

Prilozhenie 1.1.

1. Создание проекта

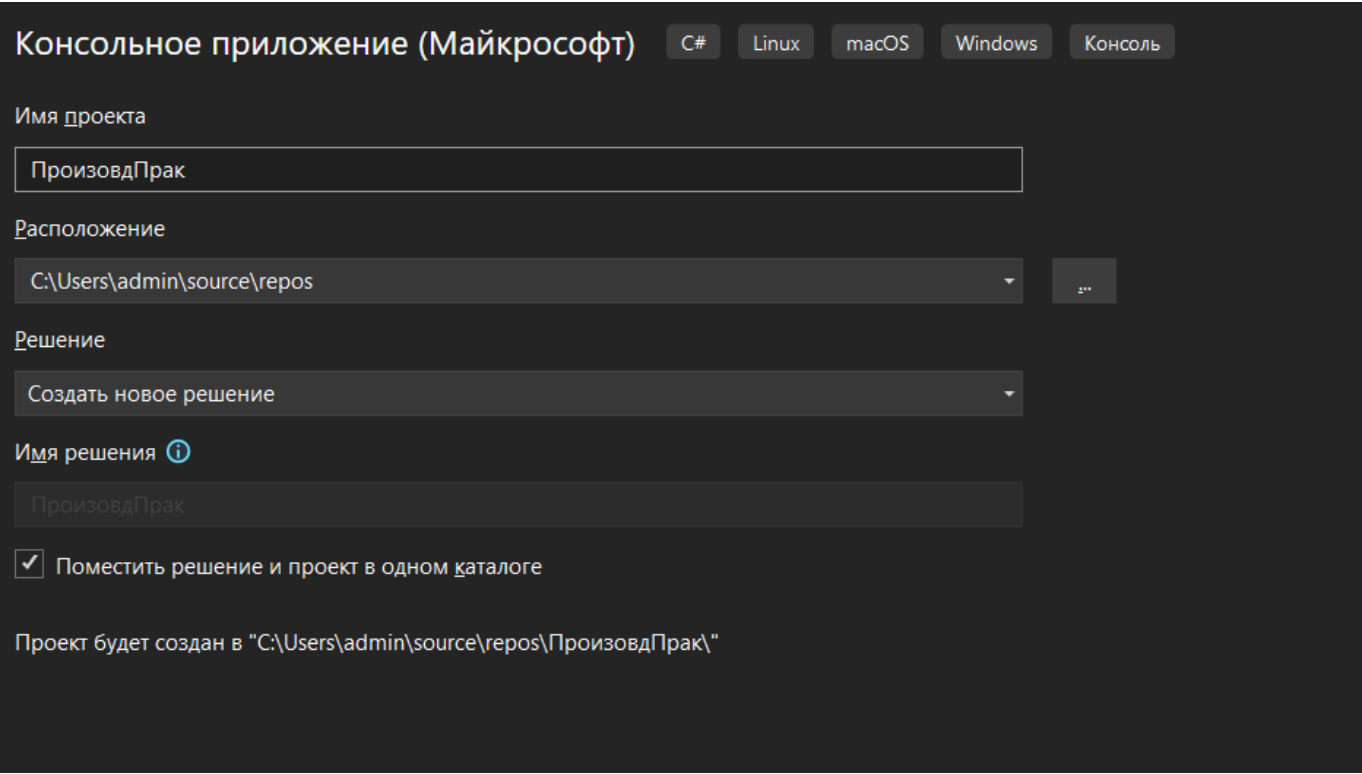


Рисунок 1 - Создание проекта

2. Созданный проект

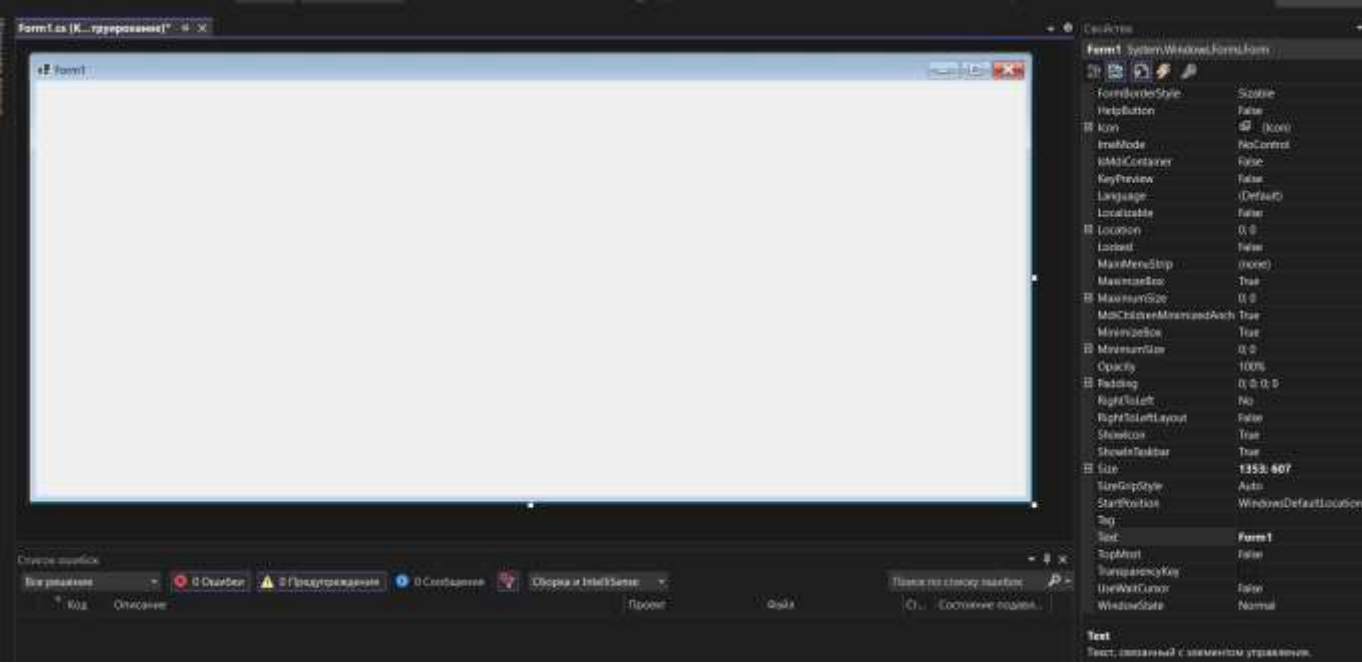


Рисунок 2 - Созданный проект

3. Удаление Form1

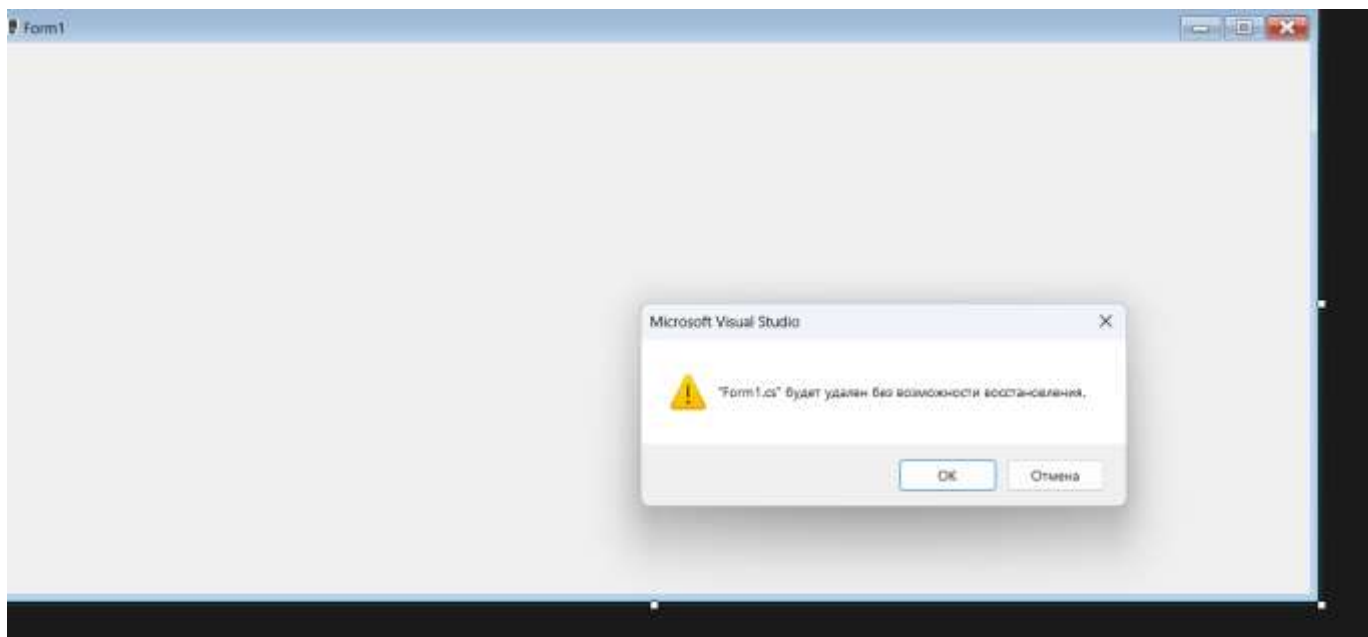


Рисунок 3 - Удаление формы

4. Создание LoginForm

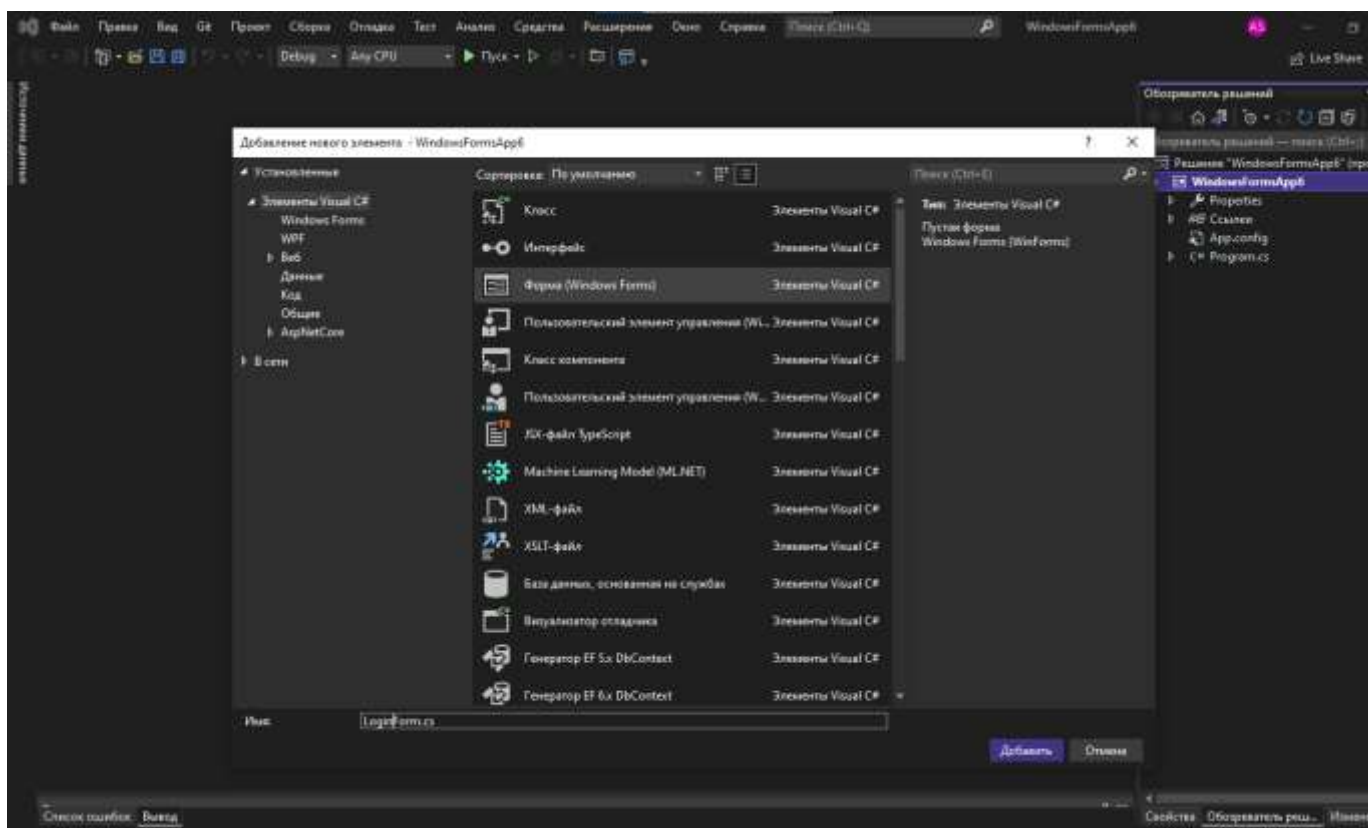


Рисунок 4 - Создание формы

5. Открываем панель элементов

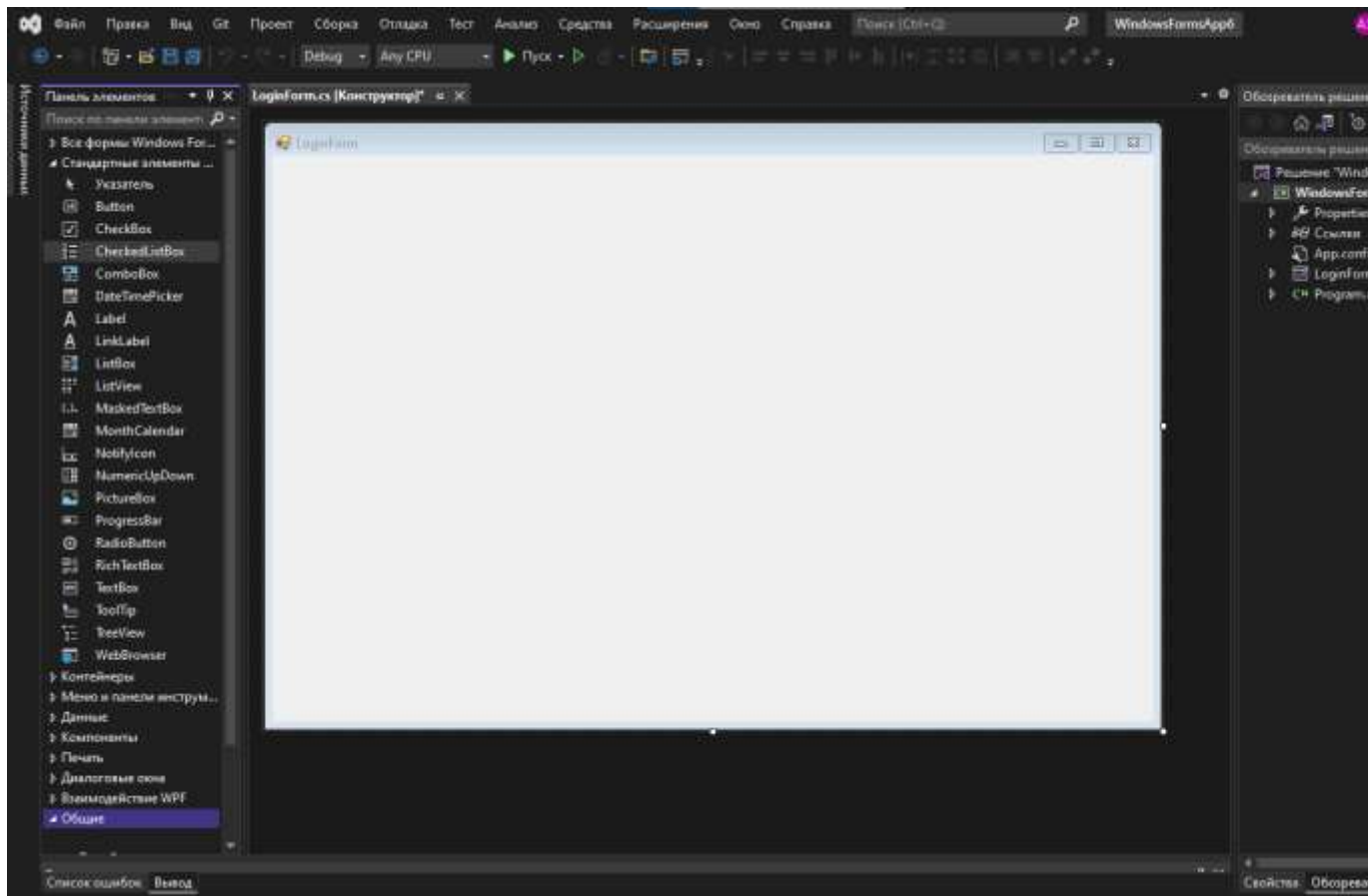


Рисунок 5 - Панель элементов

