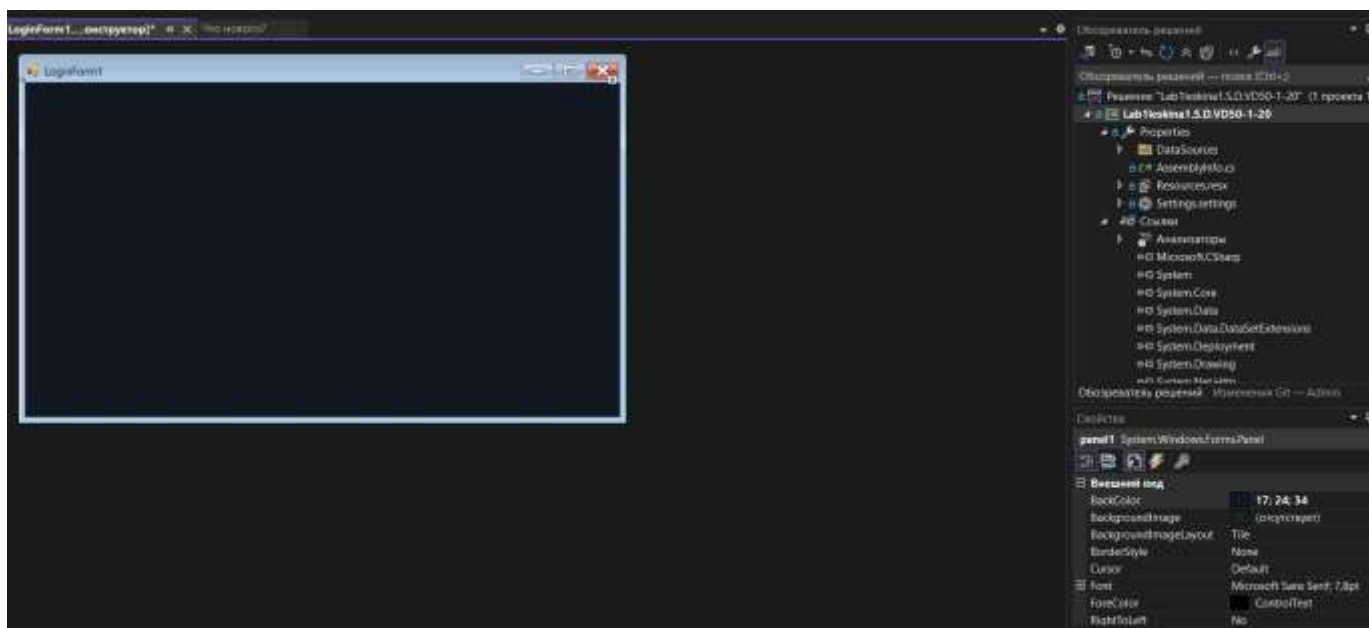


Рисунок 8 - Цвет панели

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

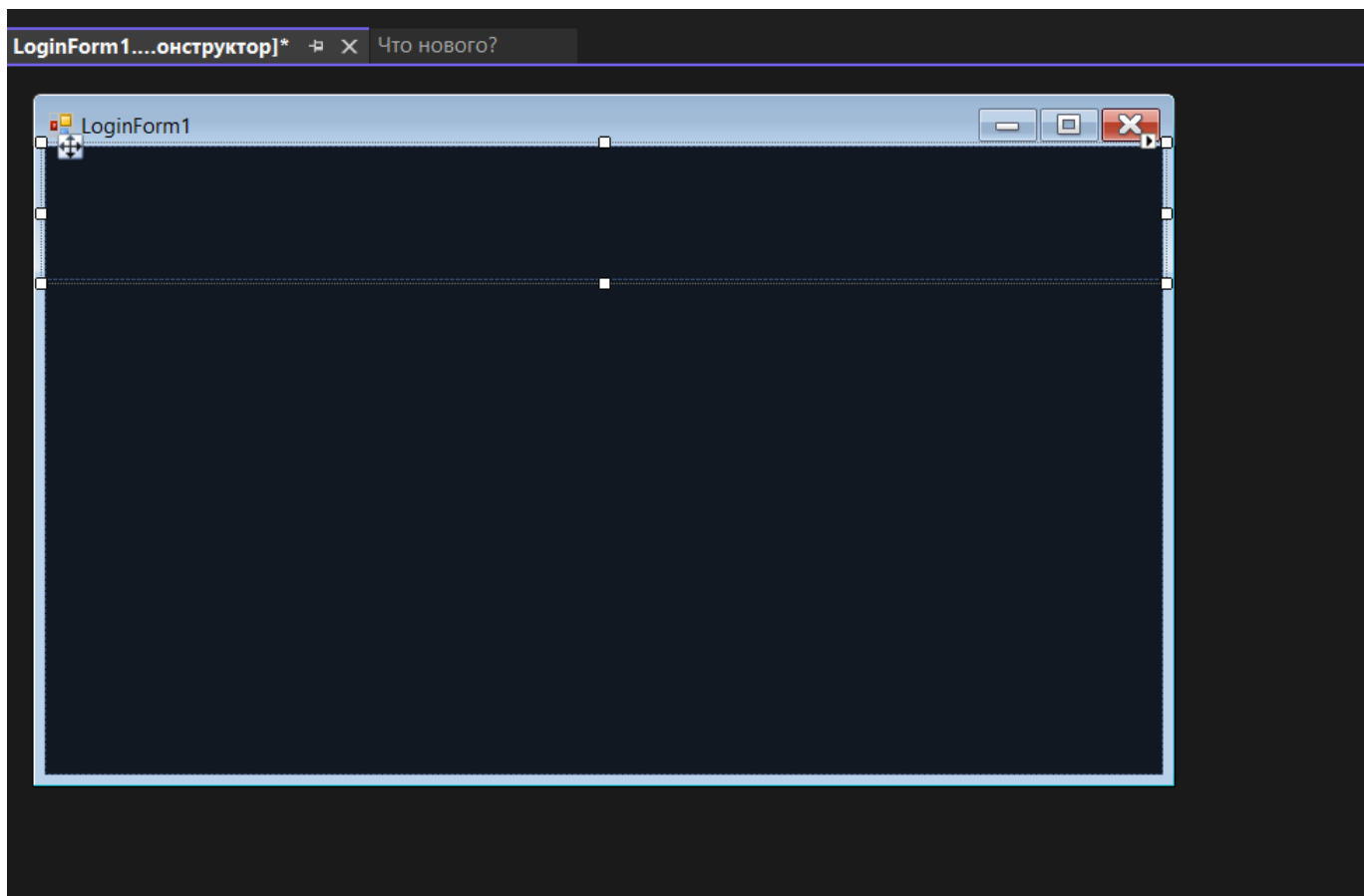


Рисунок 9 - Создание второй панели

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

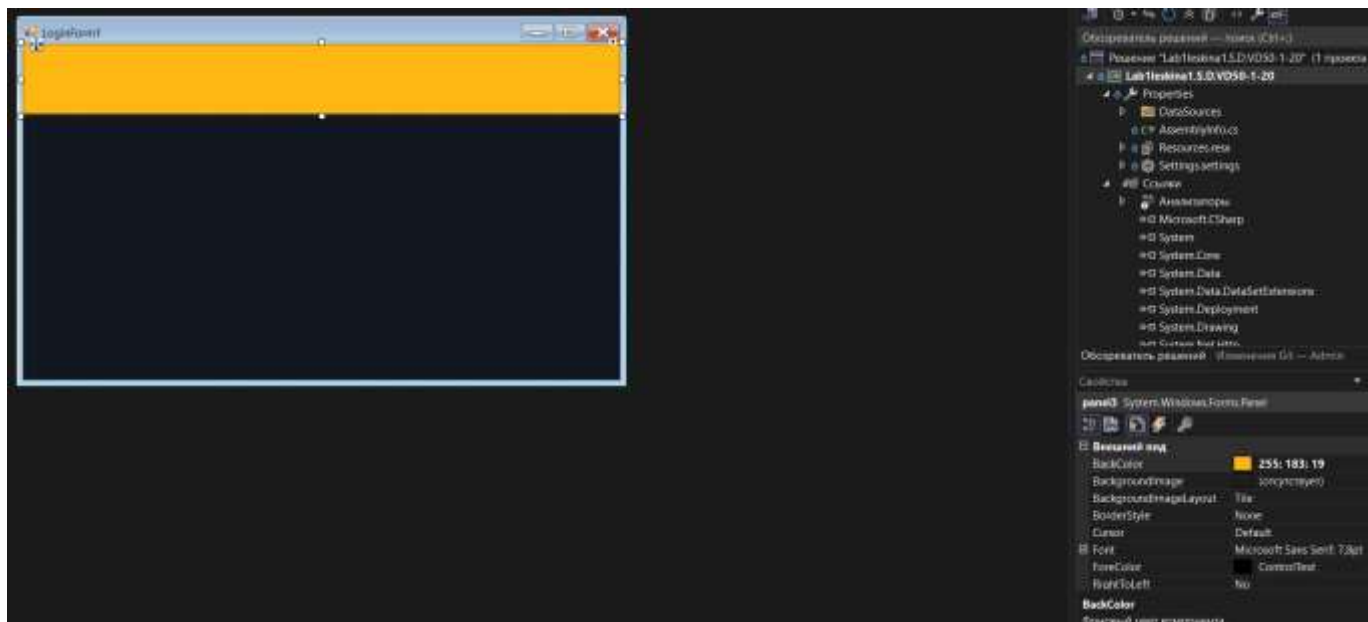


Рисунок 10 - Цвет панели

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

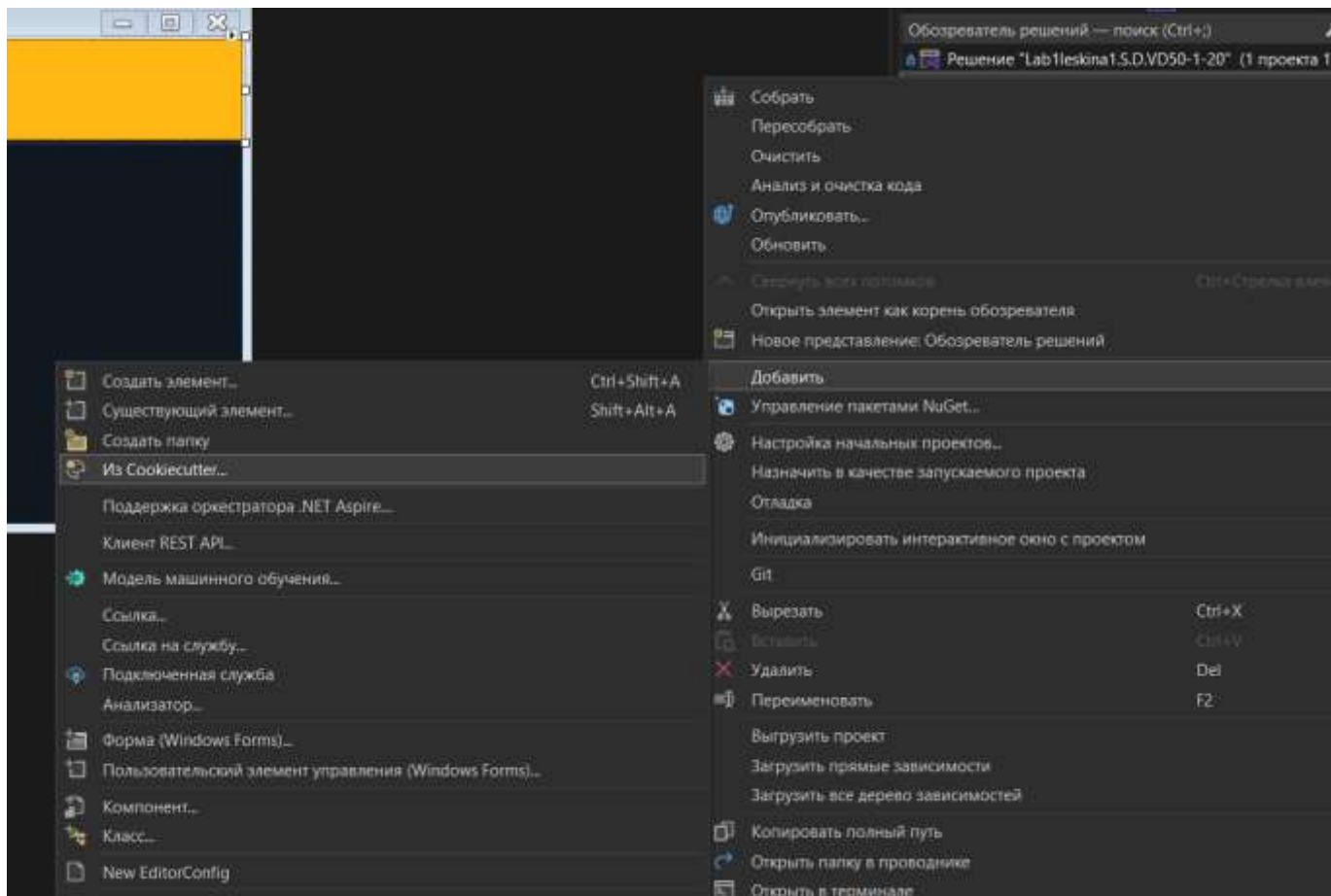


Рисунок 11 - Создание папки

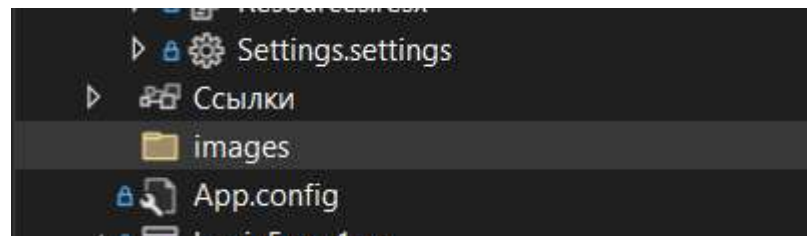


Рисунок 12 - Созданная папка

12. Создаем label и настраиваем шрифт и текст

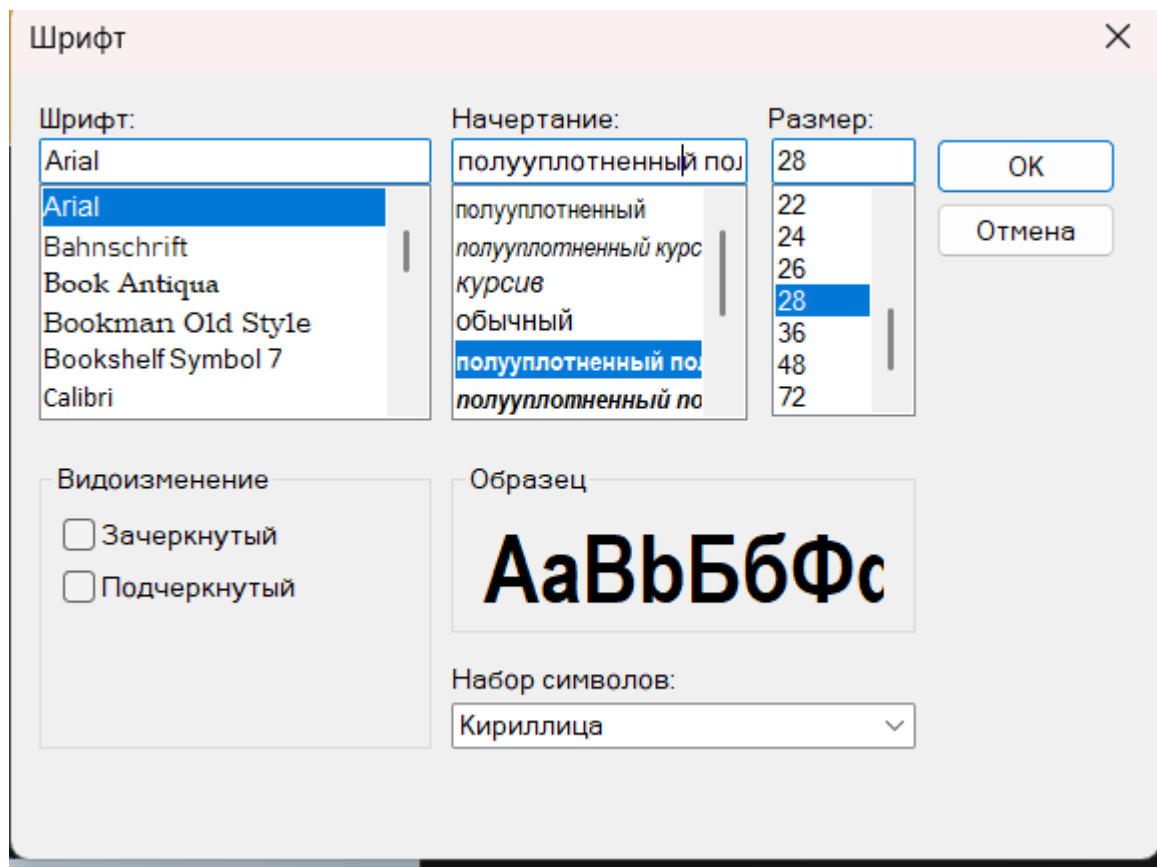
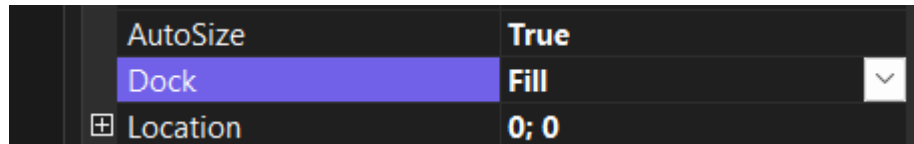