6. Выбираем в панели элементов элемент panel

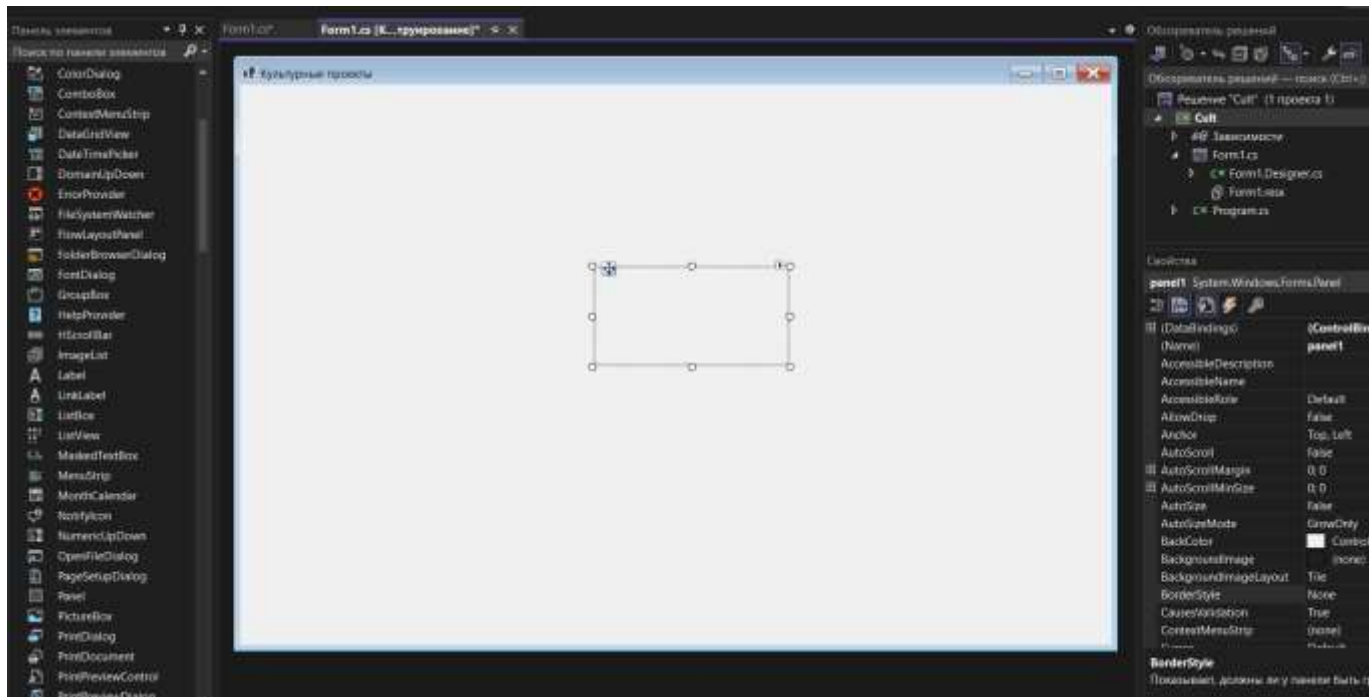


Рисунок 6 - Элемент panel

7. Закрепляем panel в родительском контейнере

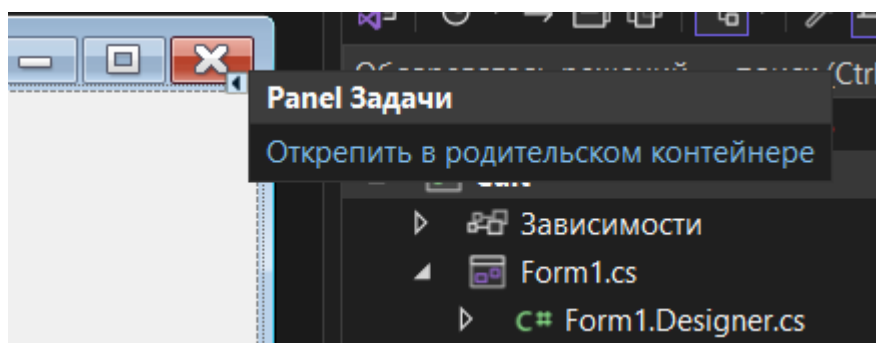


Рисунок 7 – Закрепление panel

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

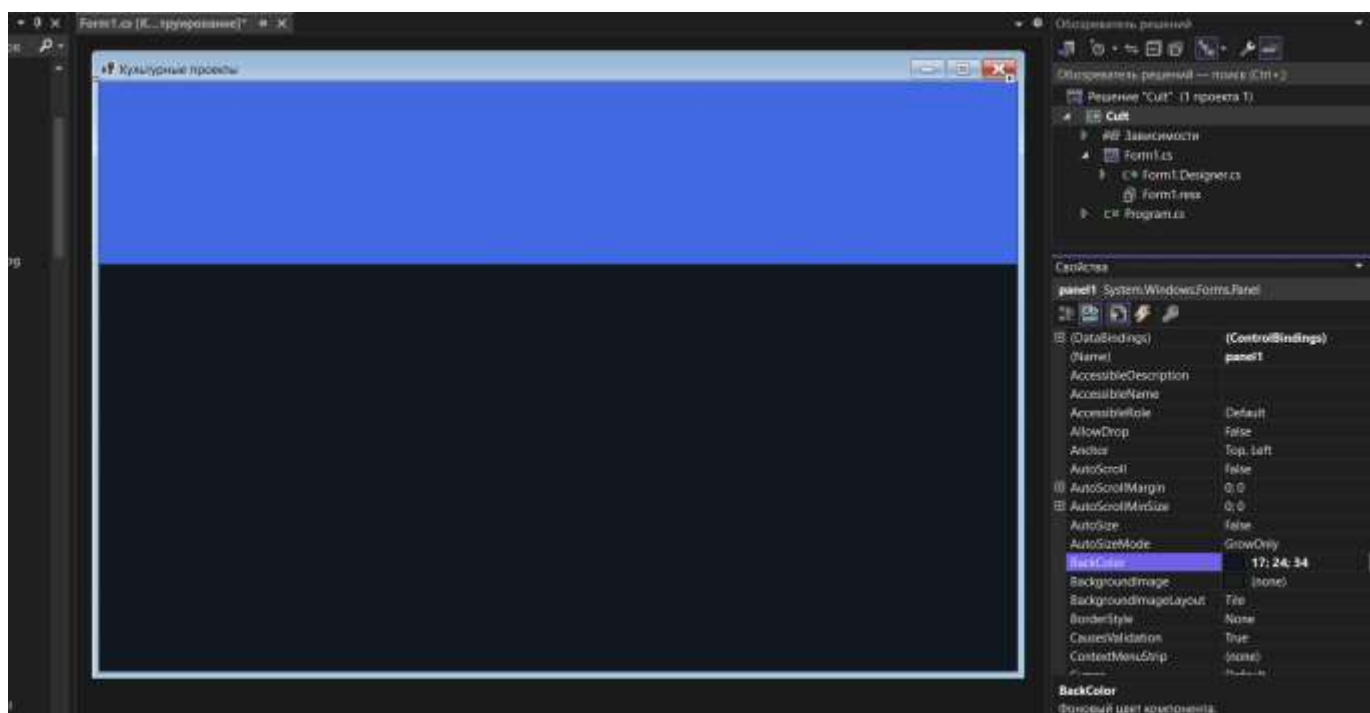


Рисунок 8 - Цвет панели

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

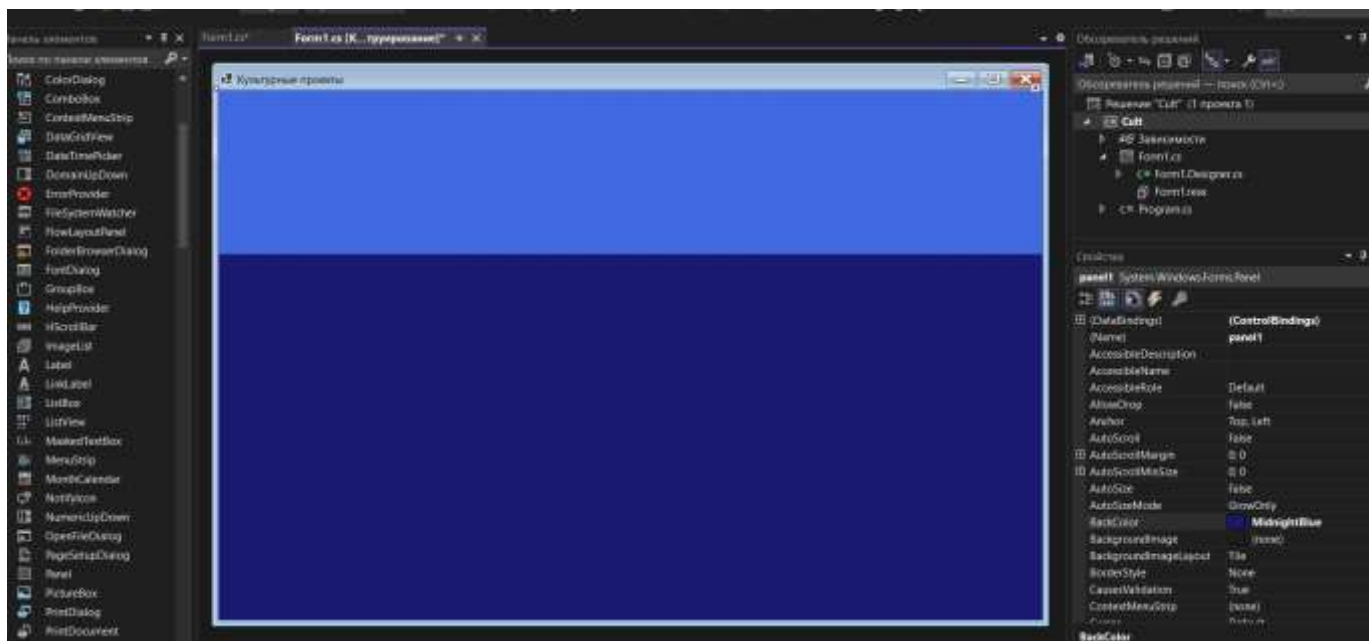


Рисунок 9 - Создание второй панели