Рисунок 13 - Настройки шрифта

ImageList	(Нет)
RightToLeft	No
Text	USER LOGIN
TextAlign	TopLeft


Рисунок 14 - Текст label

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



AutoSize	True
Dock	Fill
Location	0; 0

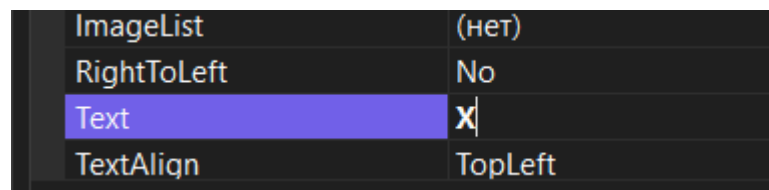
Рисунок 15 - Свойство Dock



Text	USER LOGIN
TextAlign	MiddleCenter
UseMnemonic	True

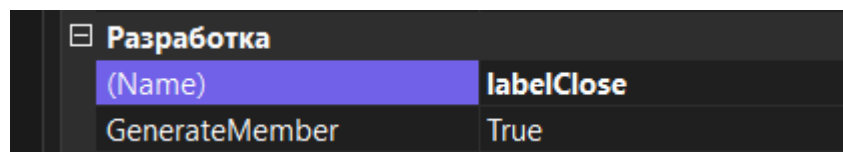
Рисунок 16 - Свойство TextAlign

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



ImageList	(нет)
RightToLeft	No
Text	x
TextAlign	TopLeft

Рисунок 17 - Текст label



Разработка	
(Name)	labelClose
GenerateMember	True

Рисунок 18 - Имя label

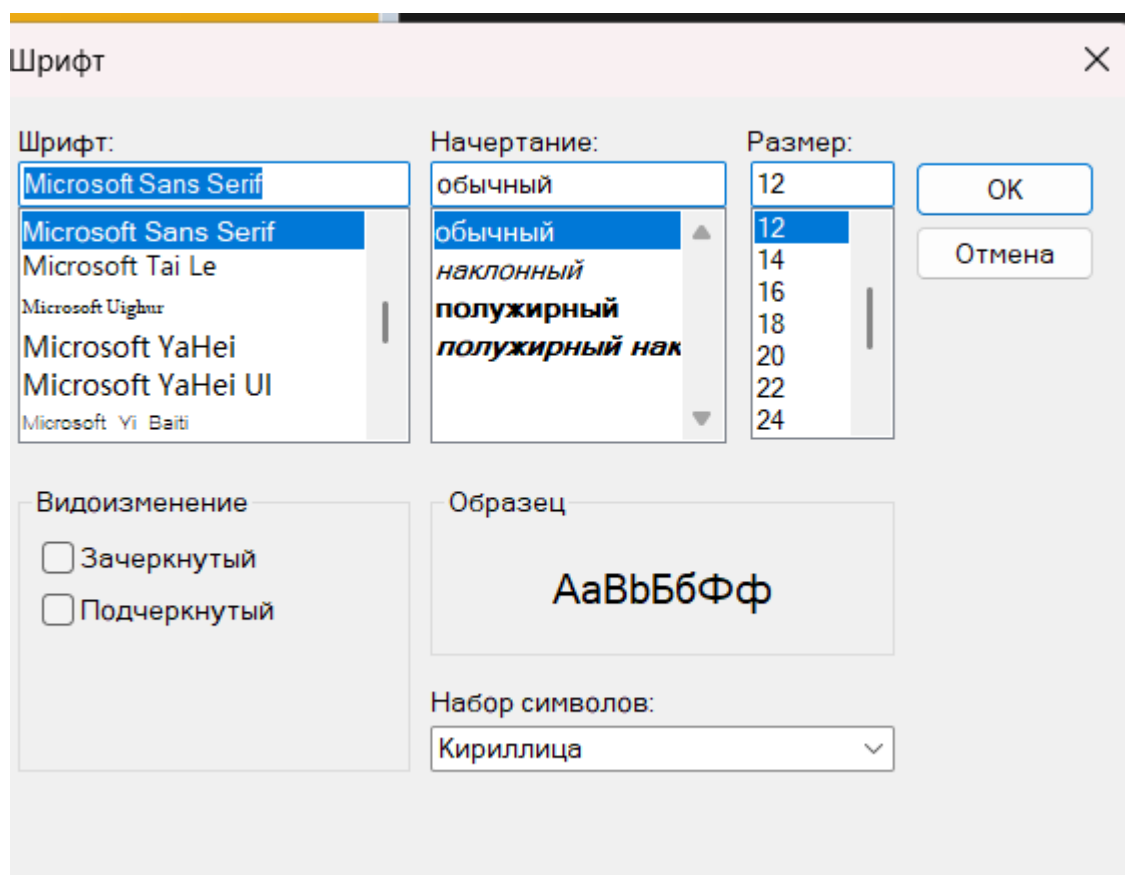


Рисунок 19 - Настройки шрифта

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

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Lab1leskina1.S.D.VD50_1_20
{
    Ссылка: 0
    internal static class Program
    {
        Ссылка: 0
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new LoginForm1());
        }
    }
}
```

Рисунок 20 - Program.cs

16. Промежуточный результат

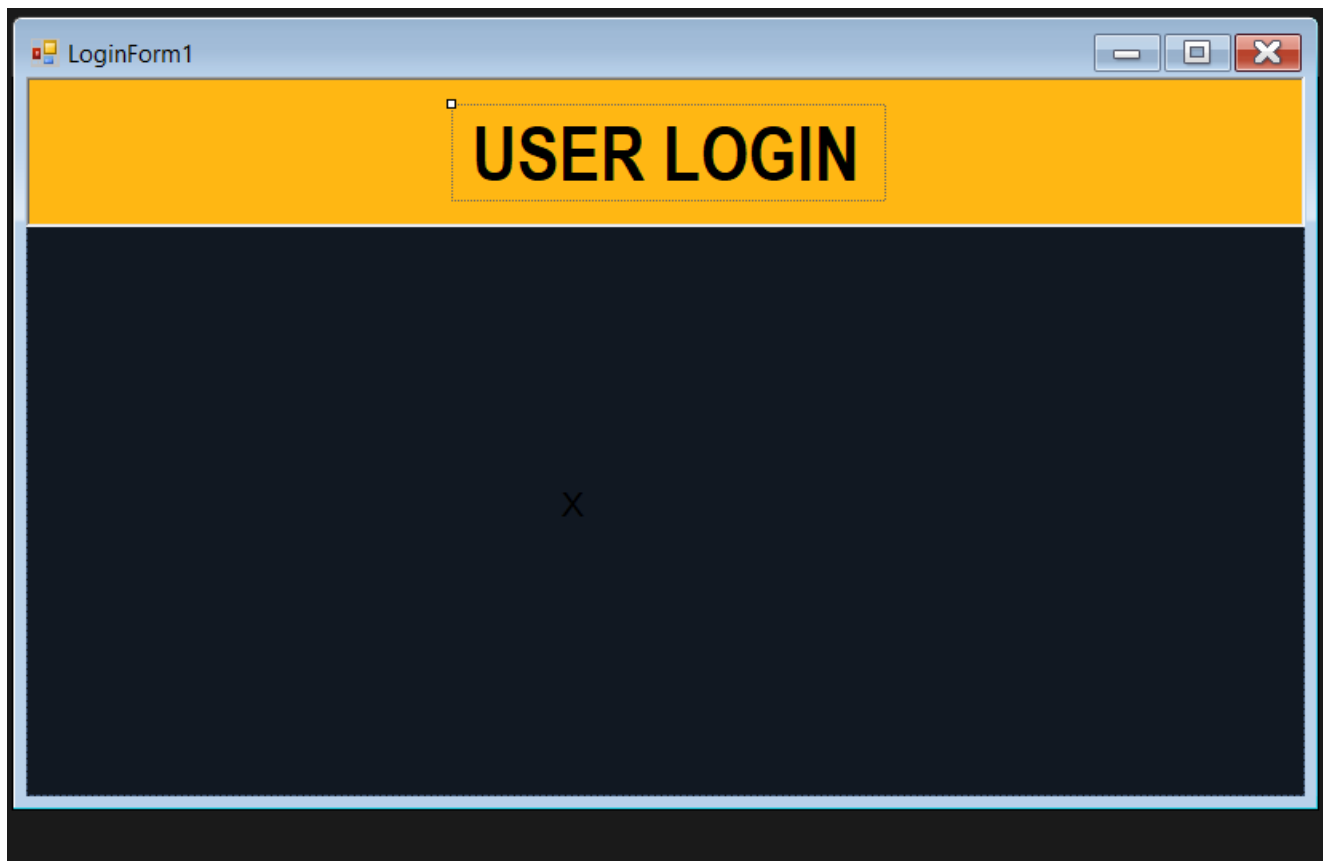


Рисунок 21 - Промежуточный результат

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

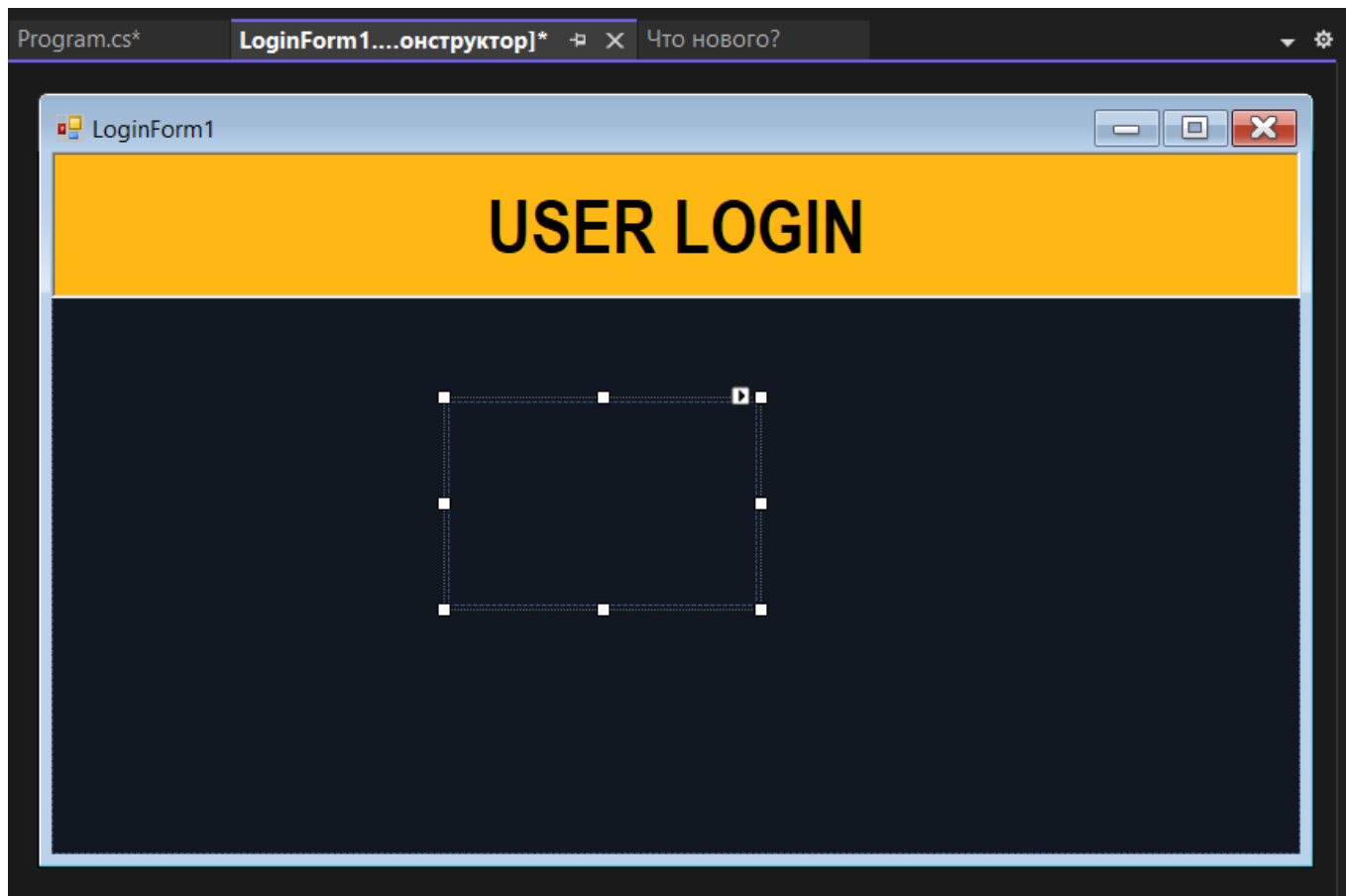


Рисунок 22 - pictureBox



Рисунок 23 - Размер pictureBox

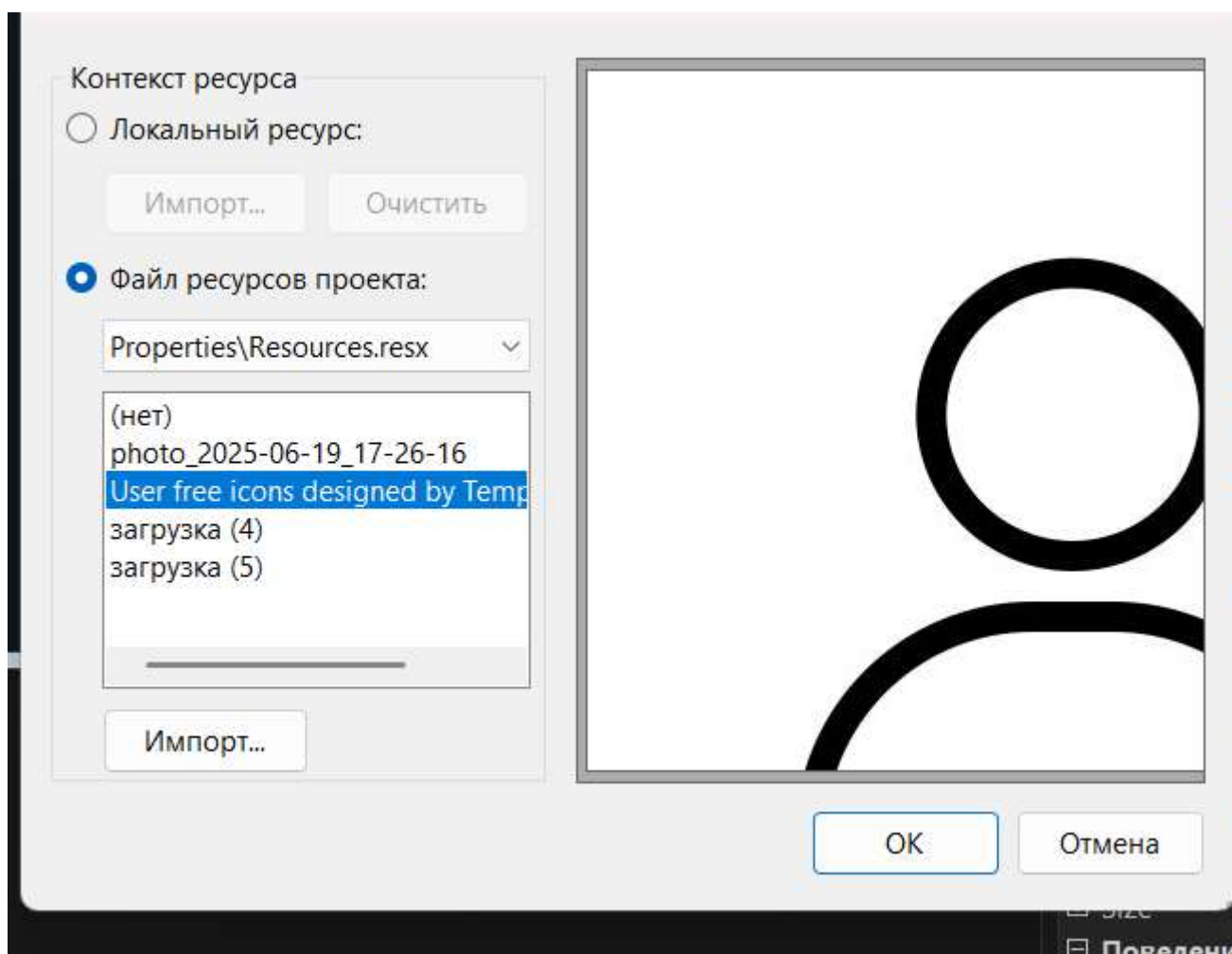


Рисунок 24 - Выбор изображения для pictureBox

18. Готовый pictureBox

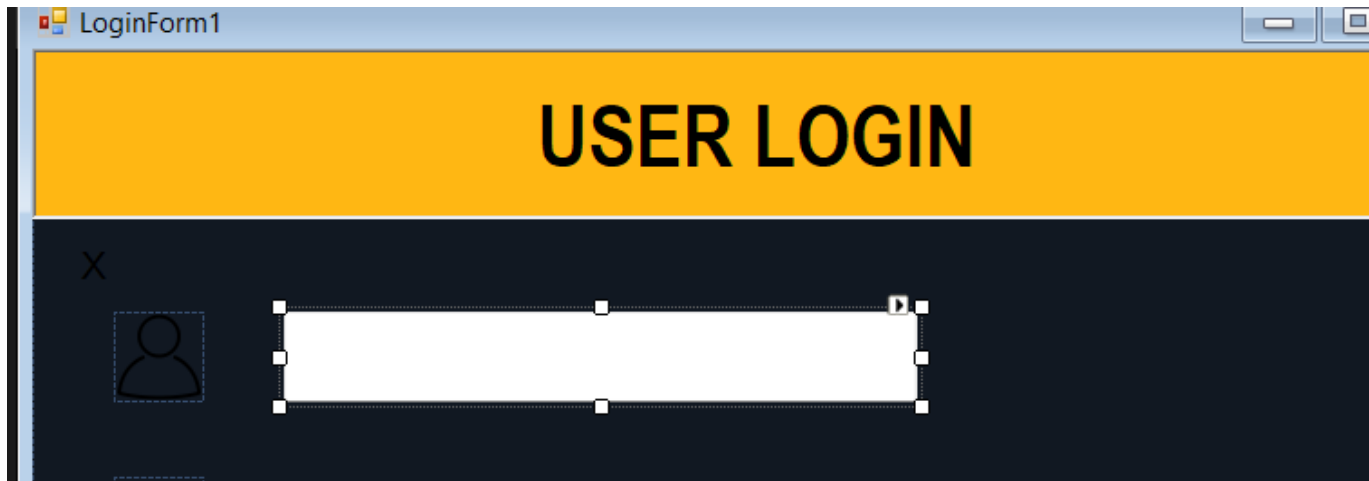


Рисунок 25 - Готовый pictureBox

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



Рисунок 26 - Свойство MultiLine