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

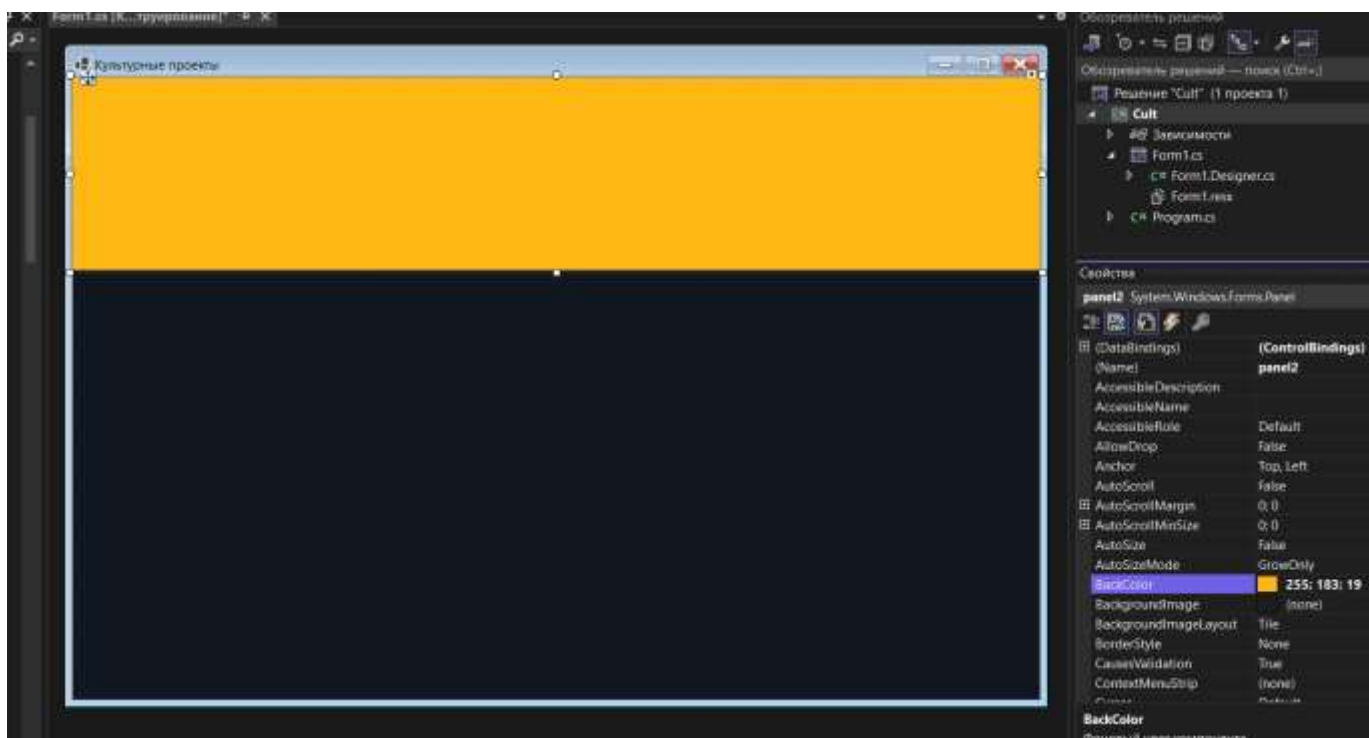


Рисунок 10 - Цвет панели

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

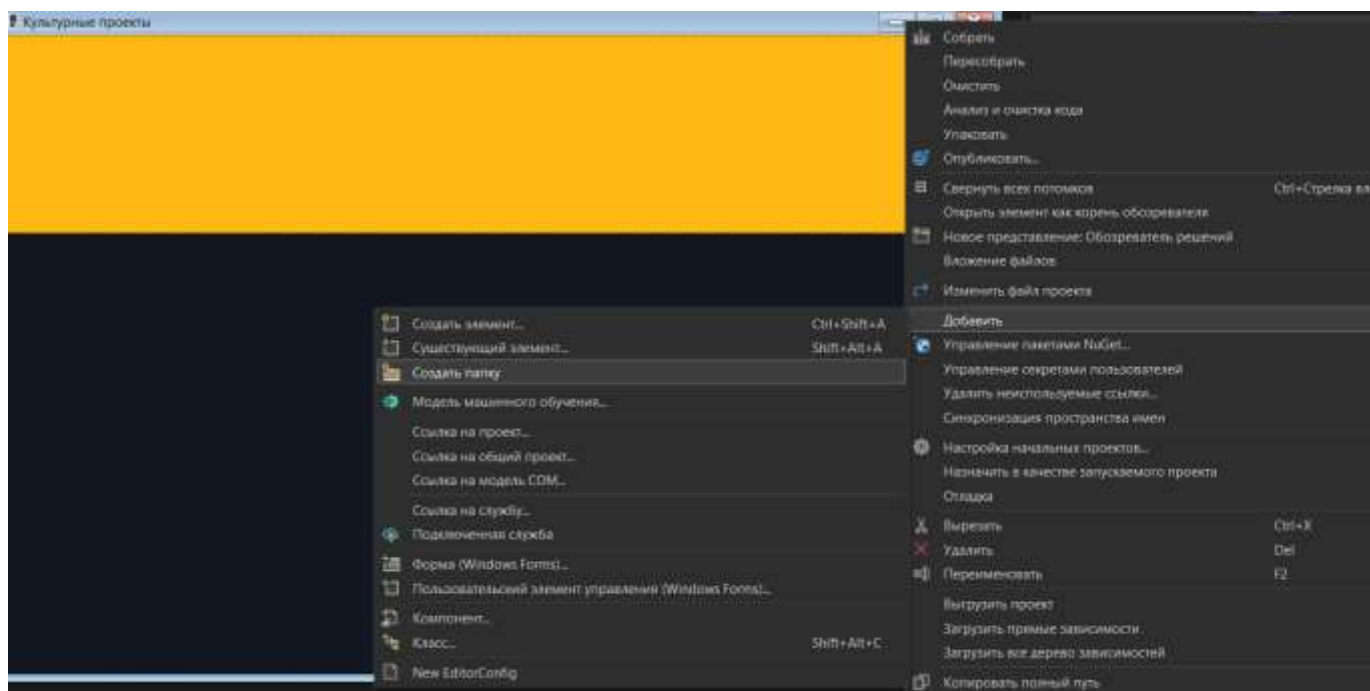


Рисунок 11 - Создание папки

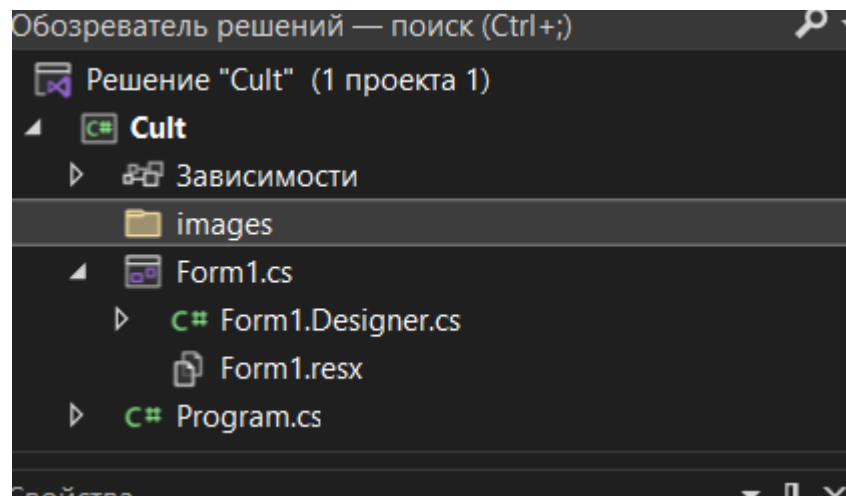


Рисунок 12 - Созданная папка

12. Создаем label и настраиваем шрифт и текст

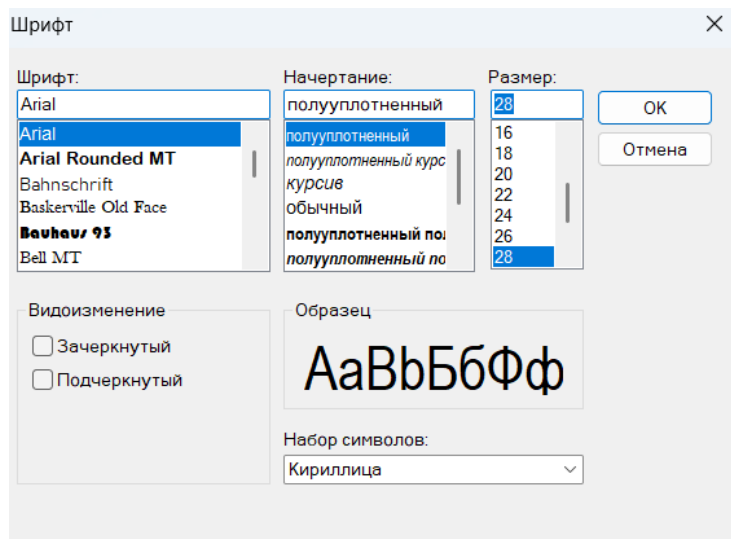


Рисунок 13 - Настройки шрифта

Tag	
Text	USER LOGIN
TextAlign	TopLeft
UseCompatibleTextRenderir	False
UseMnemonic	True

Рисунок 14 - Текст label

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



Рисунок 15 - Свойство Dock

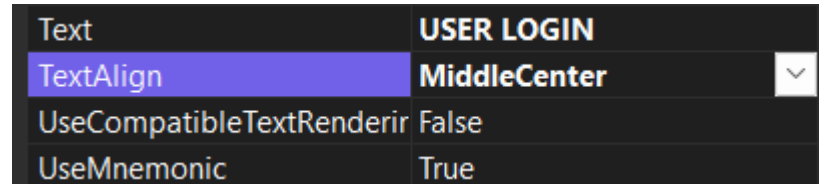


Рисунок 16 - Свойство TextAlign

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

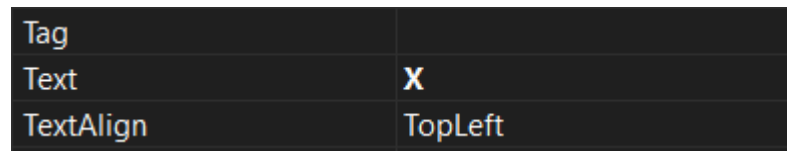


Рисунок 17 - Текст label

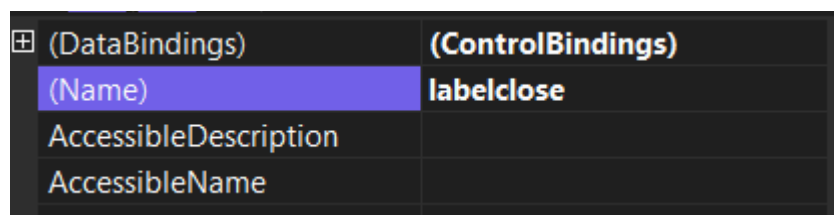


Рисунок 18 - Имя label

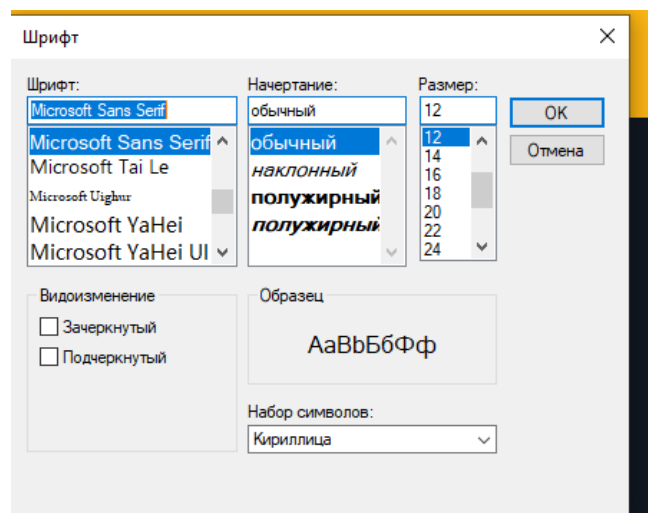


Рисунок 19 - Настройки шрифта

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

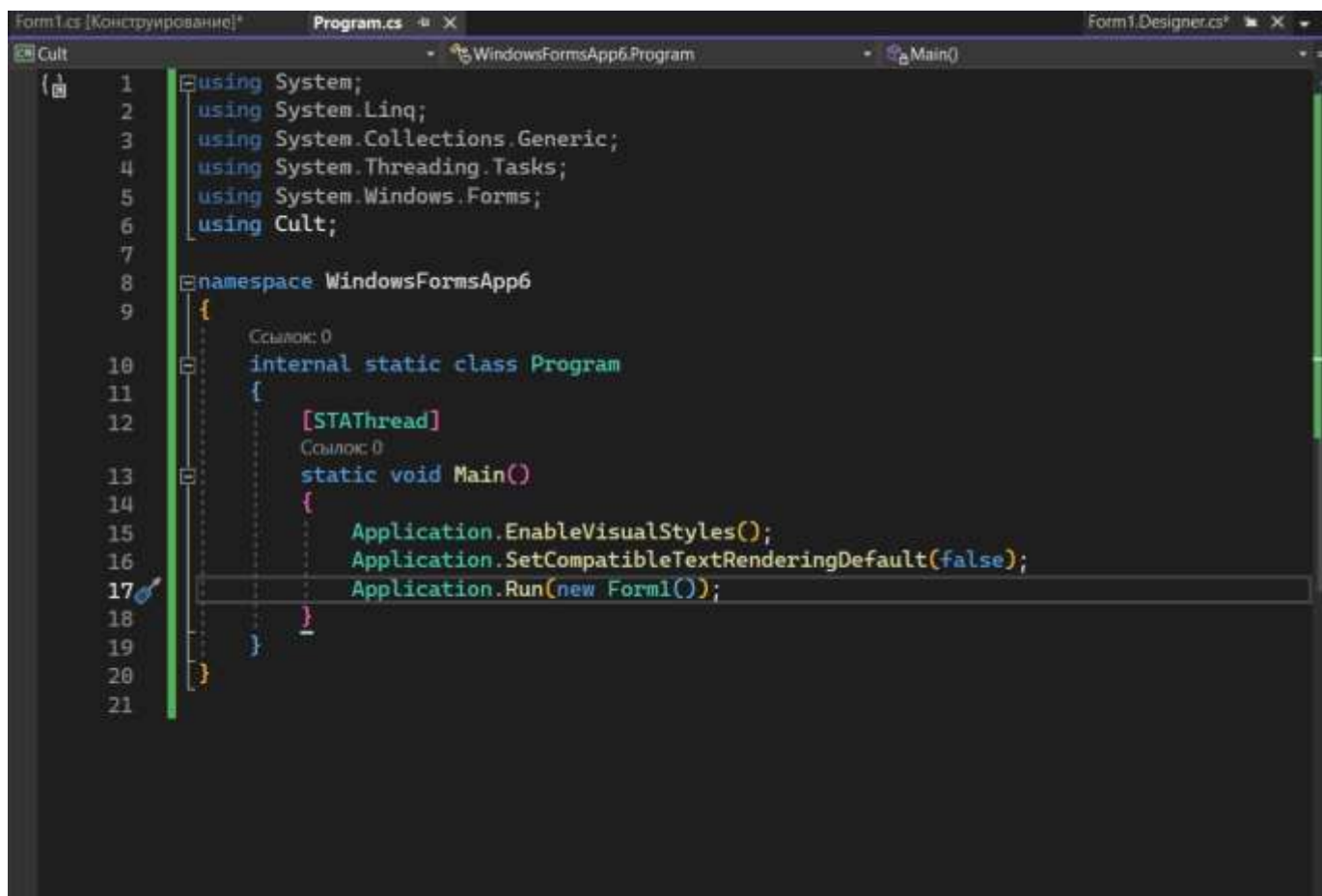


Рисунок 20 - Program.cs

16. Промежуточный результат

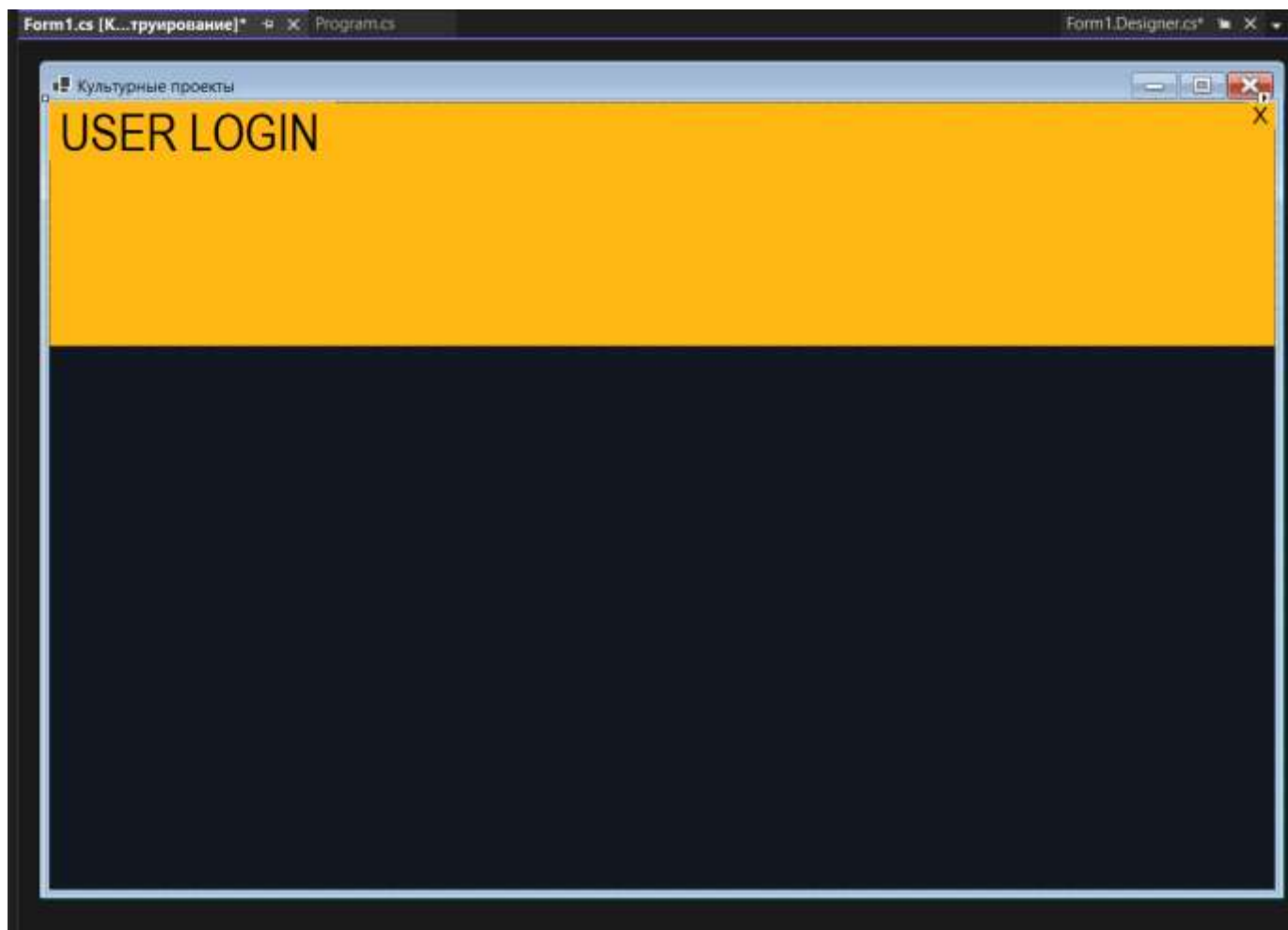


Рисунок 21 - Промежуточный результат

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

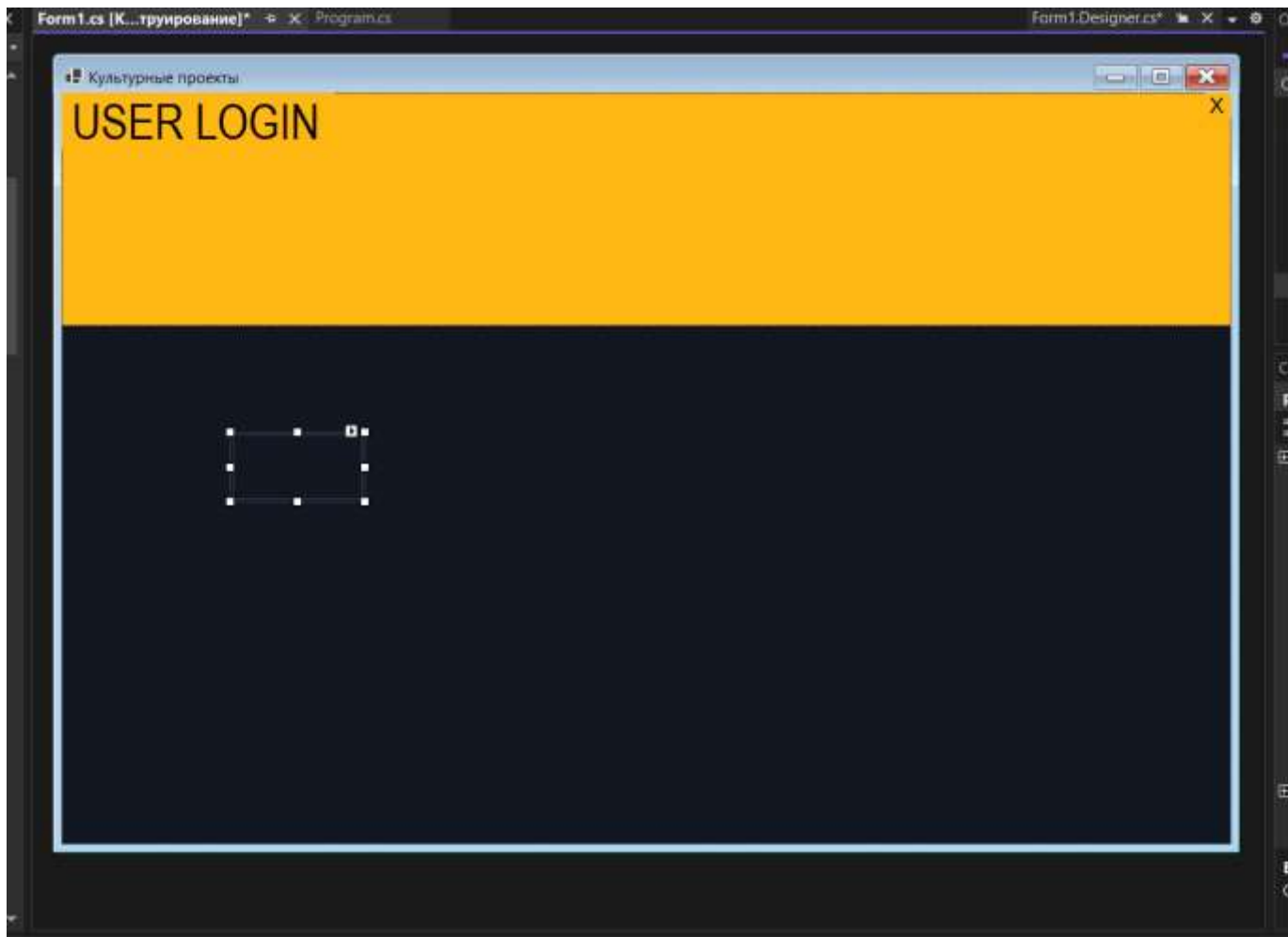


Рисунок 22 - pictureBox

Modifiers	Private
Padding	0; 0; 0; 0
Size	50; 50
SizeMode	Normal
Tag	
UseWaitCursor	False

Рисунок 23 - Размер pictureBox

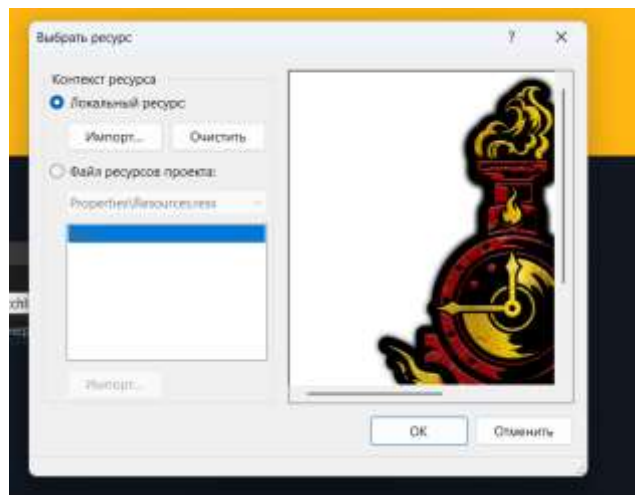


Рисунок 24 - Выбор изображения для pictureBox

18. Готовый pictureBox

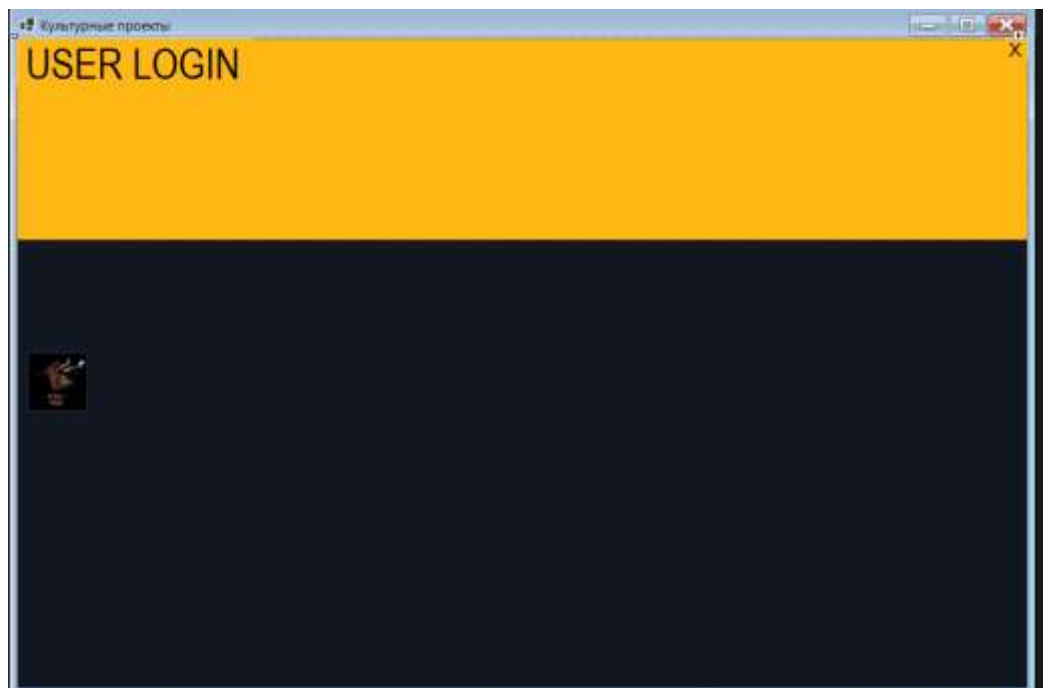


Рисунок 25 - Готовый pictureBox

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

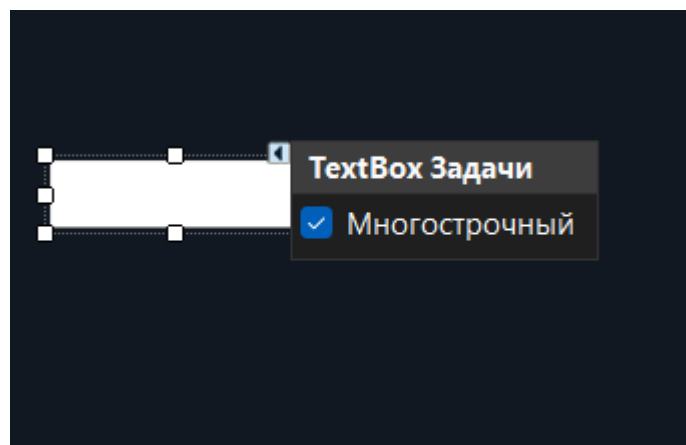


Рисунок 26 - Свойство MultiLine

RightToLeft	No
ScrollBars	None
ShortcutsEnabled	True
Size	350; 50
TabIndex	3
TabStop	True
Tag	