⊞	MinimumSize	0; 0
⊞	Size	350; 50
⊞	Поведение	
	AcceptsReturn	False

Рисунок 27 - Размеры textbox

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

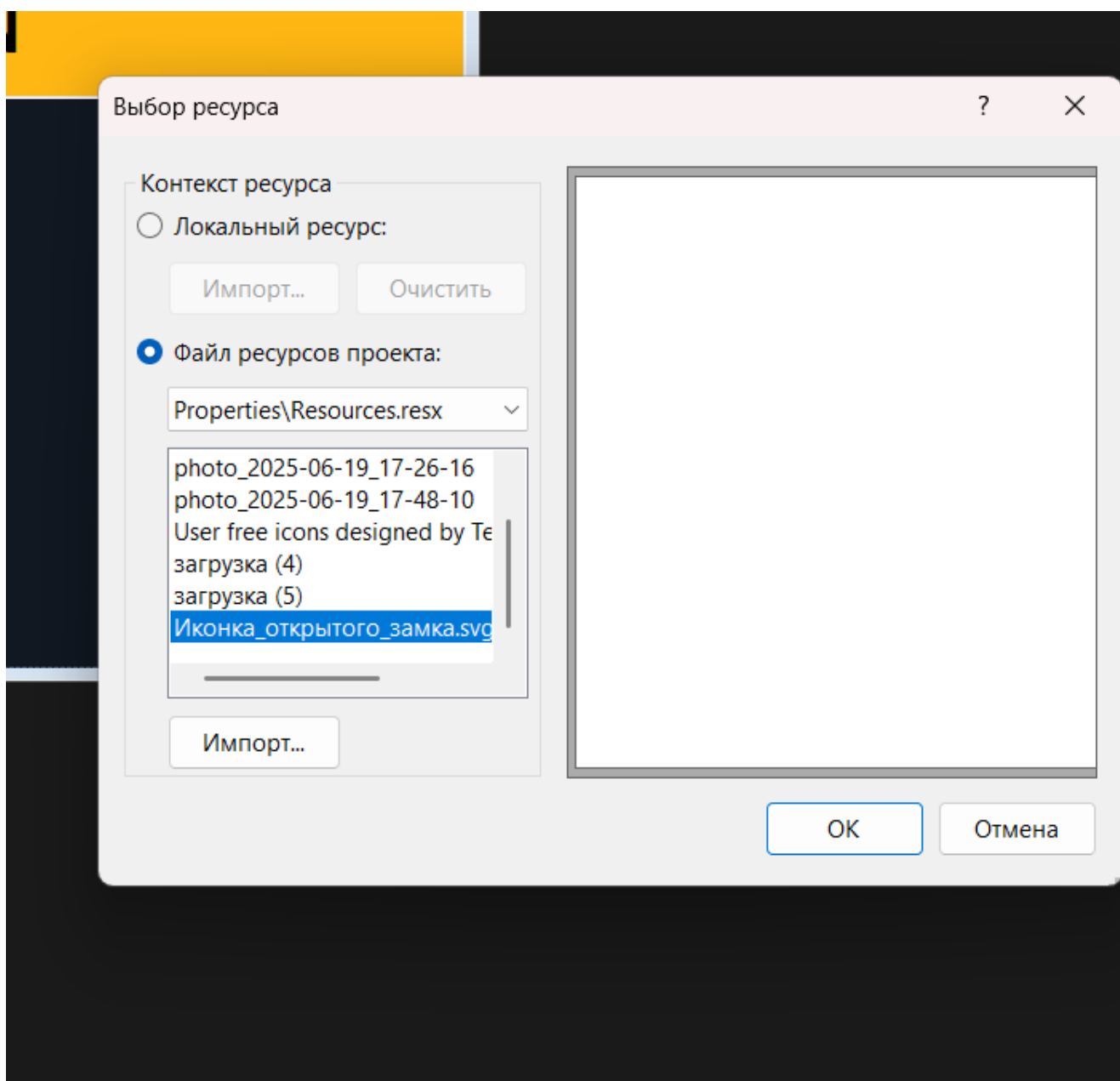


Рисунок 28 - Выбор изображения для pictureBox

21. Промежуточный результат

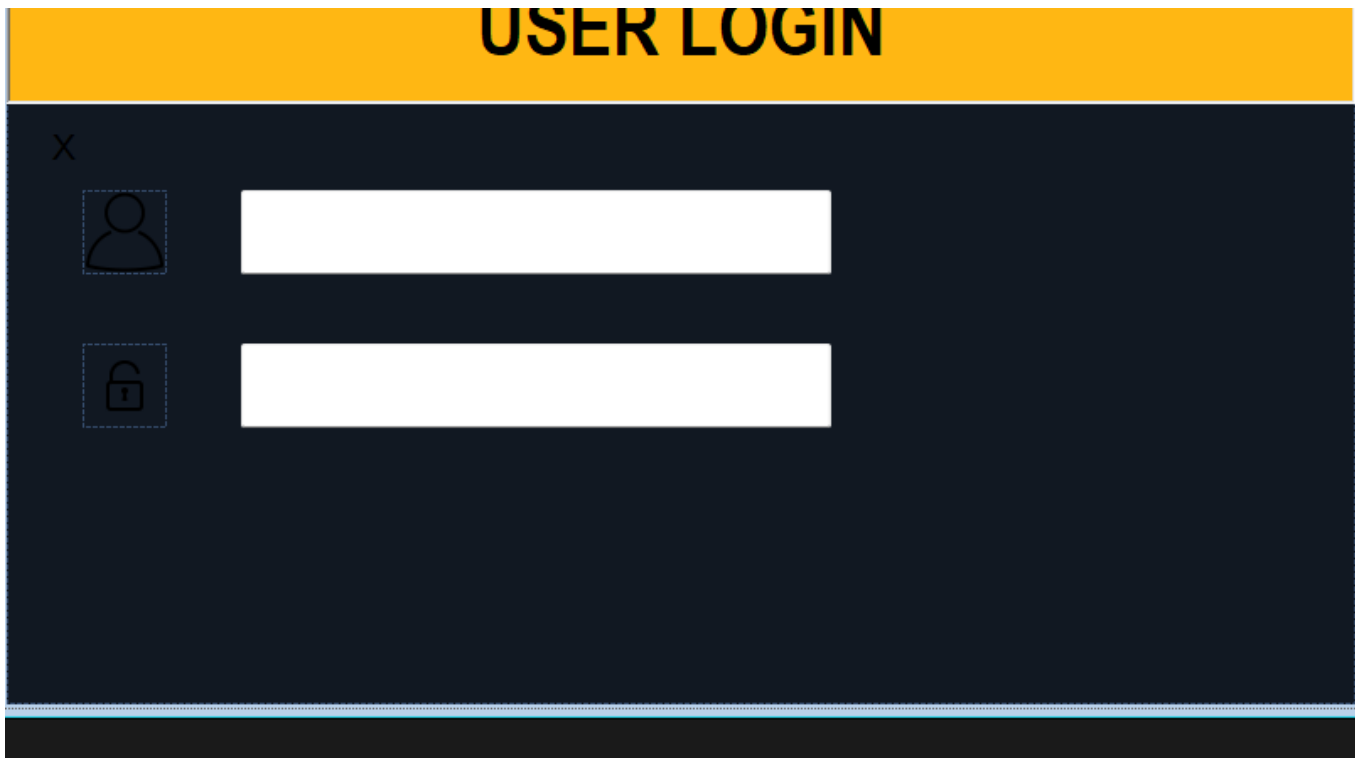


Рисунок 29 - Промежуточный результат

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

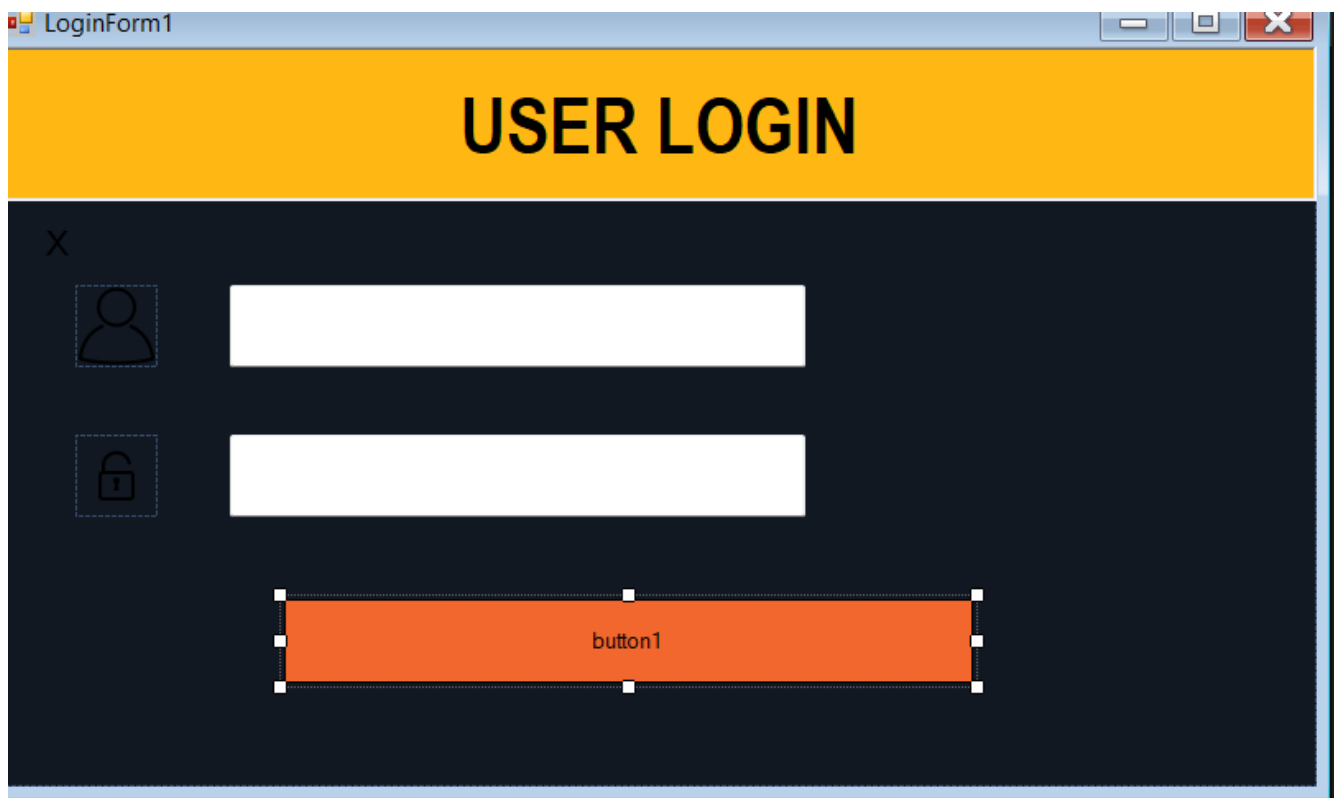


Рисунок 30 - Кнопка

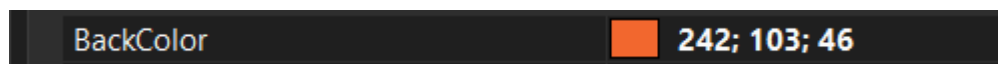


Рисунок 31 - Цвет кнопки



Рисунок 32 - Свойство FlatStyle

23. Настраиваем шрифт кнопки и цвет текста

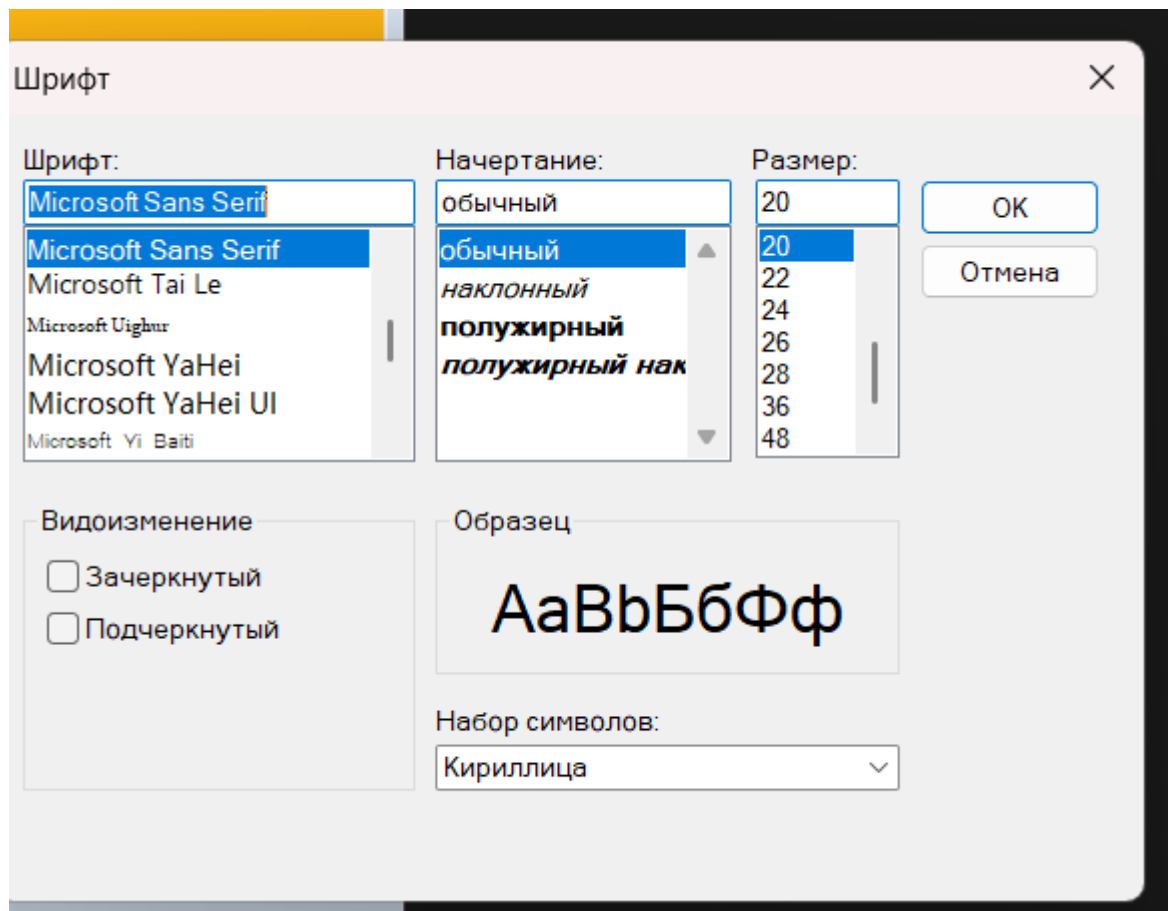


Рисунок 33 - Настройки шрифта

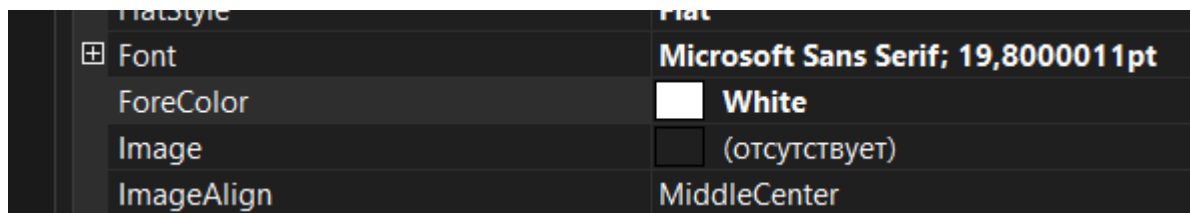


Рисунок 34 - Цвет текста

24. Убираем контур у кнопки

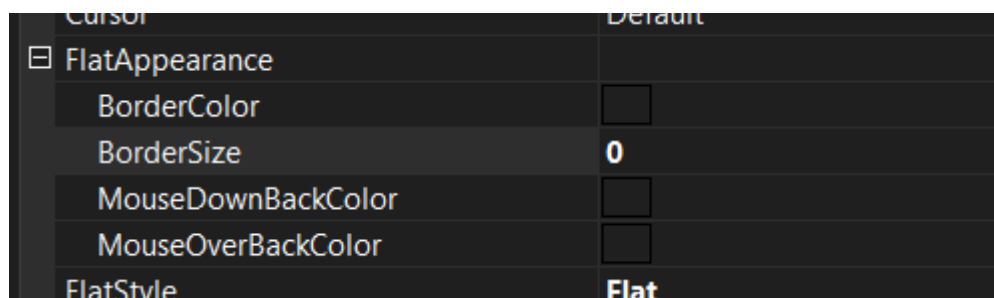


Рисунок 35 - Контур кнопки

25. У текстовых настроек настраиваем шрифты

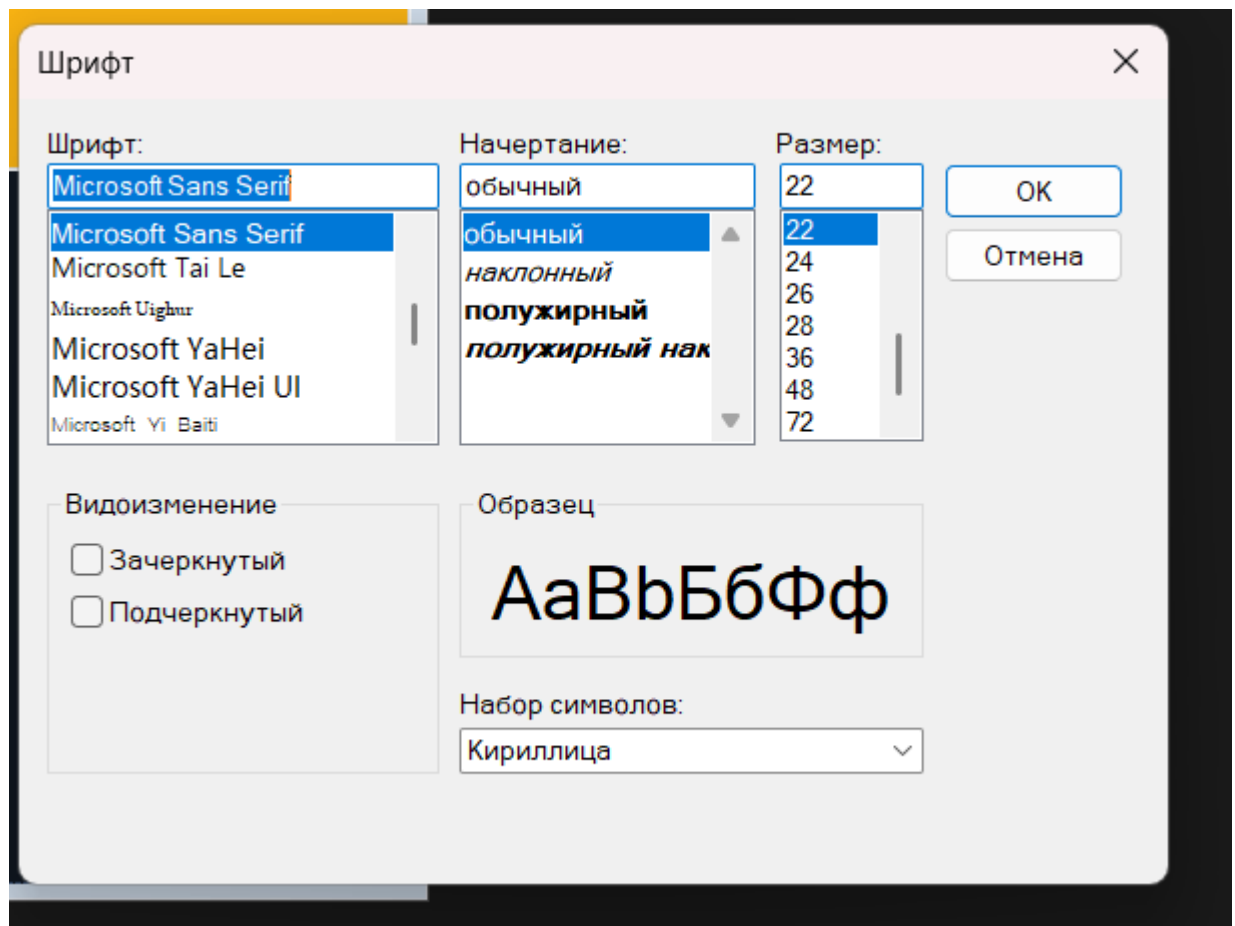


Рисунок 36 - Настройка шрифта

26. Далее переименовываем элементы

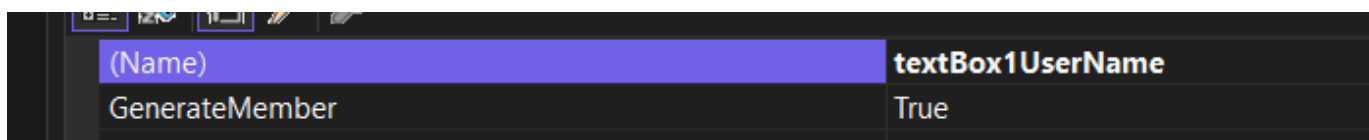


Рисунок 37 - Название textBoxName

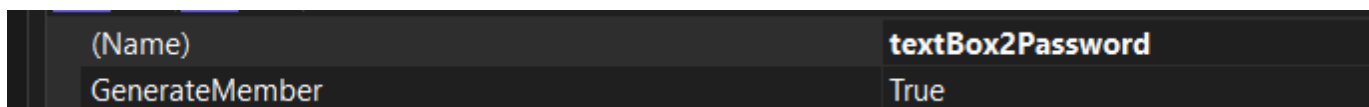


Рисунок 38 - Название textBoxPassword



Рисунок 39 - Название buttonLogin

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

tabstop	True
UseSystemPasswordChar	True
Visible	True

Рисунок 40 - Свойство UseSystemPassword

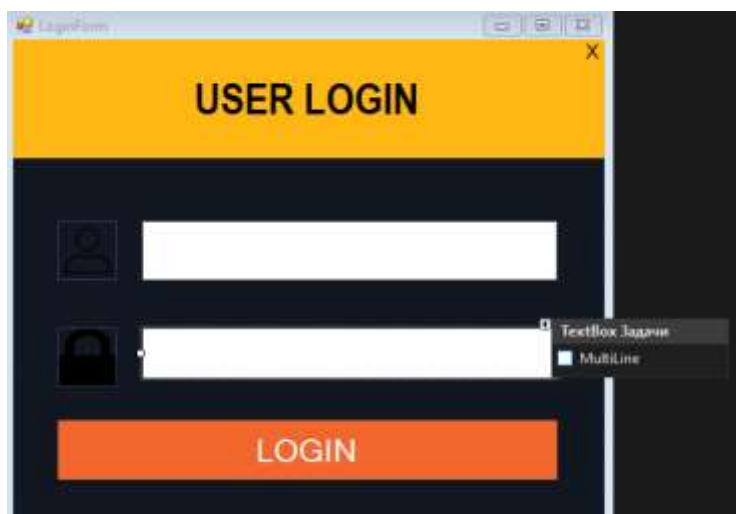


Рисунок 41 - Свойство MultiLine

28. Промежуточный результат

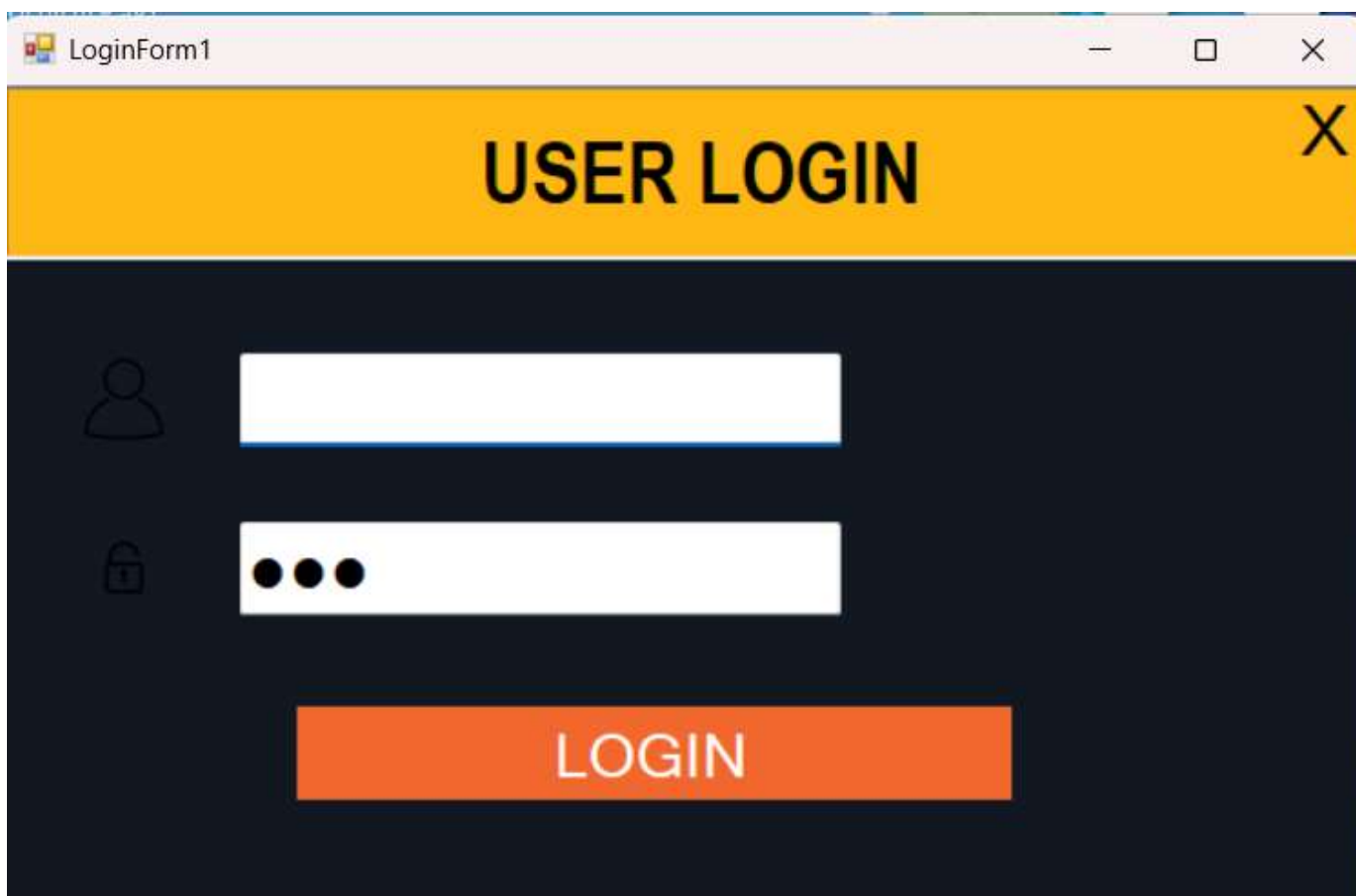


Рисунок 42 - Промежуточный результат

29. Переходим к коду, задаем textBoxPassword размер