Рисунок 27 - Размеры textbox

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

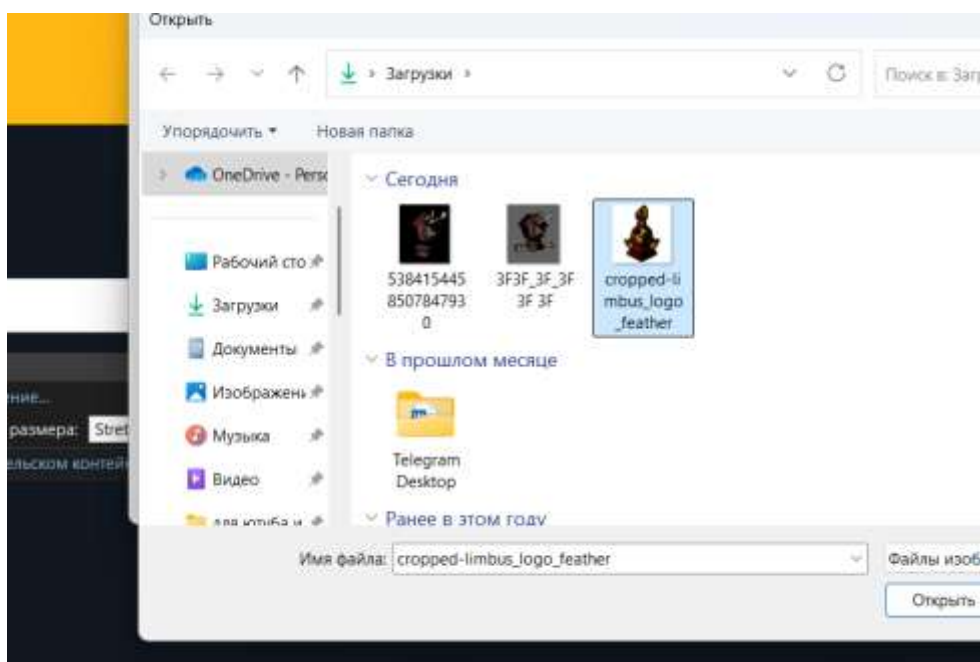


Рисунок 28 - Выбор изображения для pictureBox

21. Промежуточный результат

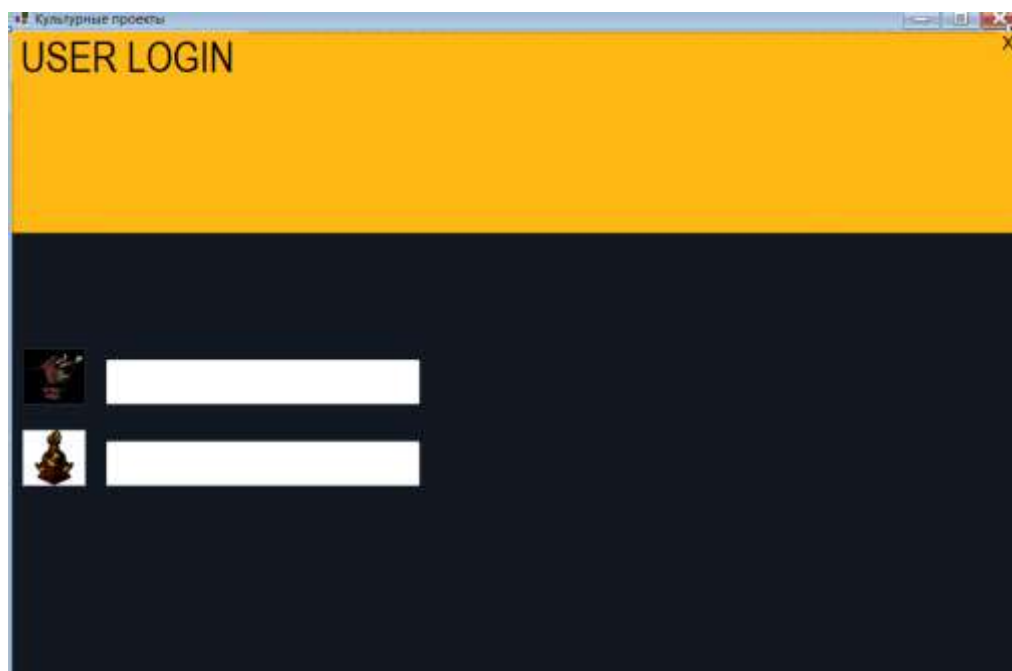


Рисунок 29 - Промежуточный результат

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

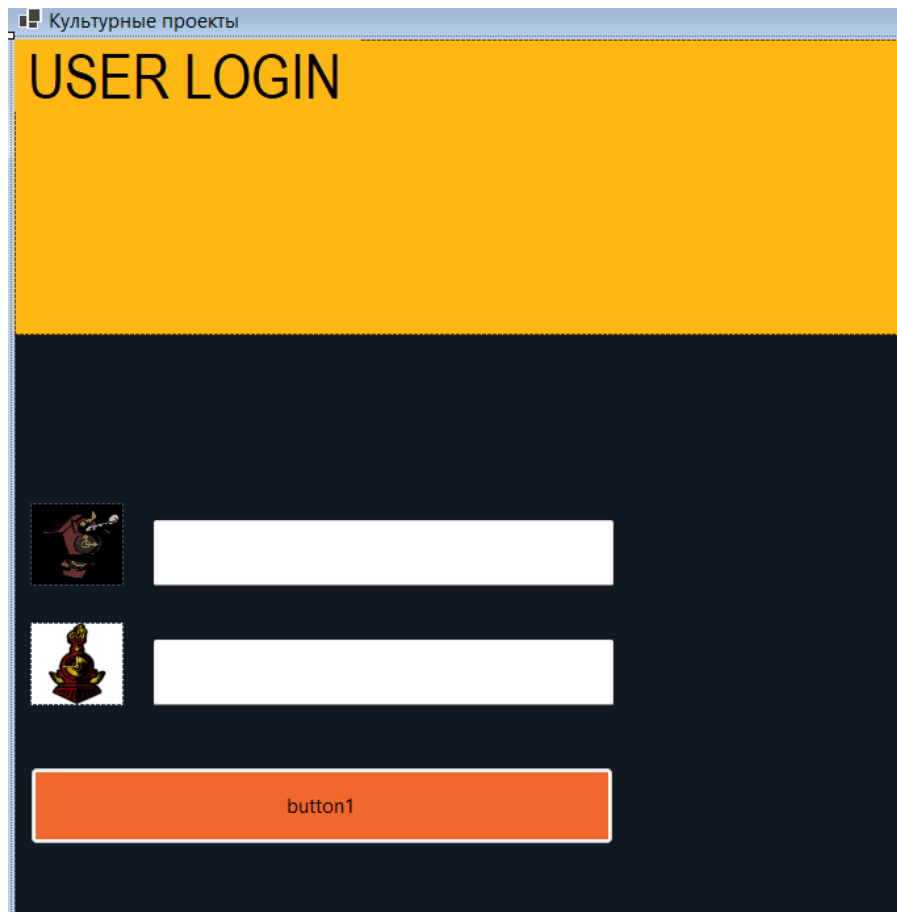


Рисунок 30 - Кнопка

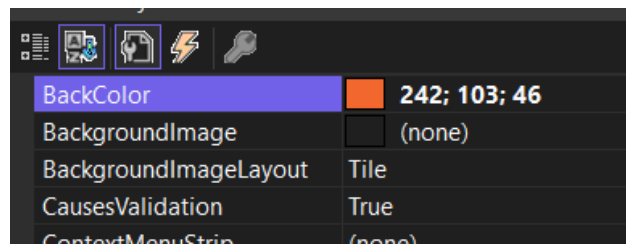


Рисунок 31 - Цвет кнопки

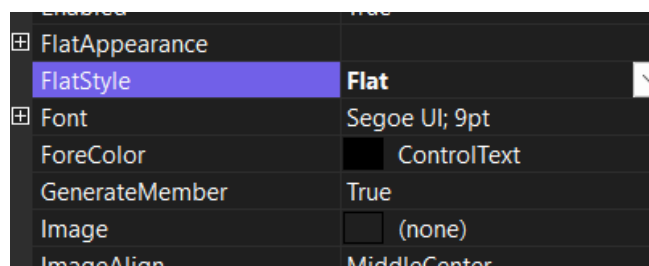


Рисунок 32 - Свойство FlatStyle

23. Настраиваем шрифт кнопки и цвет текста

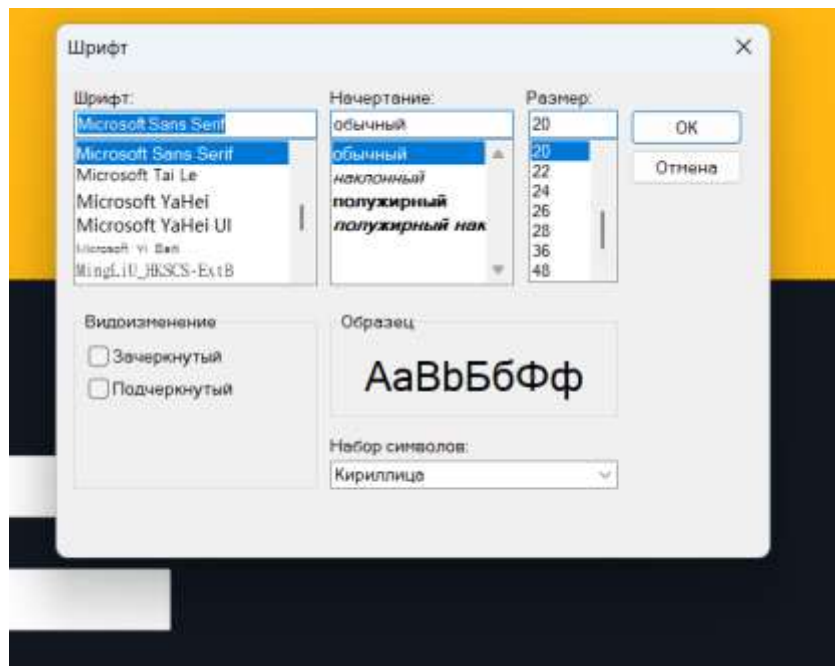


Рисунок 33 - Настройки шрифта

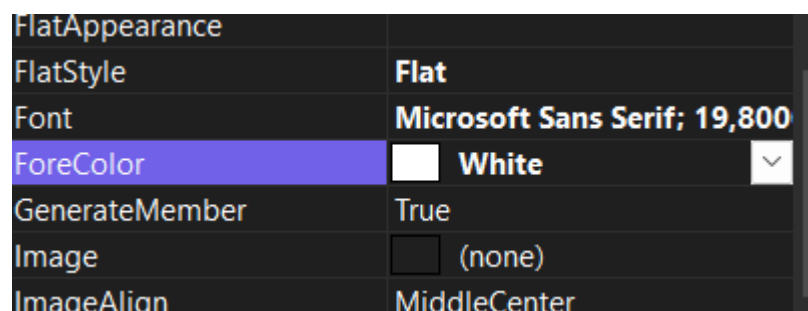


Рисунок 34 - Цвет текста

24. Убираем контур у кнопки

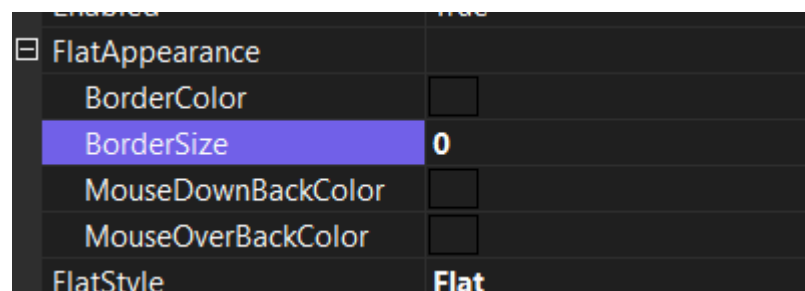


Рисунок 35 - Контур кнопки

25. У текстовых настроек настраиваем шрифты

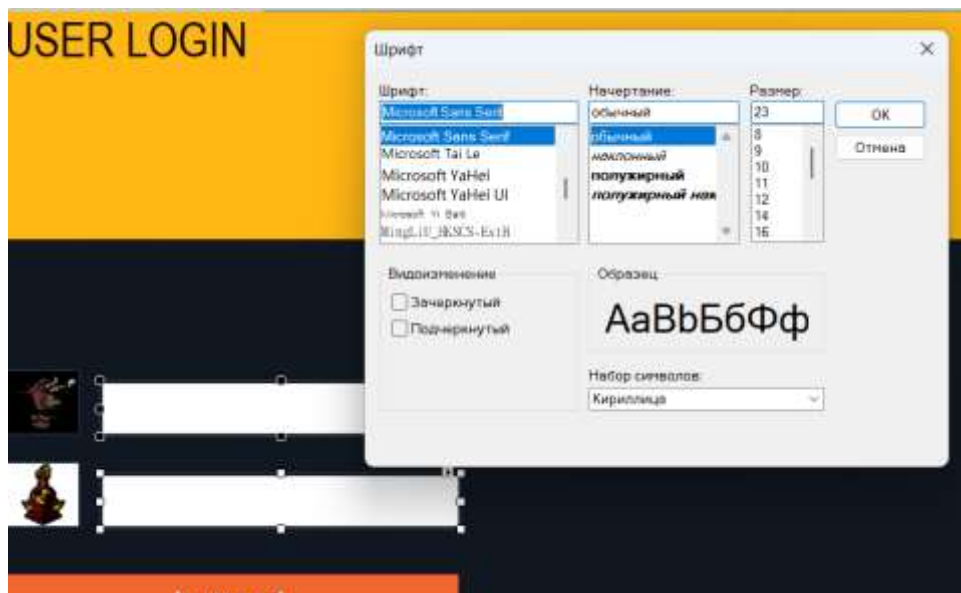


Рисунок 36 - Настройка шрифта

26. Далее переименовываем элементы

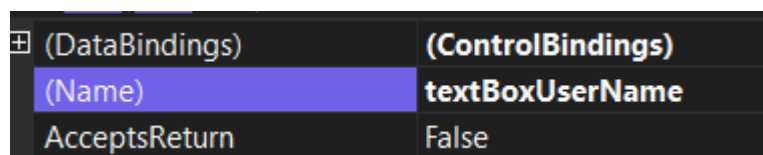


Рисунок 37 - Название textBoxName

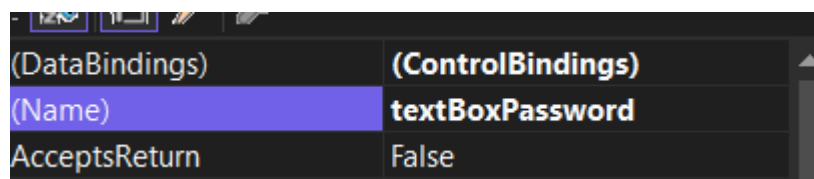


Рисунок 38 - Название textBoxPassword

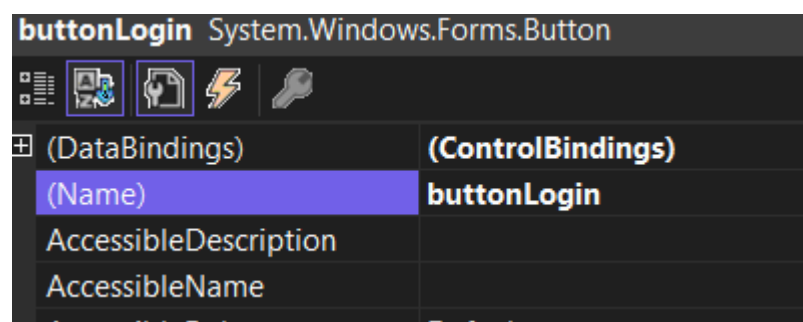


Рисунок 39 - Название buttonLogin

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

Modifiers	Private
Multiline	True
PasswordChar	

Рисунок 40 - Свойство UseSystemPassword

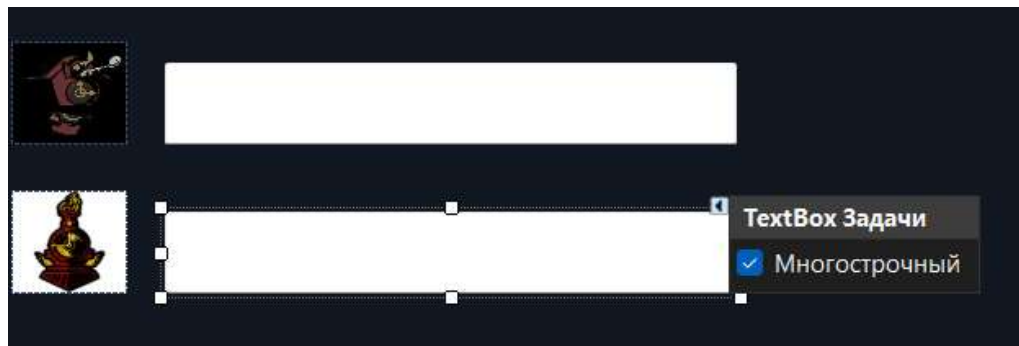


Рисунок 41 - Свойство MultiLine

28. Промежуточный результат

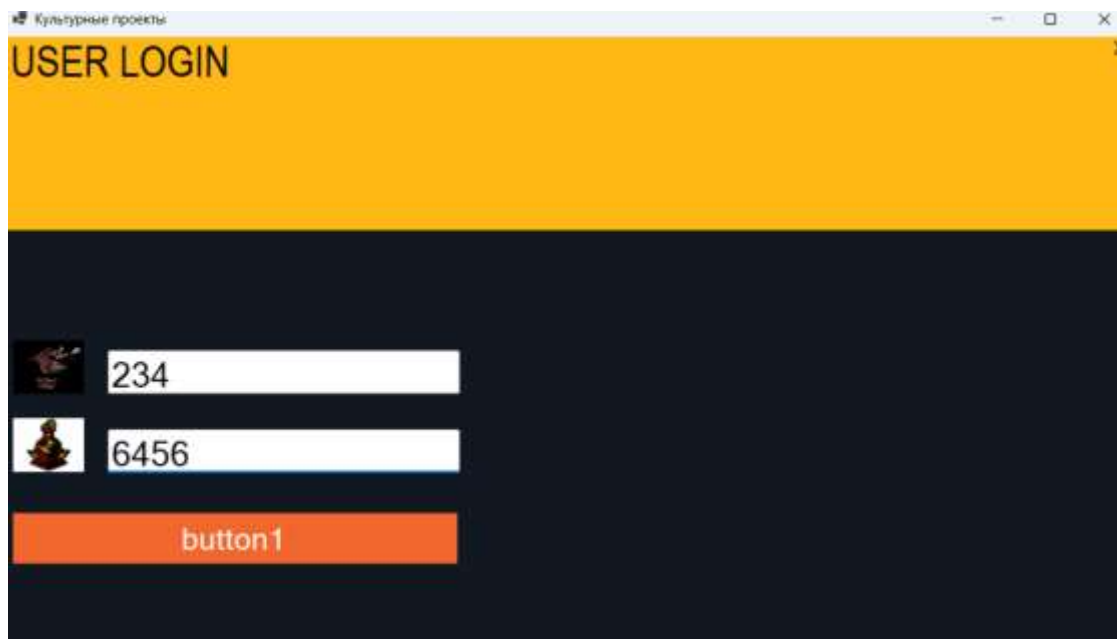


Рисунок 42 - Промежуточный результат

29. Переходим к коду, задаем textBoxPassword размер

```
Form1.cs [Конструирование] Program.cs X Components [SourceLink] X