```
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Lab1leskina1.S.D.VD50_1_20
{
    Ссылка: 3
    public partial class LoginForm1 : Form
    {
        Ссылка: 1
        public LoginForm1()
        {
            InitializeComponent();
            this.textBox2Password.AutoSize = false;
            this.textBox2Password.Size = new Size(this.textBox2Password.Size.Width, 50);
        }

        Ссылка: 1
        private void panel1_Paint(object sender, PaintEventArgs e)
        {
        }
    }
}
```

Рисунок 43 - Размер textbox

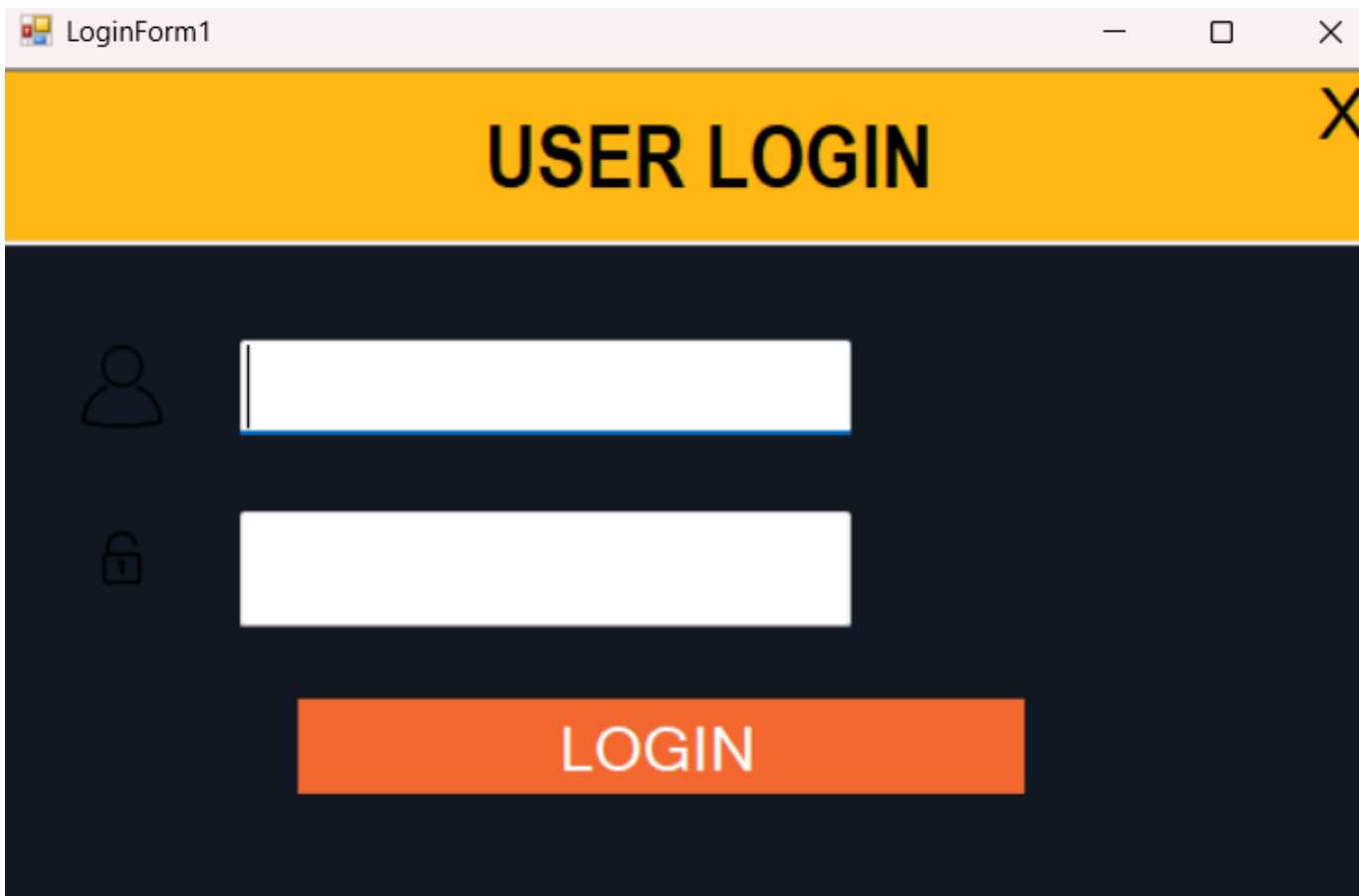
The screenshot shows a Windows application window titled "LoginForm1". The interface has a dark blue background. At the top, there is a yellow header bar with the text "USER LOGIN" in bold black letters. Below the header, there are two white input fields. The first input field is preceded by a user icon, and the second input field is preceded by a lock icon. At the bottom of the form, there is a large orange button with the word "LOGIN" in white capital letters.

Рисунок 44 - Промежуточный результат

30. 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

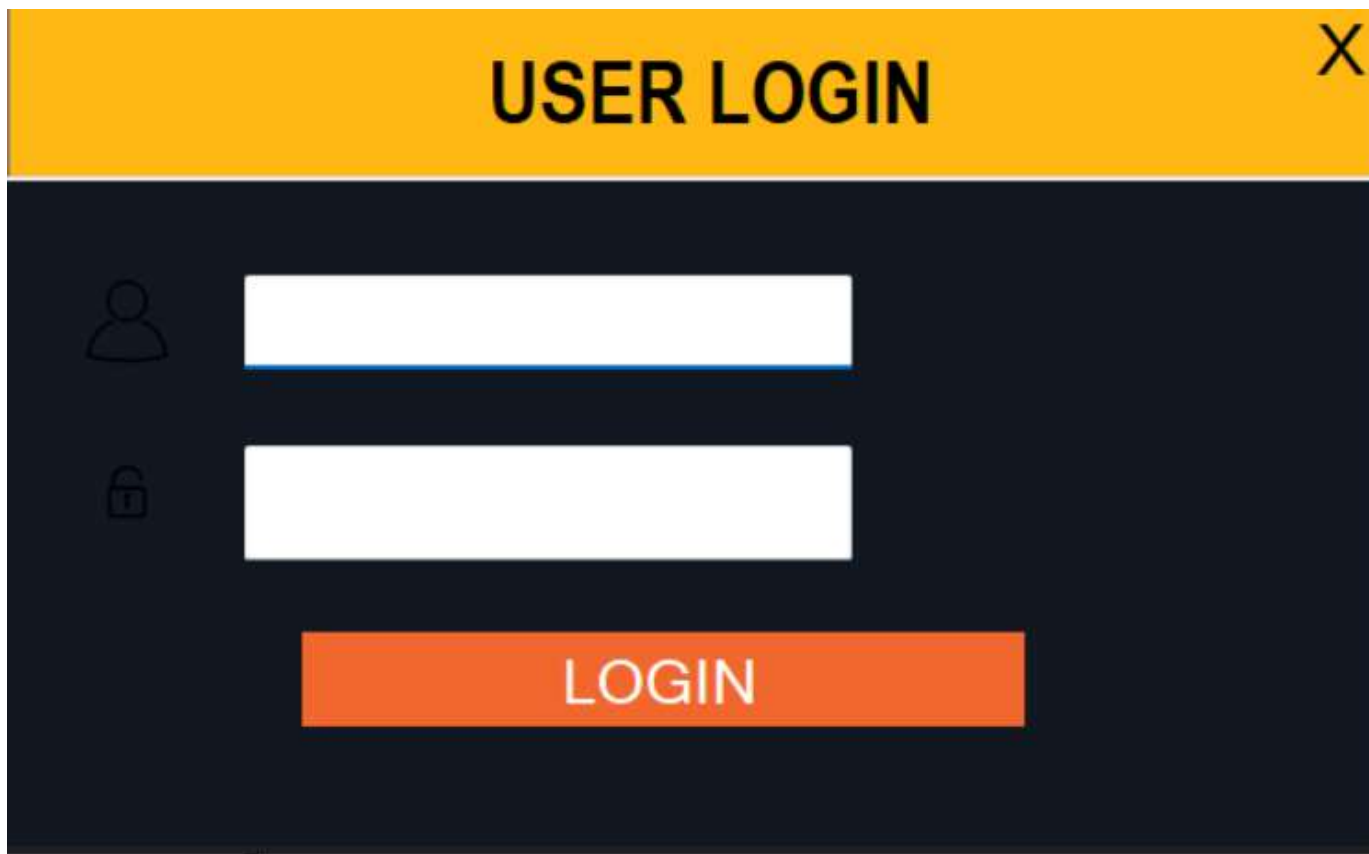


Рисунок 47 - Промежуточный результат

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

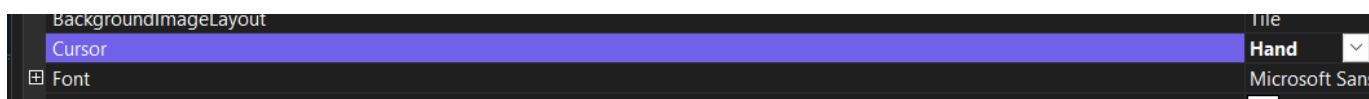


Рисунок 48 - Свойство Cursor

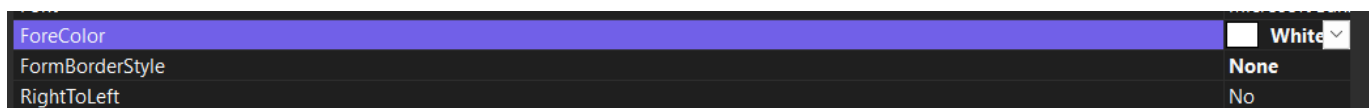


Рисунок 49 - Цвет

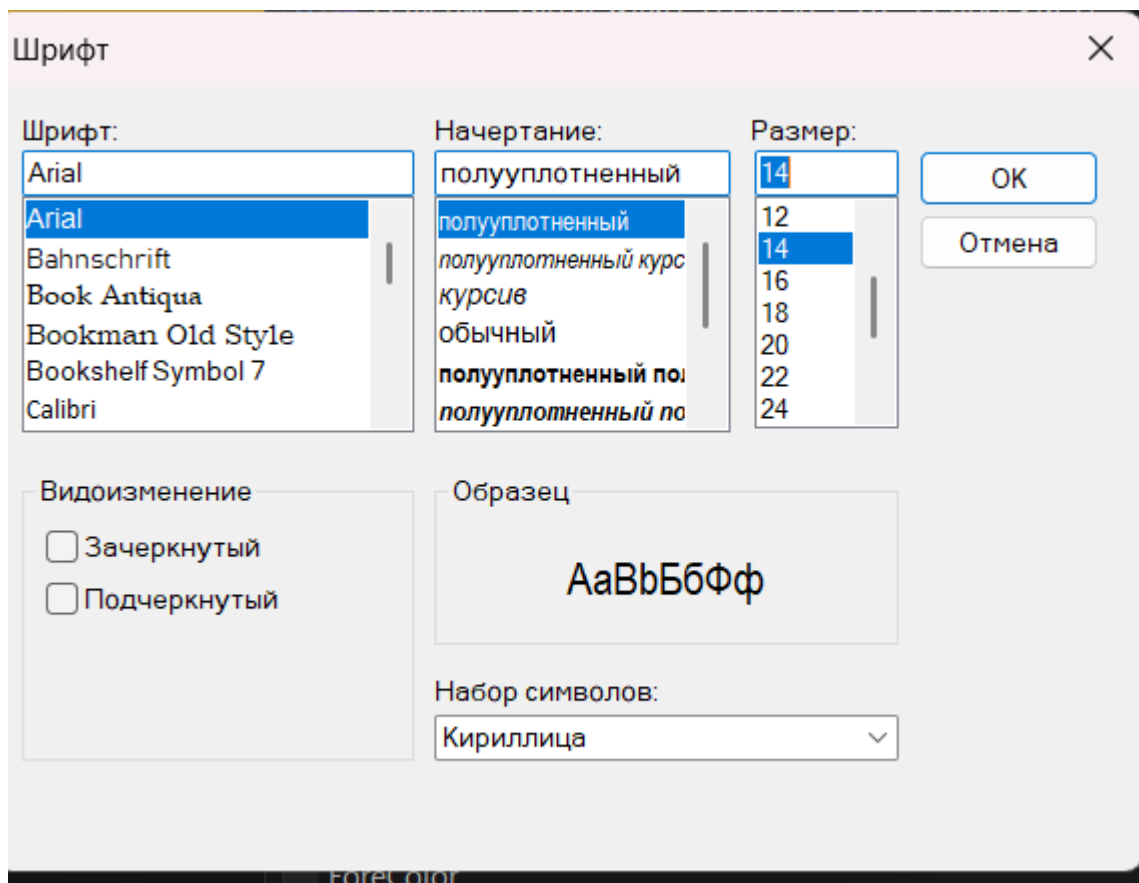


Рисунок 50 - Настройка шрифта

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

мышшь	
MouseDown	
MouseEnter	labelClose_MouseEnter
MouseHover	
MouseLeave	labelClose_MouseLeave
MouseMove	
MouseUp	

Рисунок 51 - Событие MouseEnter

```

Ссылка: 0
private void labelClose_MouseEnter(object sender, EventArgs e)
{
    labelClose.ForeColor = Color.Black; labelClose.Location = new Point(
}
  
```

Рисунок 52 - Код события MouseEnter

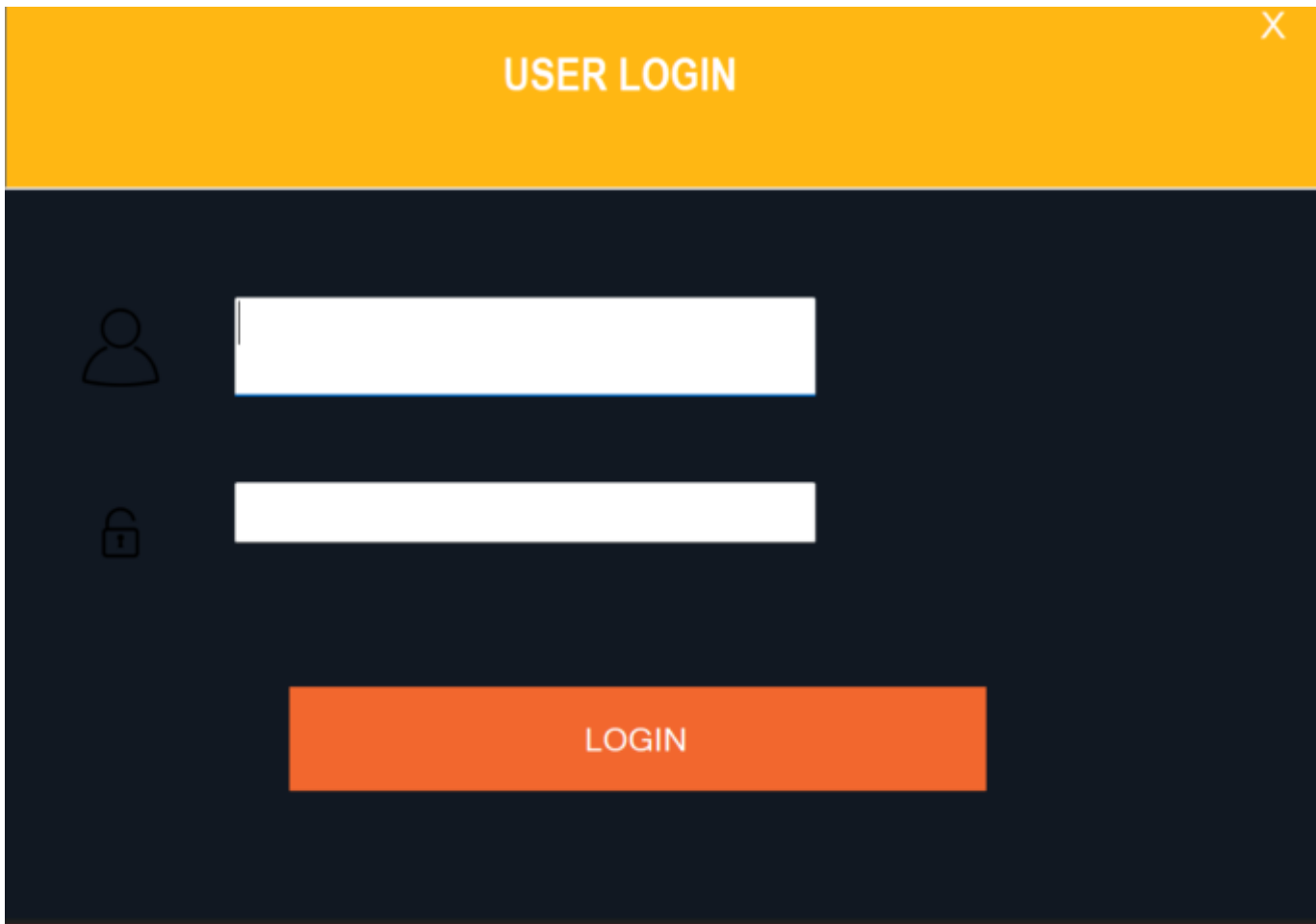
MouseDown	
MouseEnter	labelClose_MouseEnter
MouseHover	
MouseLeave	labelClose_MouseLeave
MouseMove	
MouseUp	
Перетащить и отпустить	

Рисунок 53 - Событие MouseLeave


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


Рисунок 54 – Код события MouseLeave

33. Готовая форма

A user login form with a yellow header bar containing the text "USER LOGIN" and a close button (X) in the top right corner. The main area has a dark blue background. It features two white input fields: the first is preceded by a user icon, and the second by a lock icon. Below the input fields is a large orange button with the text "LOGIN".

USER LOGIN





LOGIN

Рисунок 55 -Готовая форма

34. Форма при наведении курсора на кнопку закрыть

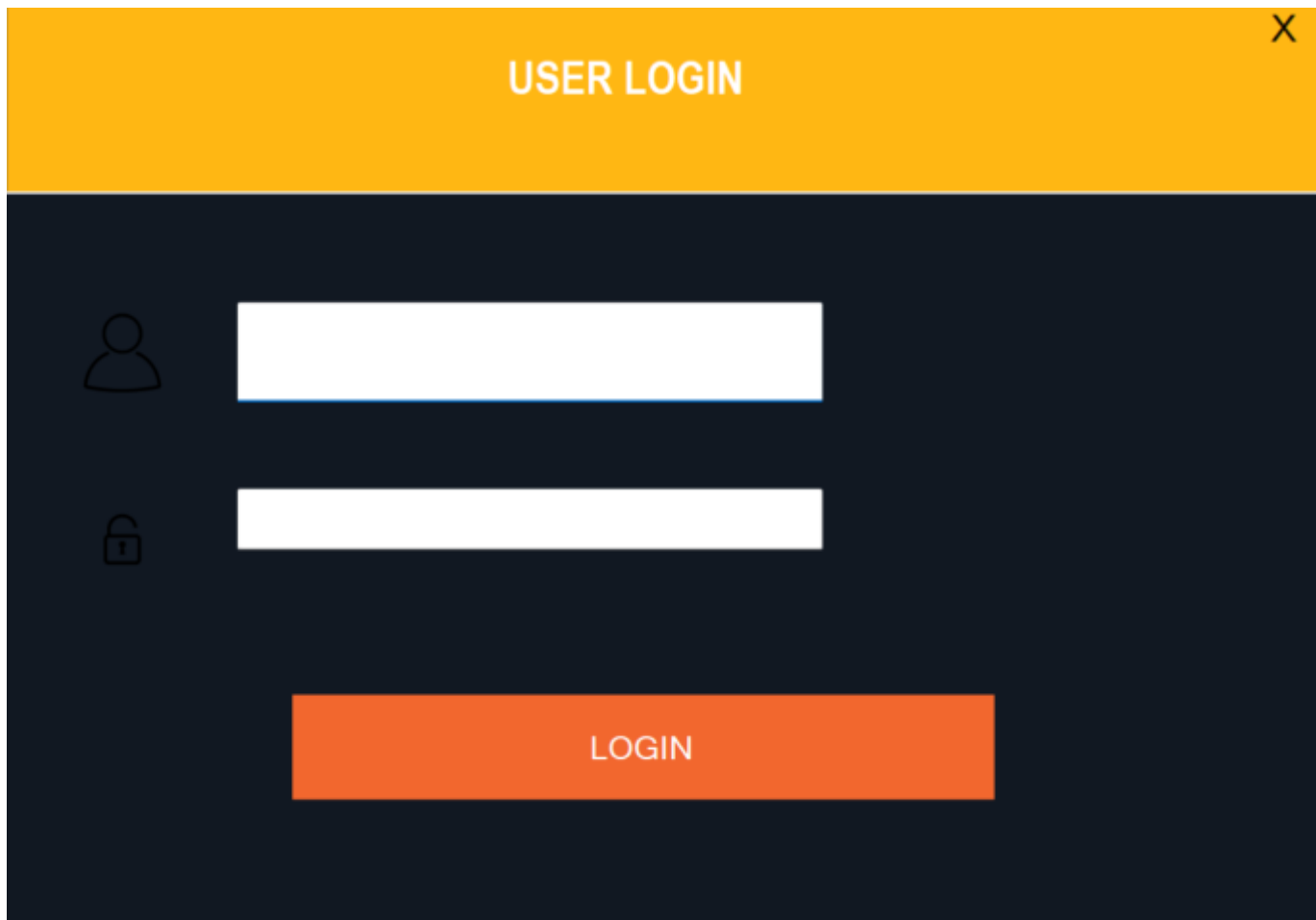


Рисунок 56 - Форма при наведении курсора на кнопку закрыть

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


```
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

⌕ Загрузки сообщества MySQL

◀ Соединитель/СЕТЬ


[Релиз общедоступной версии \(GA\)](#) [Архивы](#) 

Connector/NET 9.3.0

Выберите Операционную систему:

Microsoft Windows ▾

Windows (x86, 32-разрядная версия), установщик MSI	9.3.0	1,6 М	Скачать
(mysql-connector-net-9.3.0.msi)		MD5: 5fe4764865f6ce37b88525007dd588c7 Подпись	

 Мы рекомендуем использовать [контрольные суммы MD5](#) и [подписи GnuPG](#) для проверки целостности загружаемых пакетов.

ORACLE © 2025 Oracle

[Конфиденциальность](#) / [Не Продавайте Мою Информацию](#) | [Условия использования](#) | [Политика в отношении товарных знаков](#) | [Настройки cookie](#)

Рисунок 58 - Сайт mysql connector

3. Далее необходимо выполнить установку

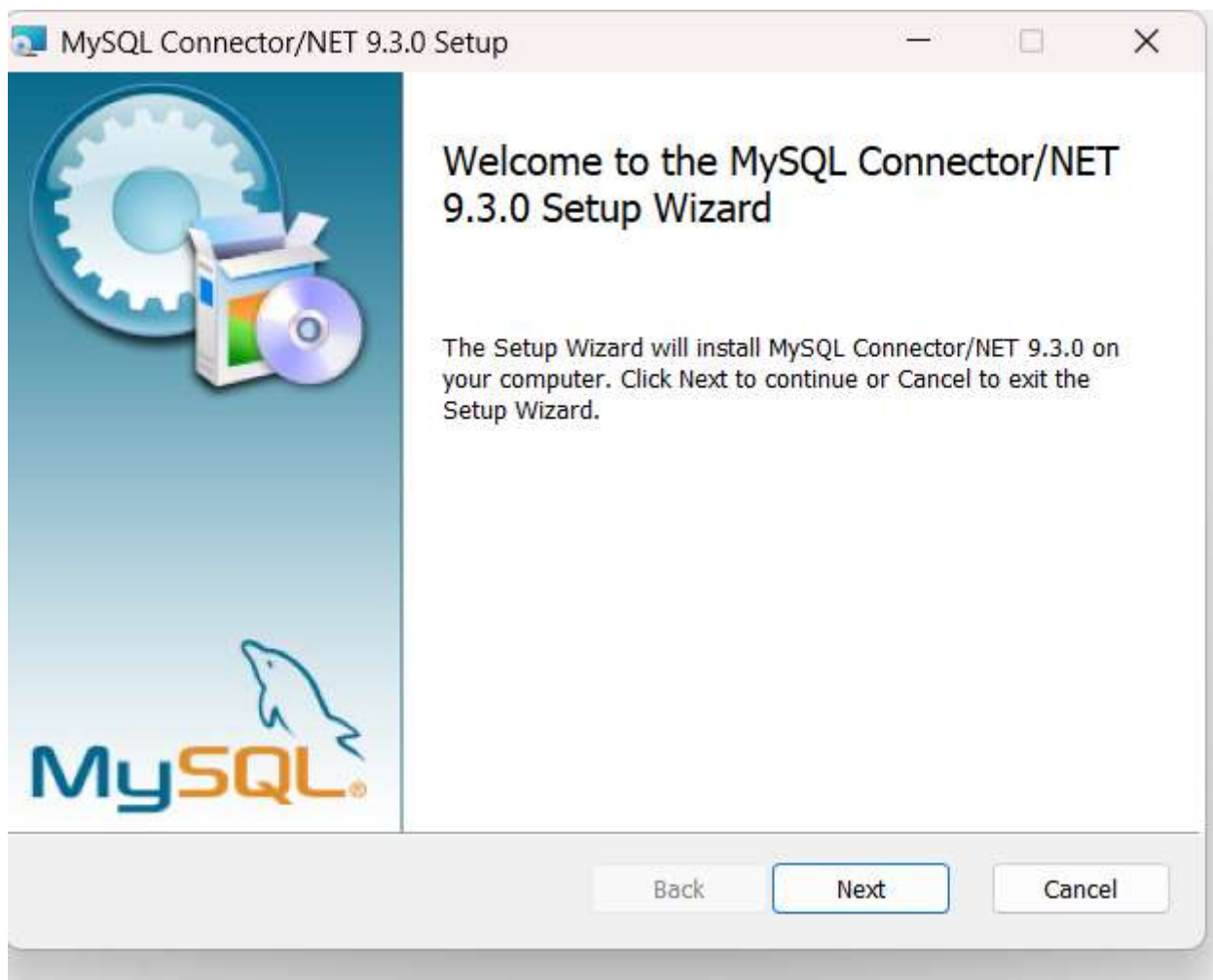


Рисунок 59 - Установка

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

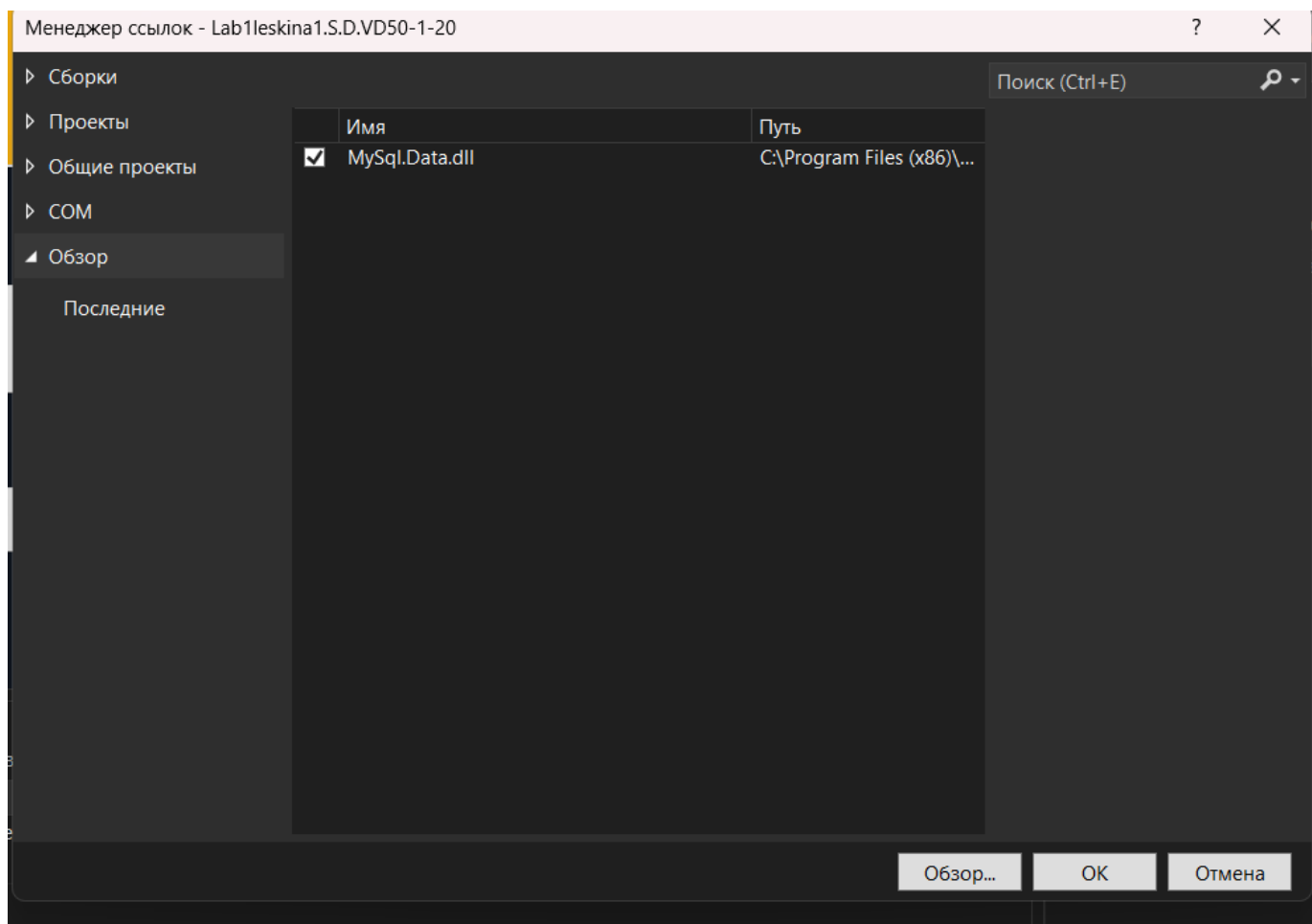


Рисунок 59 - Менеджер ссылок

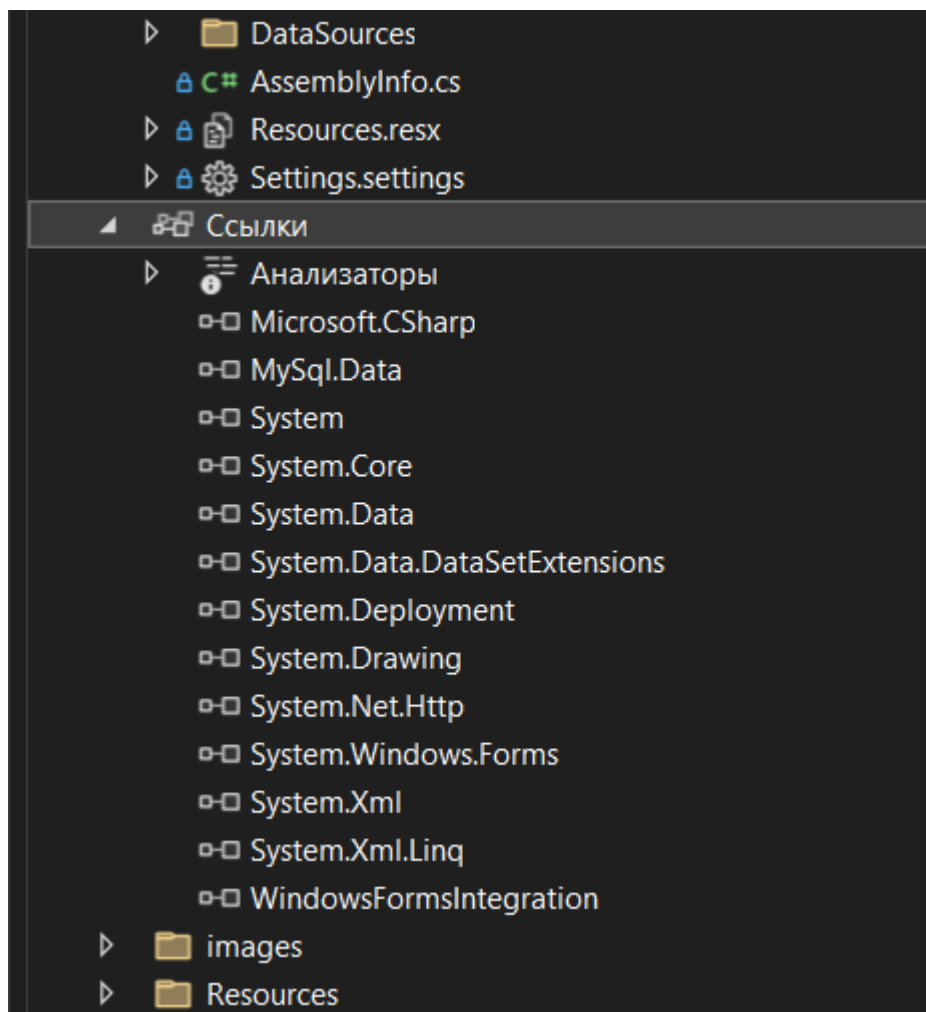


Рисунок 60 - Добавленные ссылки

5. Создаем подключение и в пространство имен подключаем MySql.Data.MySqlClient

```
using System.Drawing;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
using System.Windows.Forms;  
  
namespace Lab1leskina1.S.D.VD50_1_20  
{  
    Ссылка: 3  
    public partial class LoginForm1 : Form  
    {  
        MySqlConnection connection = new MySqlConnection();  
    }  
}
```