1 using System;
2 using System.Linq;
3 using System.ComponentModel;
4 using System.Data;
5 using System.Drawing;
6 using System.Text;
7 using System.Collections.Generic;
8 using System.Threading.Tasks;
9 using System.Windows.Forms;
10 using System.Drawing.Text;
11
12 namespace WindowsFormsApp6
13 {
14     public partial class LoginForm : Form
15     {
16         private object textBoxPassword;
17
18         public LoginForm(object textBoxPassword)
19         {
20             InitializeComponent();
21             this.textBoxPassword.AutoSize = false;
22             this.textBoxPassword.Size = new Size(this.textBoxPassword.Size.Width, 50);
23             this.textBoxPassword = textBoxPassword;
24         }
25     }
26 }
```

Рисунок 43 - Размер textbox

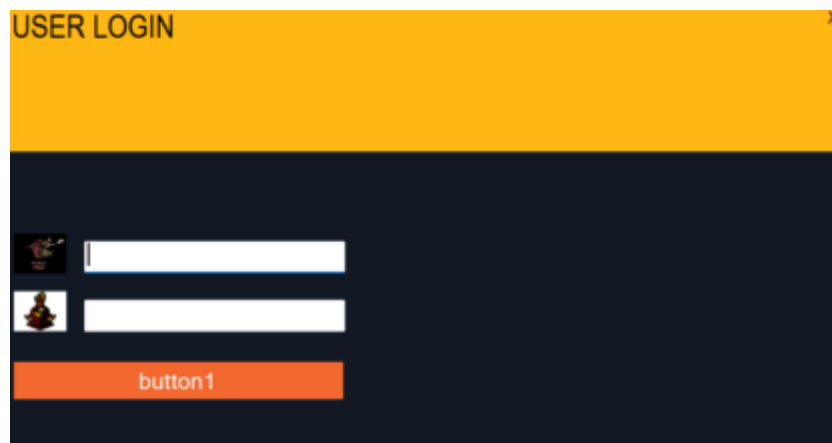


Рисунок 44 - Промежуточный результат

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

Size	1140, 772
SizeGripStyle	Auto
StartPosition	CenterScreen
Tag	

Рисунок 45 - Свойство CenterScreen

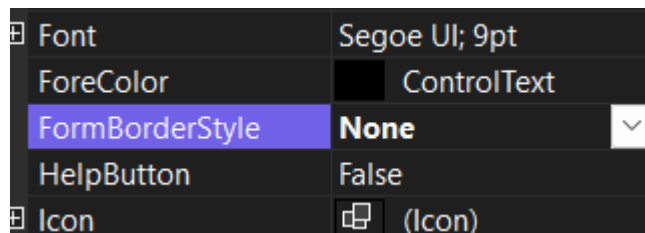


Рисунок 46 - Свойство FormBorderStyle

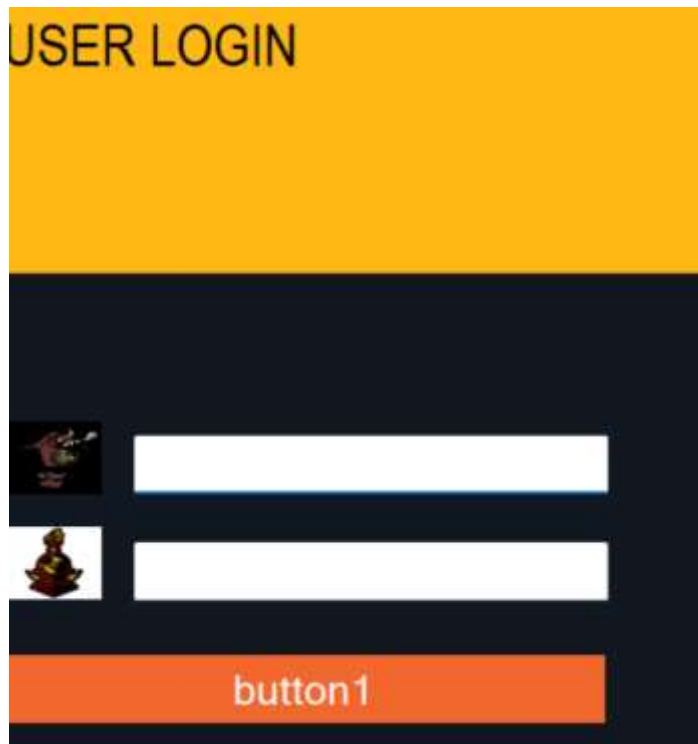


Рисунок 47 - Промежуточный результат

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



Рисунок 48 - Свойство Cursor

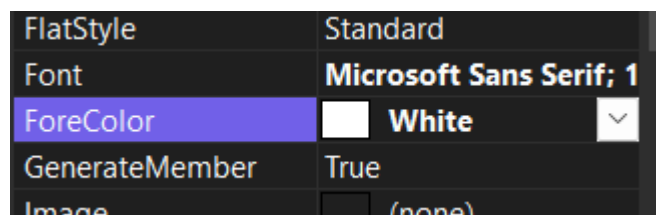


Рисунок 49 - Цвет

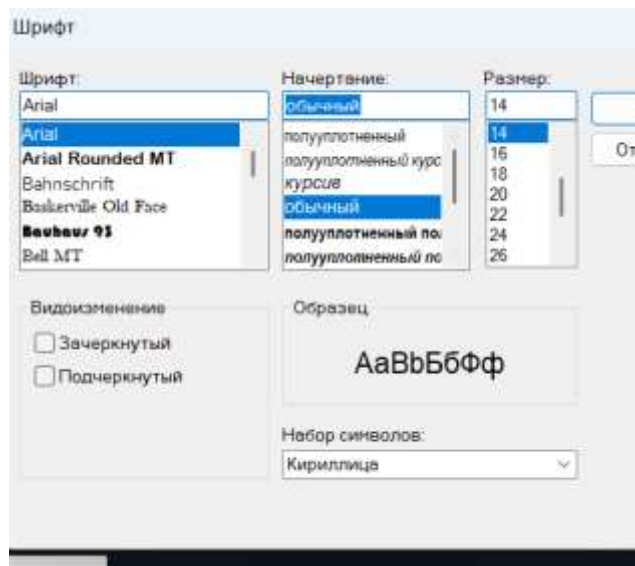


Рисунок 50 - Настройка шрифта

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

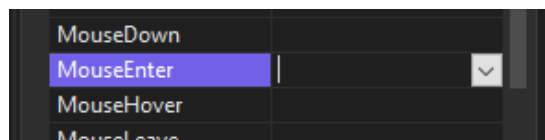


Рисунок 51 - Событие MouseEnter

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

Рисунок 52 - Код события MouseEnter

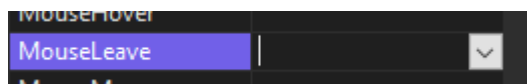
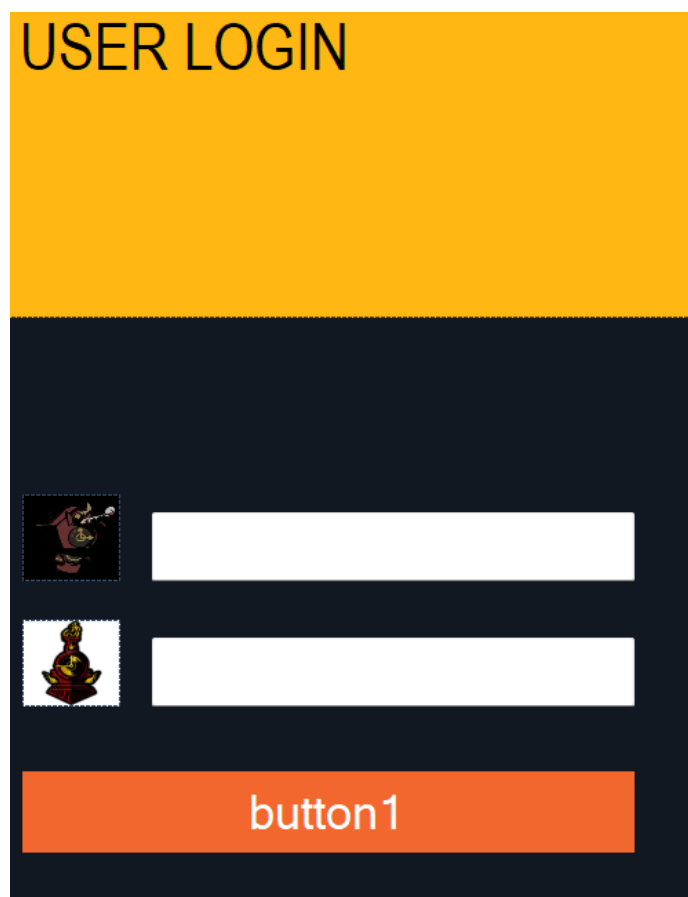


Рисунок 53 - Событие MouseLeave


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


Рисунок 54 – Код события MouseLeave

33. Готовая форма



USER LOGIN

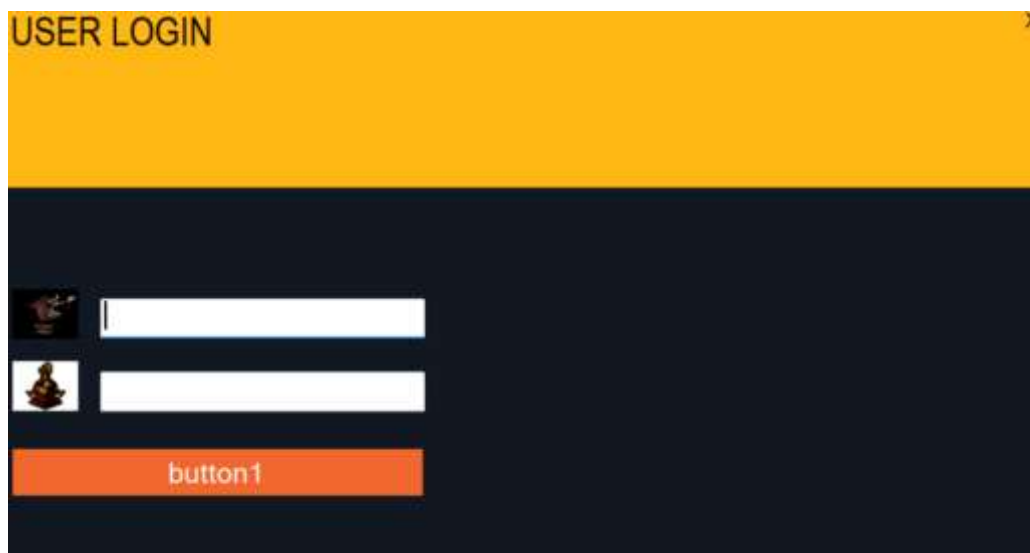





button1


Рисунок 55 -Готовая форма

34. Форма при наведении курсора на кнопку закрыть



USER LOGIN





button1

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

```
static 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

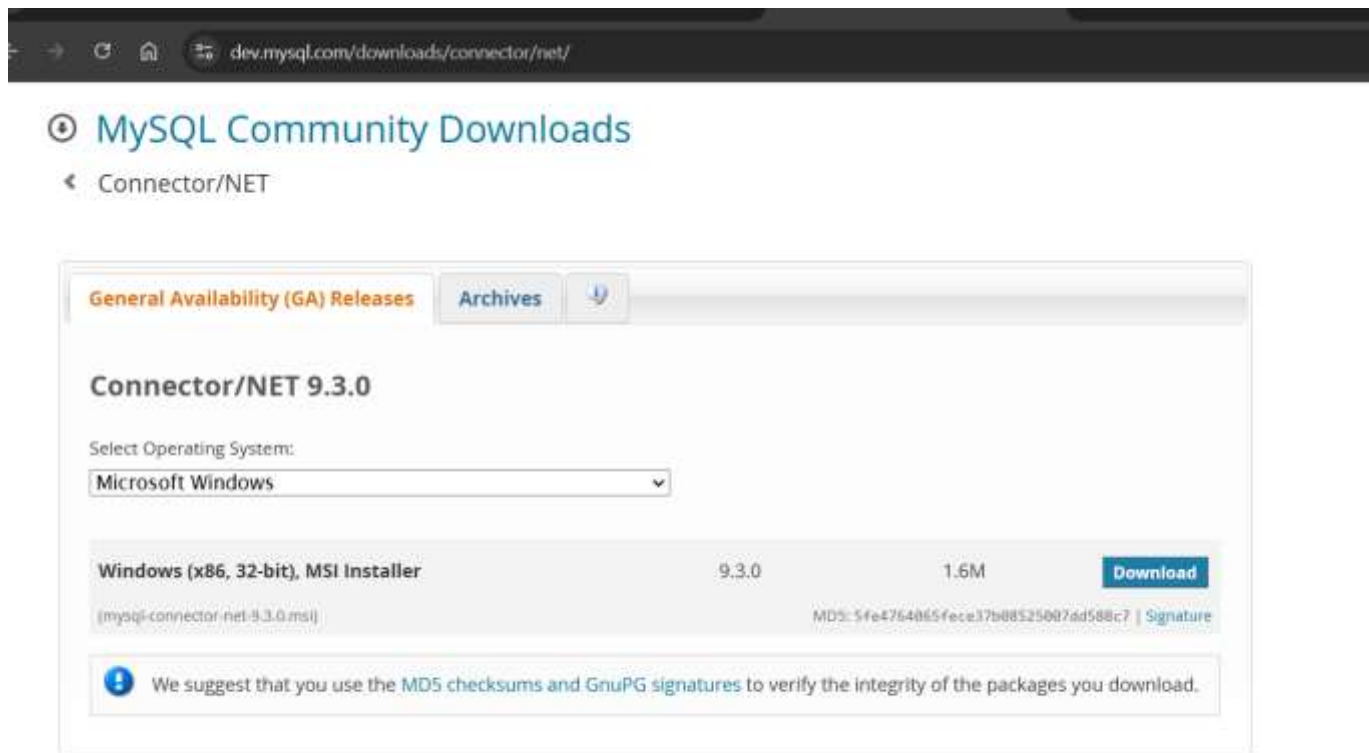


Рисунок 58 - Сайт mysql connector

3. Далее необходимо выполнить установку



Рисунок 59 - Установка

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

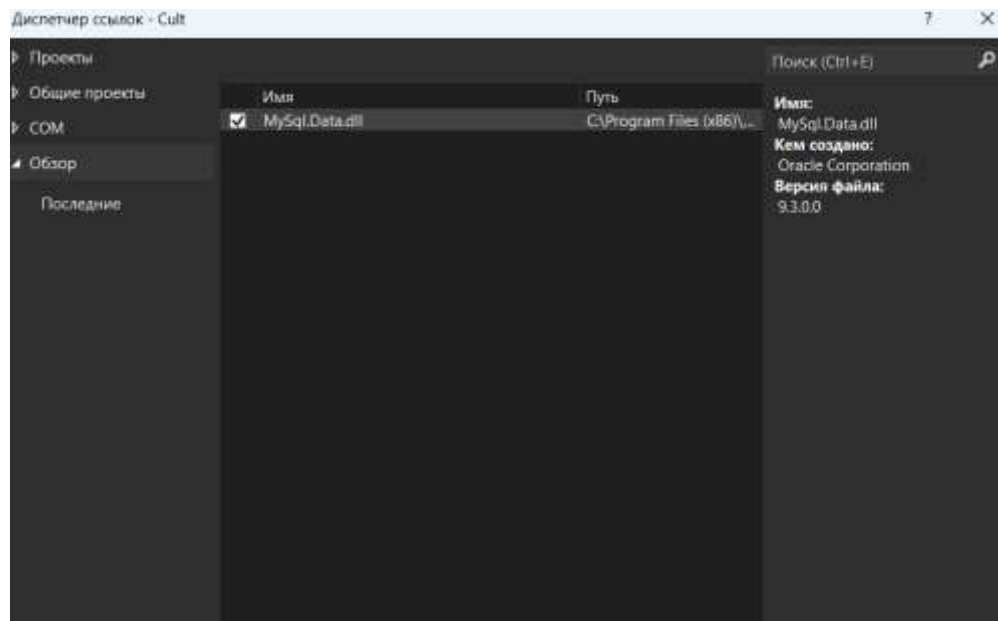


Рисунок 59 - Менеджер ссылок

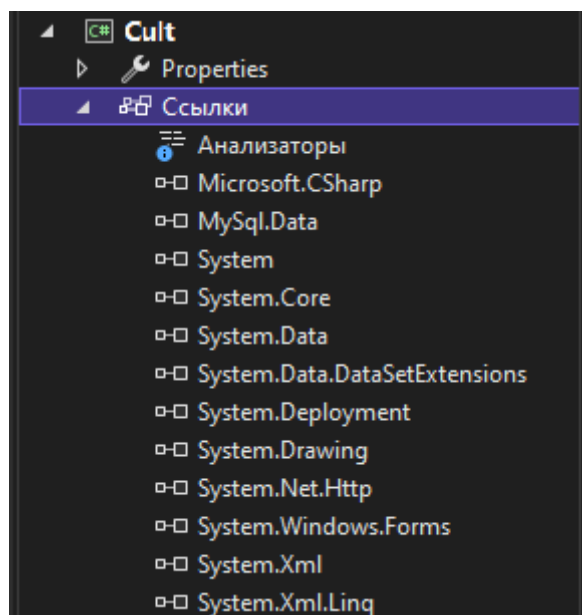


Рисунок 60 - Добавленные ссылки

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

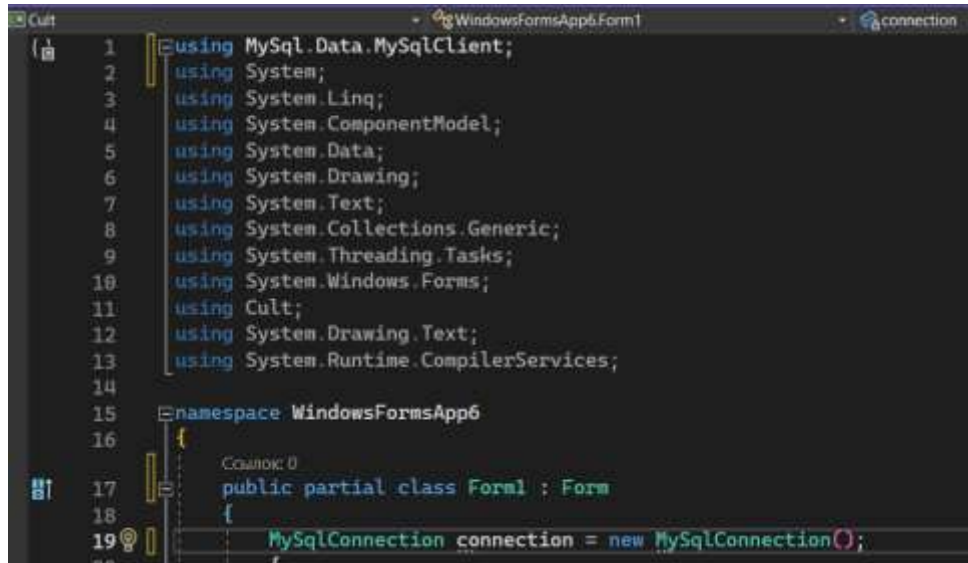


Рисунок 61 - Пространство имен

6. Запускаем сервер и заходим в PhpMyAdmin

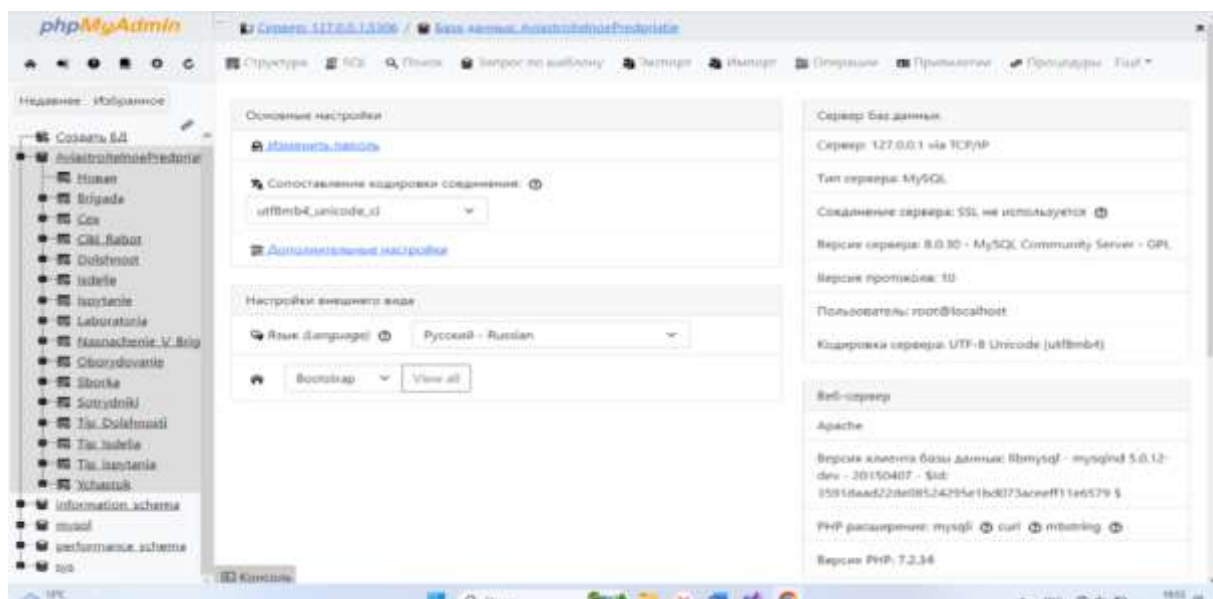


Рисунок 62 - PhpMyAdmin

7. Создаем новую бд под названием csharp_users_db

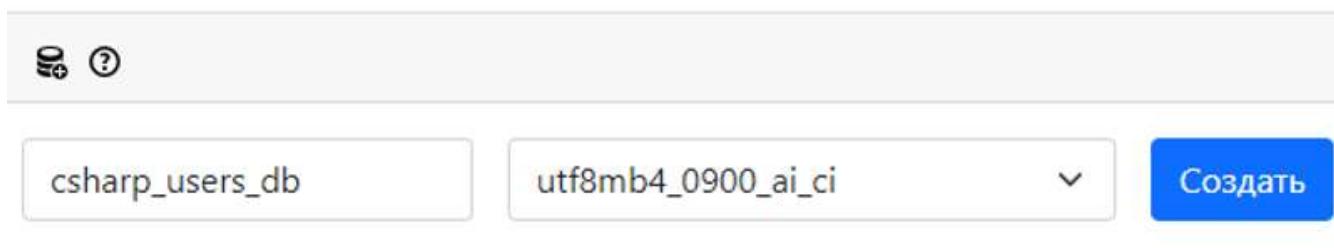
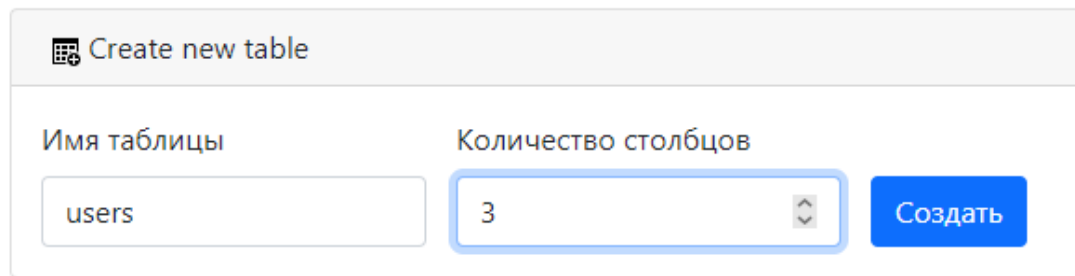


Рисунок 63 - Создание бд

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



Create new table

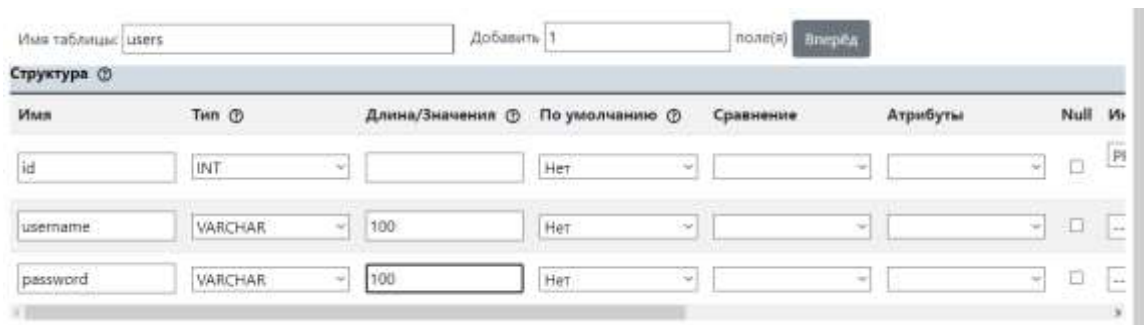
Имя таблицы: users

Количество столбцов: 3

Создать

Рисунок 64 - Создание таблицы

9. Заполняем в таблице атрибуты



Имя таблицы: users

Добавить 1 поле(я) Вперед

Имя	Тип	Длина/Значения	По умолчанию	Сравнение	Атрибуты	Null	Ис...
id	INT		Нет			<input type="checkbox"/>	Р...
username	VARCHAR	100	Нет			<input type="checkbox"/>	...
password	VARCHAR	100	Нет			<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;port=3306;username=root;password=;database=csharp_users_db");
```