Рисунок 61 - Пространство имен

6. Запускаем сервер и заходим в PhpMyAdmin

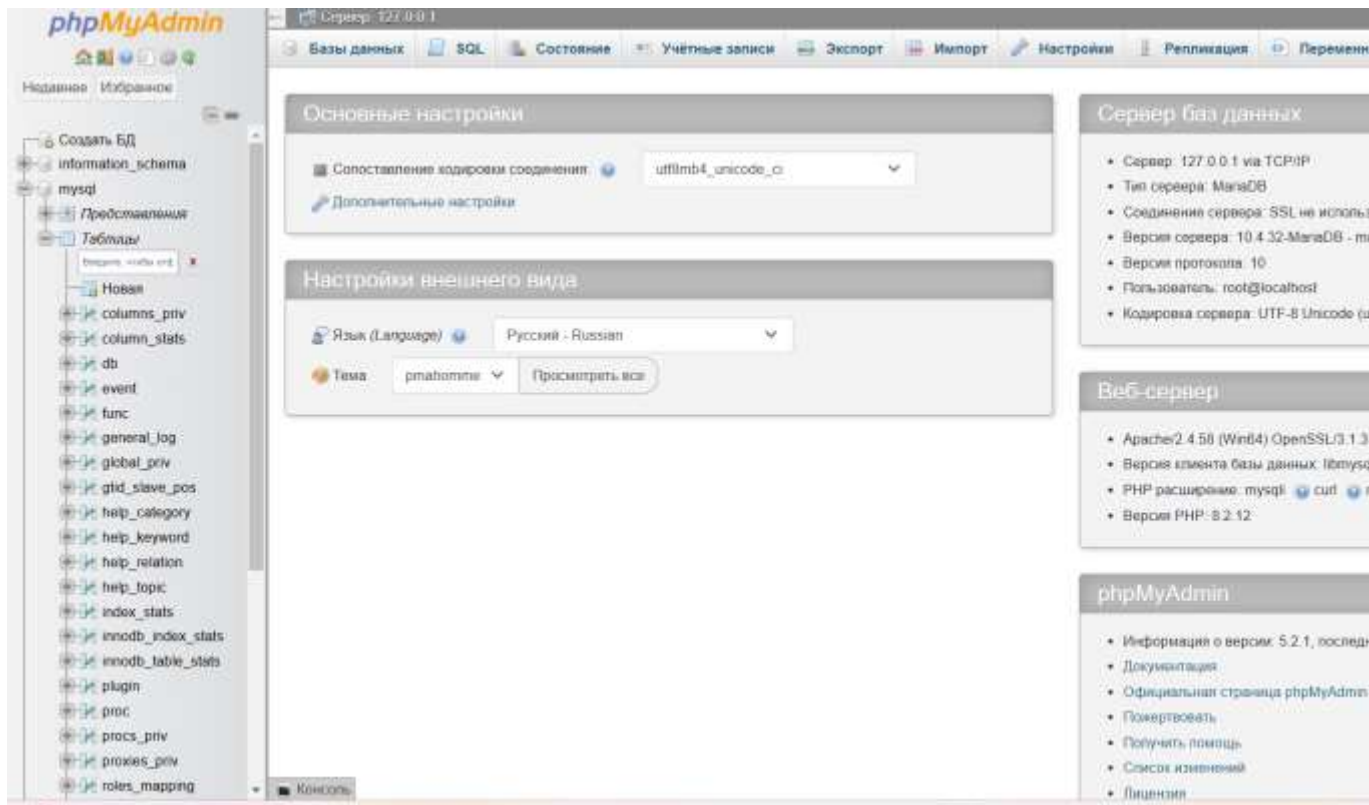


Рисунок 62 - PhpMyAdmin

7. Создаем новую бд под названием csharp_users_db

Создать базу данных

csharp_users_db utf8mb4_general_ci

Создать

Рисунок 63 - Создание бд

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

Создать новую таблицу

Имя таблицы Количество столбцов

users 3

Создать

Рисунок 64 - Создание таблицы

9. Заполняем в таблице атрибуты

ё

Имя	Тип	Длина/Значения	По умолчанию	Сравнение	Атрибуты	Null	Индекс	А_И	Коммент
id	INT		Нет			<input type="checkbox"/>	—	<input type="checkbox"/>	
username	VARCHAR	100	Нет			<input type="checkbox"/>	—	<input type="checkbox"/>	
password	VARCHAR	100	Нет			<input type="checkbox"/>	—	<input type="checkbox"/>	

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;3306;username=root;password=;database=csharp_users_db");
```

Рисунок 66 - Подключение бд

11. Создаем новый класс

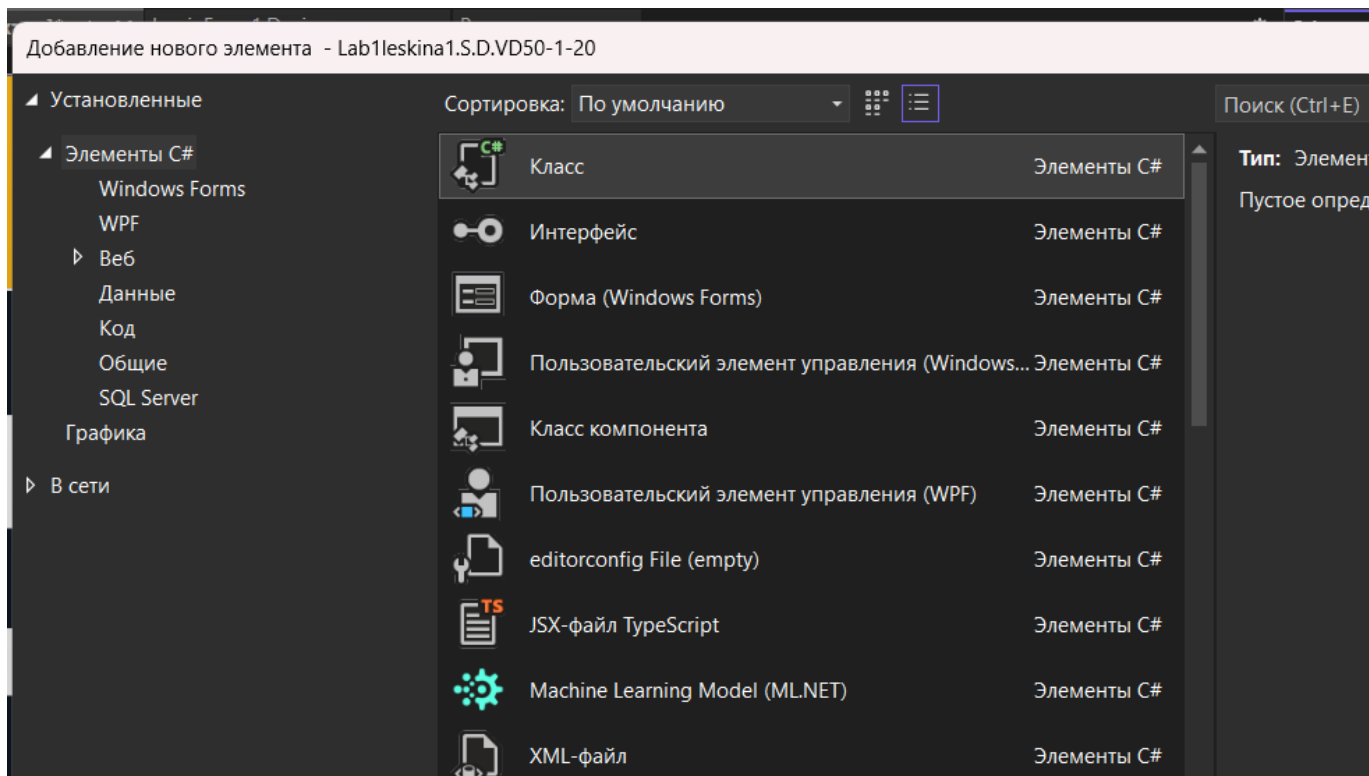


Рисунок 67 - Создание нового класса

12. Копируем наше подключение бд в новый созданный класс



Рисунок 68 - Созданный класс

13. Пишем функцию для открытия соединений

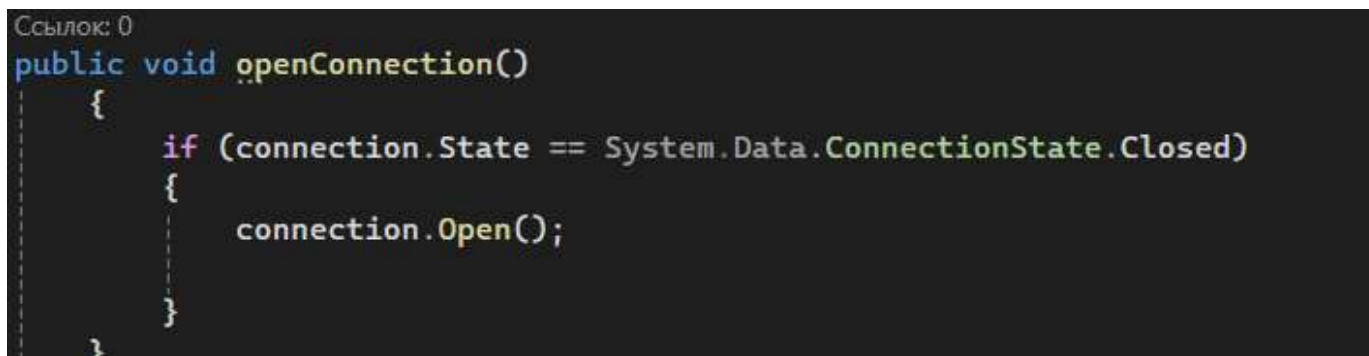


Рисунок 69 - Открытие соединения

14. Создаем функцию для закрытия соединения

```
public void closeConnection()
{
    {
        if (connection.State == System.Data.ConnectionState.Closed)
        {
            connection.Close();
        }
    }
}
```

Рисунок 70 - Закрыие соединения

15. Создаем функцию для возврата соединения

```
Ссылка: 0
public MySqlConnection getConnection()
{ return connection;
}
}
```

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 button1Login_Click(object sender, EventArgs e)
{
    DB db = new DB();
    string username = textBox1UserName.Text;
    string password = textBox2Password.Text;

    DataTable table = new DataTable();
    MySqlDataAdapter adapter = new MySqlDataAdapter();
    MySqlCommand command = new MySqlCommand("SELECT * FROM 'users' WHERE 'U=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

17. Вносим в нашу бд данные пользователя для проверки авторизации

Столбец	Тип	Функция	Null	Значение
id	int(11)	<input type="text"/>		<input type="text"/>
username	varchar(100)	<input type="text"/>		<input type="text" value="user1"/>
password	varchar(100)	<input type="text"/>		<input type="text" value="pass"/>

☒ Игнорировать

Вперёд

Рисунок 73 - Внесения данных в бд

Тестируем результат

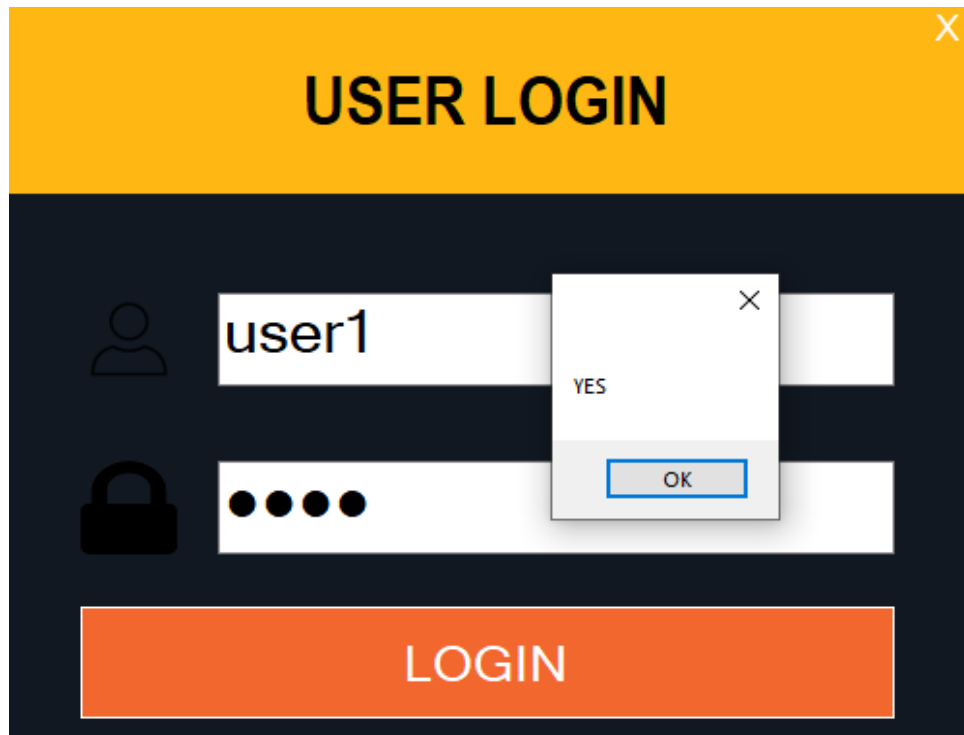


Рисунок 74 - Результат при введении верных логин и пароль

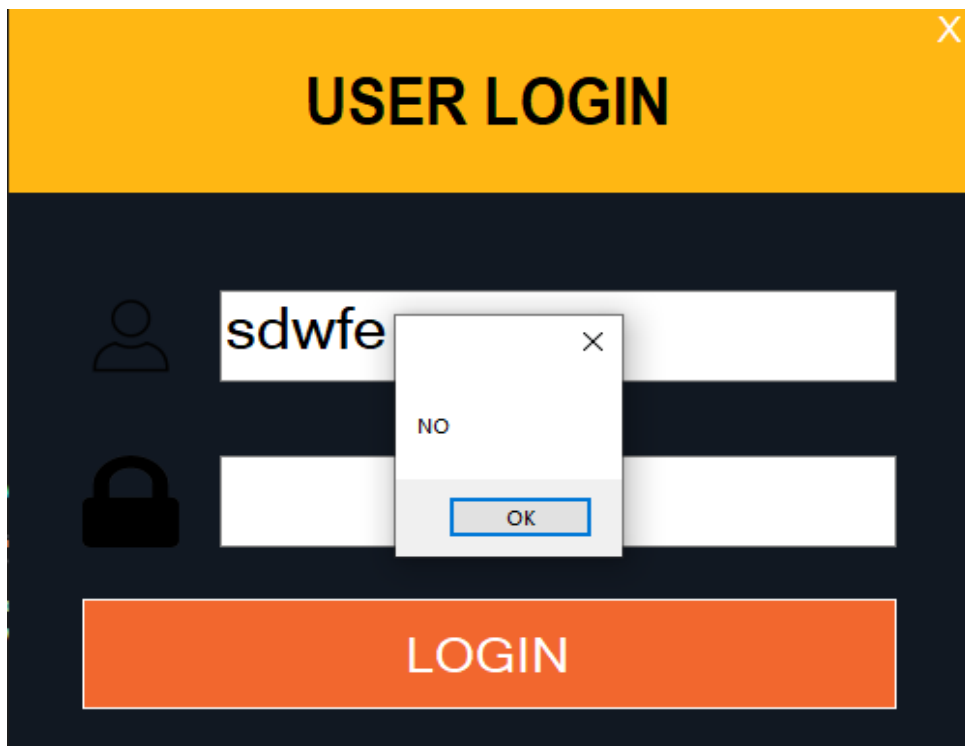


Рисунок 75 - Результат при введении неверных логин и пароль

Добавляем новую форму в проект

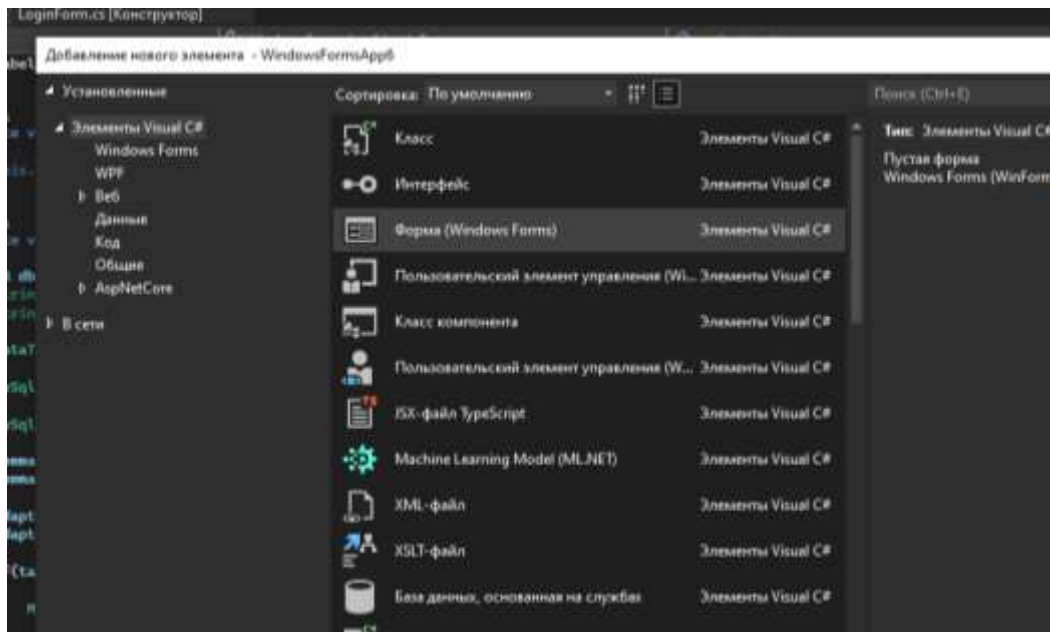


Рисунок 57 - Новая форма

Копируем содержимое LoginForm на новую созданную форму RegisterForm

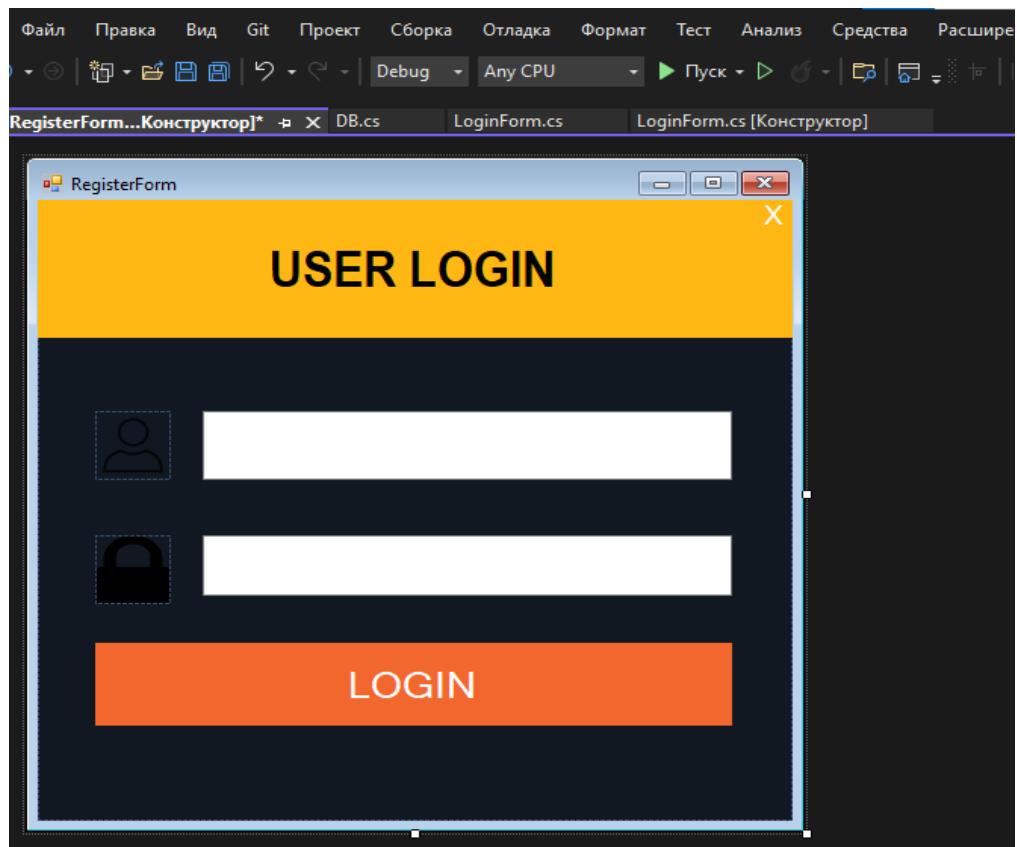


Рисунок 58 - Скопированная LoginForm

Редактируем, добавляем textbox

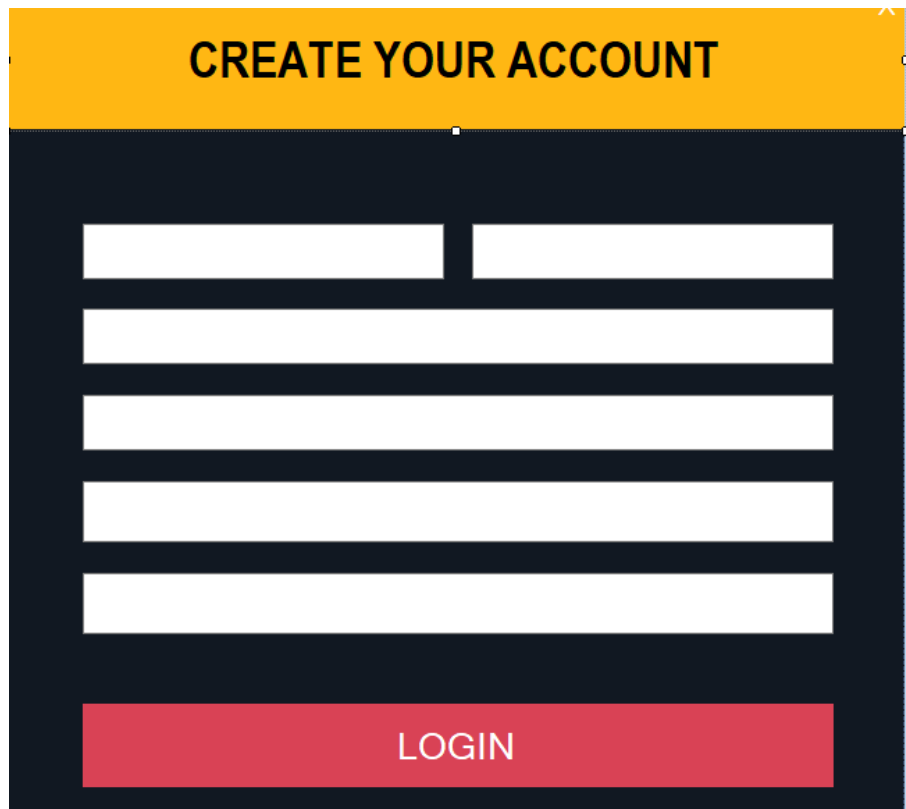


Рисунок 59 - Отредактированная форма RegisterForm

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

```

ссылка: 0
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new RegisterForm());
}

```

Рисунок 60 - Program.cs

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

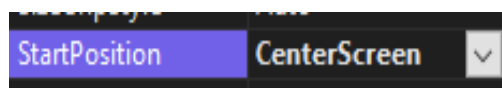


Рисунок 61 – StartPosition

Переименовываем все textbox

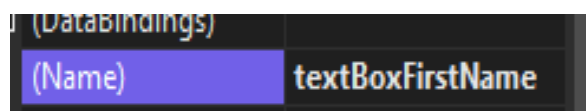


Рисунок 62 - textBoxFirstName

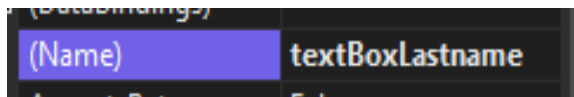


Рисунок 63 - textBoxLastName

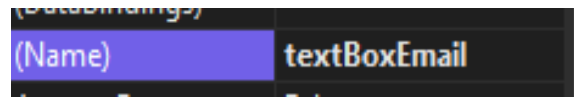


Рисунок 64 - textBoxEmail

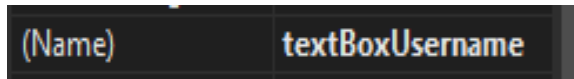


Рисунок 65 - textBoxUsername

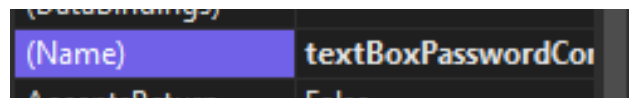


Рисунок 66 - textBoxPasswordConfirm

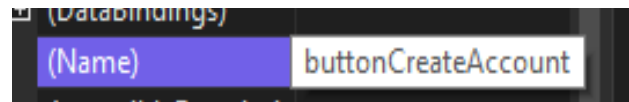
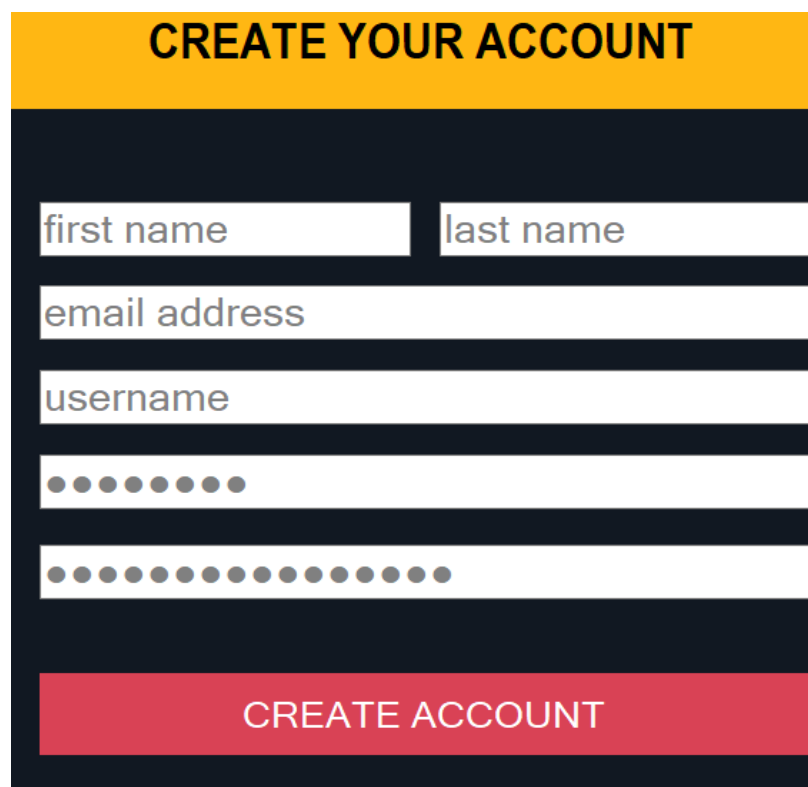


Рисунок 67 – buttonCreateAccountx

Промежуточный результат



CREATE YOUR ACCOUNT

first name last name

email address

username

••••••••

••••••••••••••••

CREATE ACCOUNT

Рисунок 68 - Промежуточный результат

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



Рисунок 69 - UseSystemPassword

×

CREATE YOUR ACCOUNT

first name

last name

email address

username

password

confirm password

CREATE ACCOUNT

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: 1
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: 1
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 текстбоксы не были активными пишем следующую функцию

```
Ссылка: 1
public RegisterForm()
{
    InitializeComponent();
    this.ActiveControl = label1;
}
```

Рисунок 73 – Активный элемент

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

```
БЛОК: 1
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;
    }
}

БЛОК: 1
private void textBoxLastName_Leave(object sender, EventArgs e)
{
    String lname = textBoxLastName.Text;
    if (lname.ToLower().Trim().Equals("last name") || lname.Trim().Equals(""))
    {
        textBoxLastName.Text = "last name";
        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

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

Ссылка: 1
private void textBoxEmail_Leave(object sender, EventArgs e)
{
    String email = textBoxEmail.Text;
    if (email.ToLower().Trim().Equals("email address") || email.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

```
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: 1
private void textBoxUsername_Leave(object sender, EventArgs e)
{
    String username = textBoxUsername.Text;
    if (username.ToLower().Trim().Equals("username") || username.Trim().Equals(""))
    {
        textBoxUsername.Text = "username";
        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

```
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: 1
private void textBoxPassword_Leave(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

```
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: 1
private void textBoxPasswordConfirm_Leave(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 = "confirm 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


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

Ссылка: 1
private void labelClose_MouseEnter(object sender, EventArgs e)
{
    labelClose.ForeColor = Color.Black;
}

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

Рисунок 79 - События кнопки закрытия приложения

Убираем у формы рамки меняя свойство FormBorderStyle

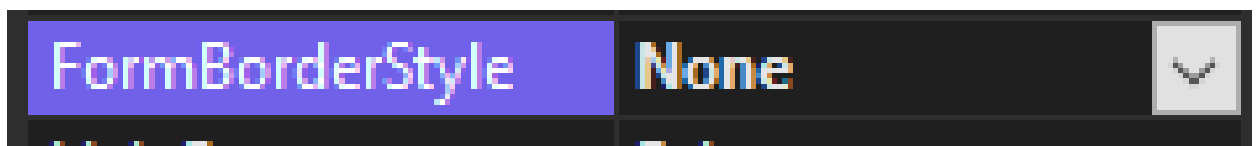


Рисунок 80 - FormBorderStyle

После выполнения всех действий получаем данный результат

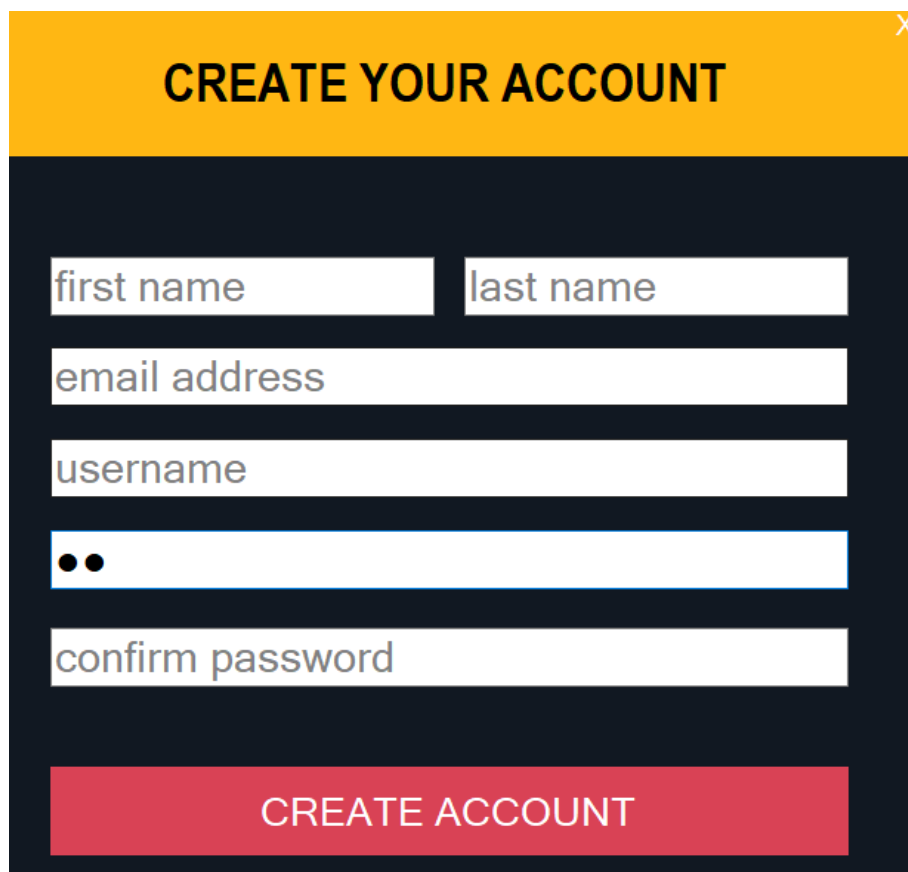


Рисунок 81 - Результат

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

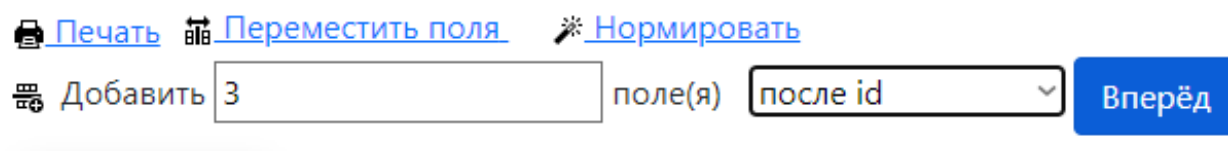


Рисунок 82 - Добавление столбцов в таблицу

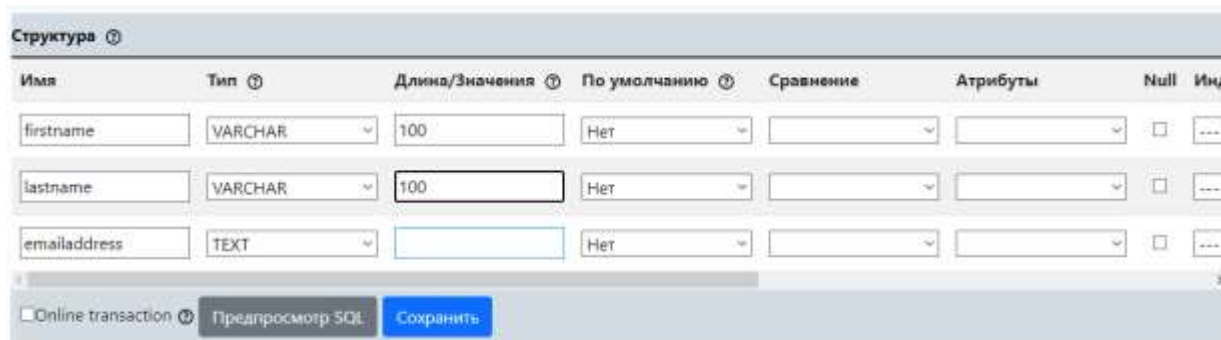


Рисунок 83 - Структура таблицы

В пространство имен подключаем MySQL


```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using MySql.Data.MySqlClient;

```

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»

```

private void buttonCreateAccount_Click(object sender, EventArgs e)
{
    DB db = new DB();
    MySqlCommand command = new MySqlCommand("INSERT INTO 'users' ('firstname', 'lastname', 'emailaddress', 'username', 'password') VALUES (@fn, @ln, @email, @usn, @pass)", db.getConnection());

    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("@usn", 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

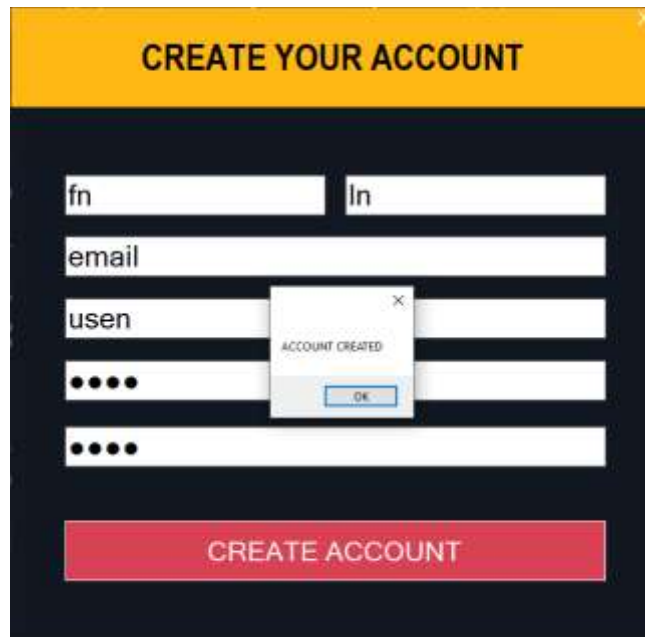


Рисунок 86 - Промежуточный результат

	id	firstname	lastname	emailaddress	username	password
<input type="checkbox"/>	1				user1	pass
<input type="checkbox"/>	2	fn	ln	email	usen	pass

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

```
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

The screenshot shows a web form titled "CREATE YOUR ACCOUNT" with a yellow header. The form has several input fields: a first name field containing "aaa", a last name field containing "aaaaa", a password field containing "aaaa", a username field containing "user1", and two masked password confirmation fields, each with five dots. A red "CREATE ACCOUNT" button is at the bottom. A white modal dialog box is centered over the form, displaying the message "This Username already exists, Select a different One" and an "OK" button.

Рисунок 91 - Создание уже существующего пользователя

Проверка создания аккаунта с данными еще существующего пользователя

The screenshot shows the same "CREATE YOUR ACCOUNT" form. The input fields now contain: first name "dddd", last name "dd", password "ddd", username "user3", and two masked password confirmation fields, each with two dots. The red "CREATE ACCOUNT" button is at the bottom. A white modal dialog box is centered over the form, displaying the message "ACCOUNT CREATED" and an "OK" button.

Рисунок 92 - Создание аккаунта

Добавление нового пользователя в базе данных




	id	firstname	lastname	emailaddress	username	password
<input type="checkbox"/>  	1				user1	pass
<input type="checkbox"/>  	2	fn	ln	email	usen	pass
<input type="checkbox"/>  	5	ddddd	dd	ddd	user3	22

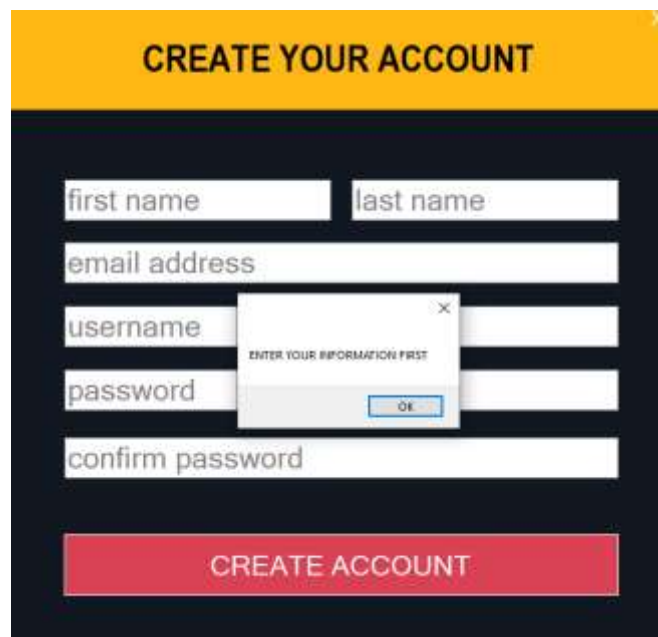
Рисунок 93 - Новый пользователь в бд

Пишем функцию, в которой проверяется введены какие-то данные пользователем или нет, если нет, то выводится меседжбокс 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 has several input fields: "first name", "last name", "email address", "username", "password", and "confirm password". A red "CREATE ACCOUNT" button is at the bottom. A small white dialog box with a close button (X) is overlaid on the form, displaying the message "ENTER YOUR INFORMATION FIRST" and an "OK" button. This indicates that the form validation failed because the 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(!checkTextBoxesValues())
{
    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 CONFIRM PASSWORD");
    }
}
```

Рисунок 96 - Проверка пароля

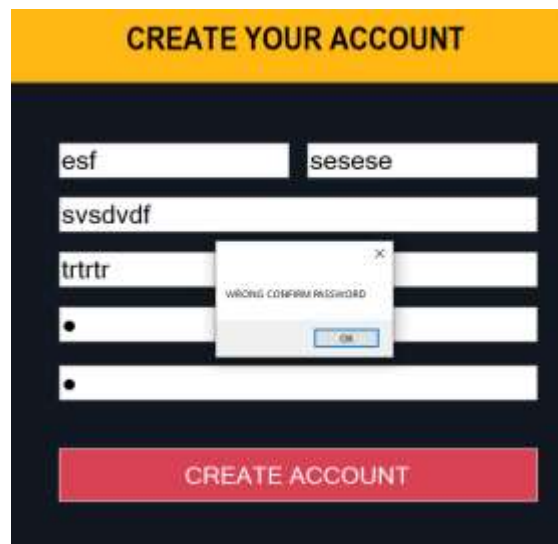


Рисунок 97 – Отрицательный результат проверки пароля

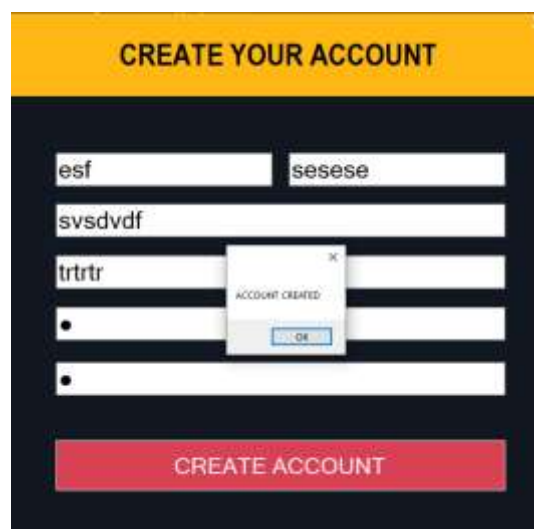


Рисунок 98 – Положительный результат проверки пароля

Добавляем в месседжбокс, 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(!checkTextBoxValues())
{
    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

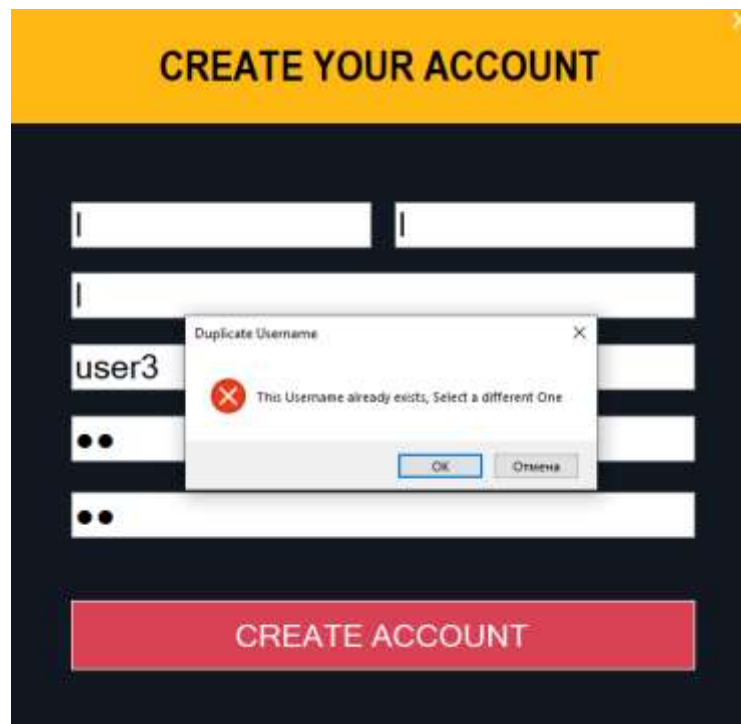


Рисунок 100 - Готовые MessageBox с ошибкой

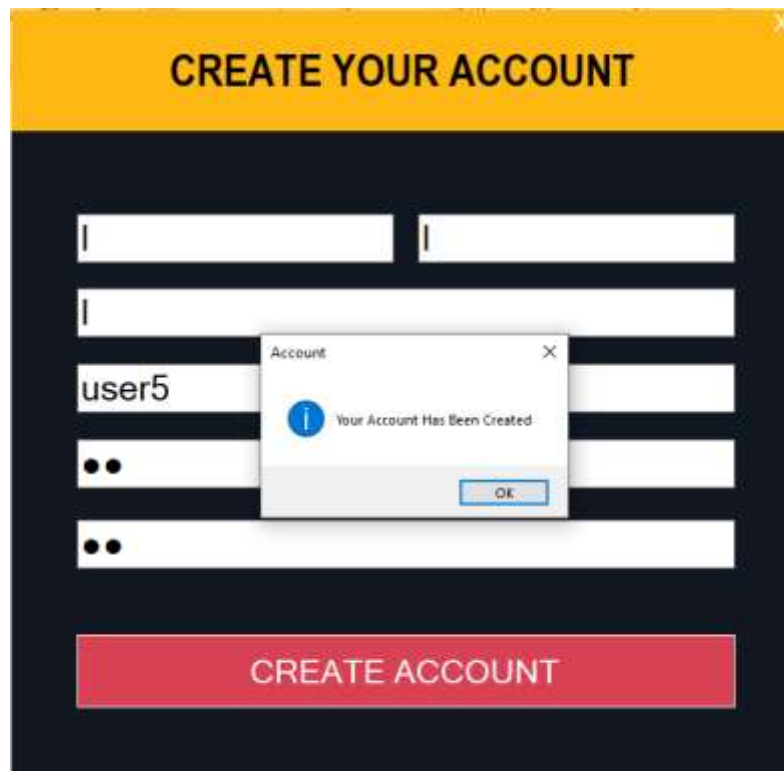


Рисунок 101 - Готовые MessageBox без ошибки

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

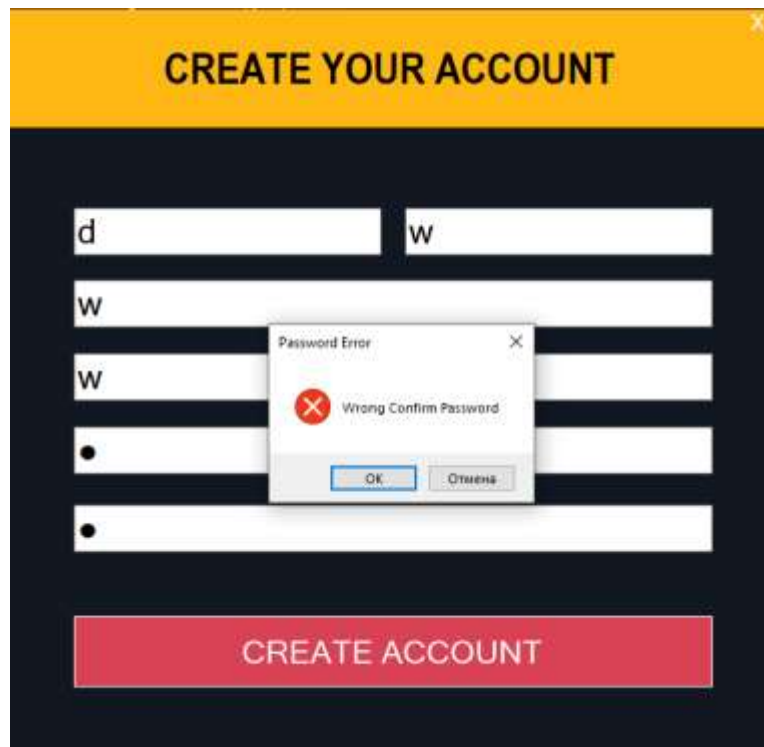


Рисунок 103 – Готовый MessageBox с ошибкой

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

The image shows a web form titled "CREATE YOUR ACCOUNT" with a yellow header. The form has a dark blue background and contains several input fields: "first name", "last name", "email address", "username", "password", and "confirm password". At the bottom is a red "CREATE ACCOUNT" button. A modal error box is displayed in the center, titled "Empty Data" with a close button (X). It contains a red circle with a white "X" icon and the text "Enter Your Informations First". At the bottom of the modal are two buttons: "OK" and "Отмена" (Cancel).

Рисунок 105 - Готовый MessageBox с ошибкой

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

```

COMMON 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
    {
        if (username.Trim().Equals(""))
        {
            MessageBox.Show("Enter Ypour Username To Login", "Empty Username", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }

        else if (password.Trim().Equals(""))
        {
            MessageBox.Show("Enter Ypour 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.

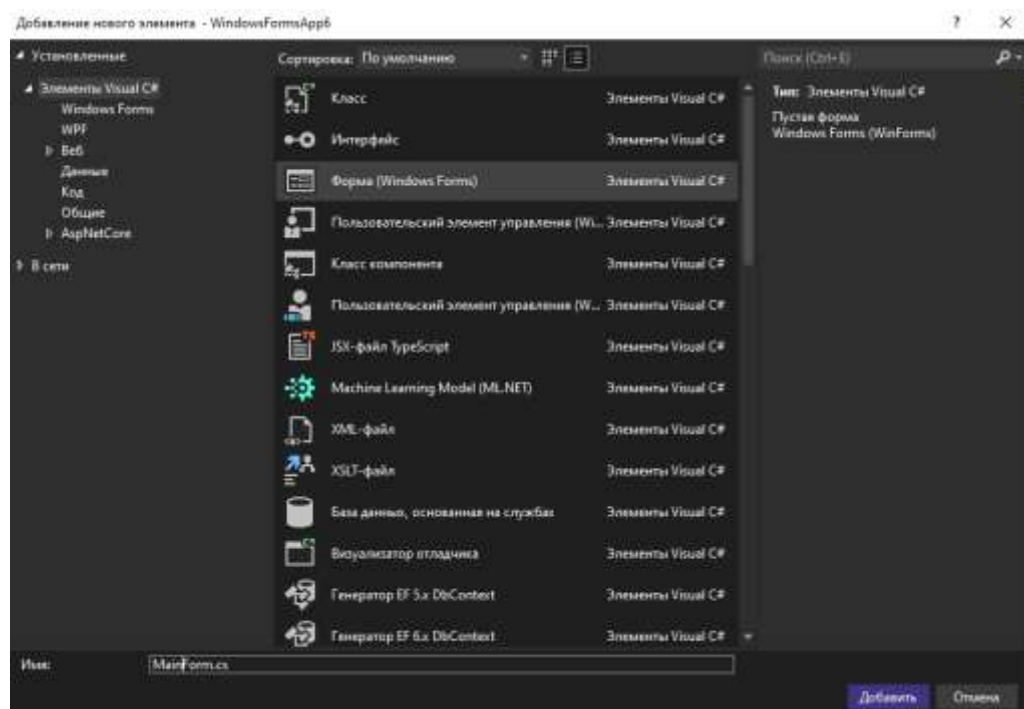


Рисунок 107 - Создание новой формы

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

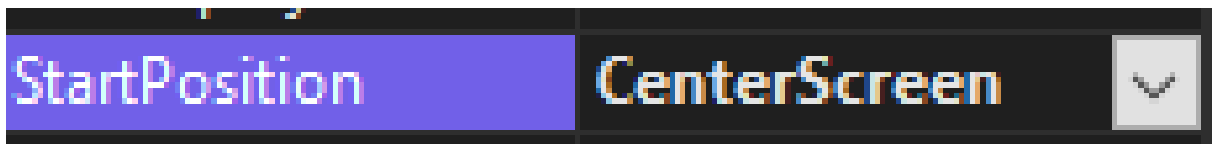


Рисунок 110 - Свойство StartPosition

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

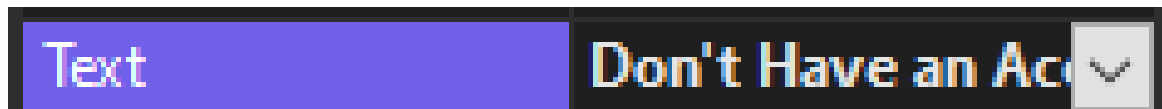


Рисунок 111 - Текст label

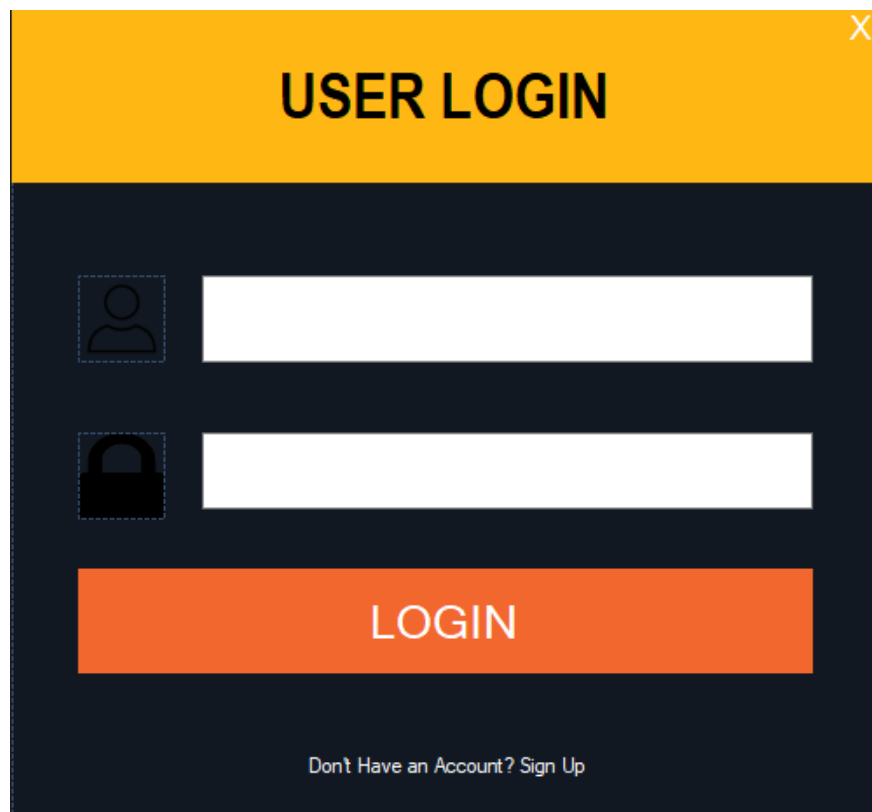


Рисунок 112 - Отредактированная LoginForm

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

```

Ссылка: 1
private void labelGoSingUp_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.

```

Ссылка: 0
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new LoginForm());
}

```

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.

```

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 labelGoSingUp_MouseEnter(object sender, EventArgs e)
{
    labelGoSingUp.ForeColor = Color.Yellow;
}

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

```

Рисунок 117 - Событие labelGoSingUp

В настройках шрифта у labelGoSingUp ставим галочку напротив «Подчеркнутый»

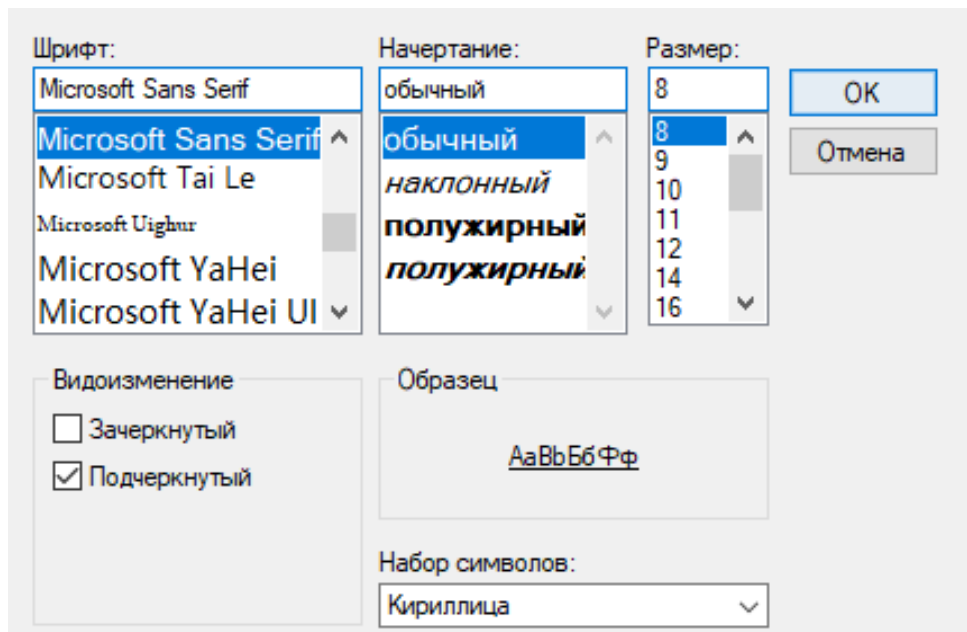


Рисунок 118 - Настройки шрифта

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.


```

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

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

```

Рисунок 121 - Событие смены цвета при наведении на label

```

private void labelGoToLogin_Click(object sender, EventArgs e)
{
    this.Hide();
    LoginForm loginform = new LoginForm();
    loginform.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();
    MainForm mainform = new MainForm();
    mainform.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:

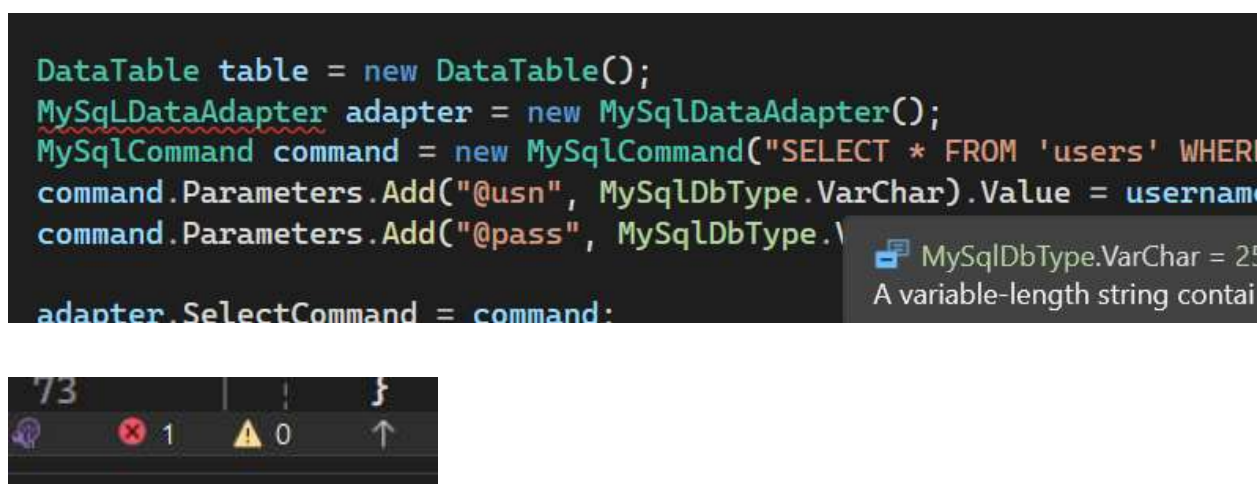
Отладка программных модулей - это процесс поиска и устранения ошибок в коде, чтобы программа работала корректно.

Цель:

Главная цель отладки - сделать программу надежной и безошибочной.

Задание:

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



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

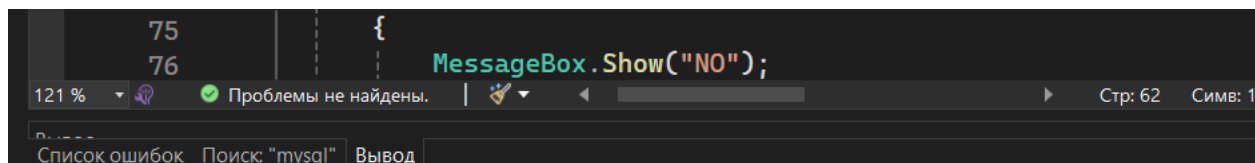
```
MySqlDataAdapter adapter = new MySqlDataAdapter();
```

Синтаксическая ошибка

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

```
string username = textBox1.Username.Text;  
string password = textBox2.Password.Text;  
  
DataTable table = new DataTable();  
MySqlDataAdapter adapter = new MySqlDataAdapter();  
MySqlCommand command = new MySqlCommand("SELECT * FROM 'users' WHERE username = @usn AND password = @pass", connection);  
command.Parameters.Add("@usn", MySqlDbType.VarChar).Value = username;  
command.Parameters.Add("@pass", MySqlDbType.VarChar).Value = password;  
  
adapter.SelectCommand = command;  
adapter.Fill(table);
```

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



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

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.