Рисунок 66 - Подключение бд

11. Создаем новый класс

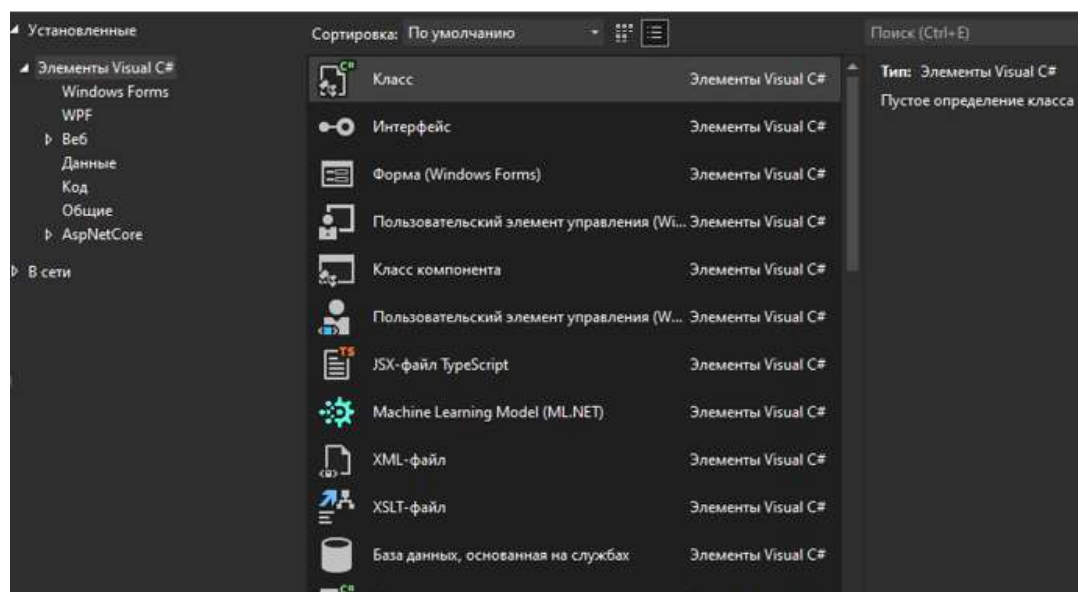


Рисунок 67 - Создание нового класса

12. Копируем наше подключение бд в новый созданный класс

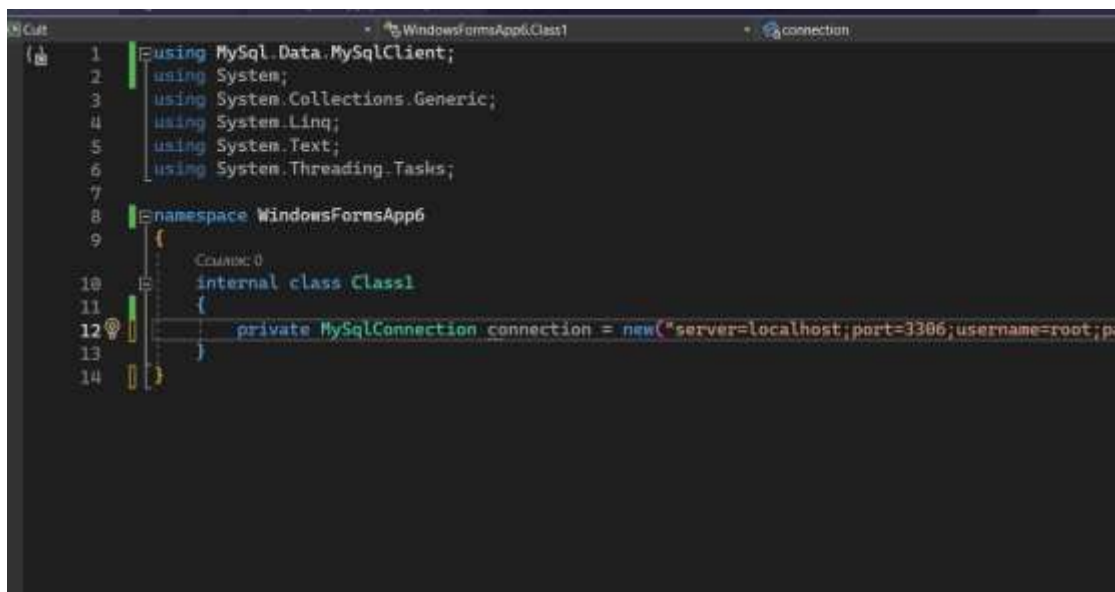


Рисунок 68 - Созданный класс

13. Пишем функцию для открытия соединений

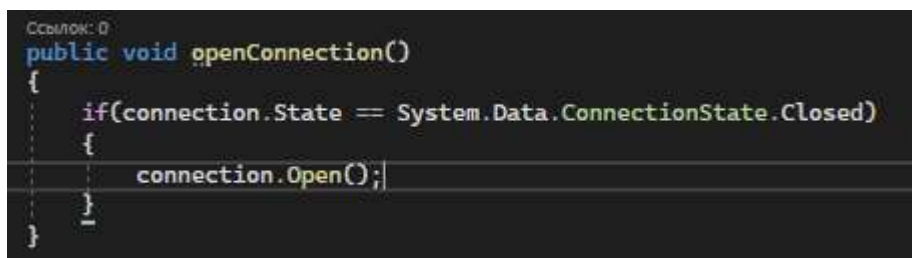


Рисунок 69 - Открытие соединения

14. Создаем функцию для закрытия соединения

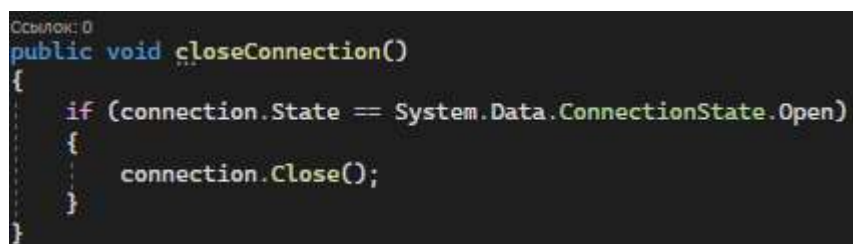


Рисунок 70 - Закрытие соединения

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

```
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

17. Вносим в нашу бд данные пользователя для проверки авторизации

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

[Вперед](#)

Рисунок 73 - Внесения данных в бд

Тестируем результат

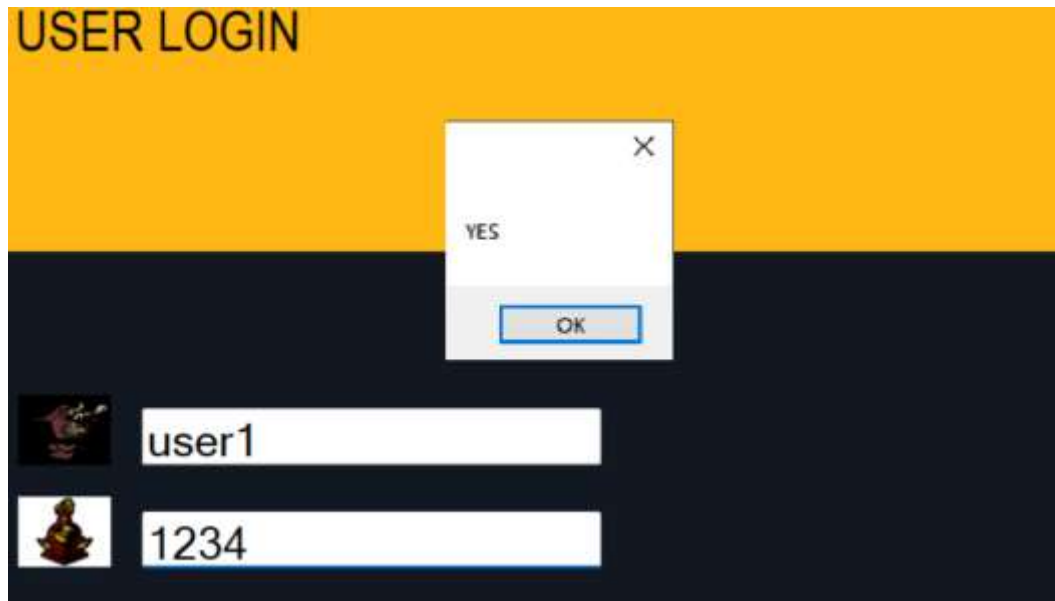


Рисунок 74 - Результат при введении верных логин и пароль

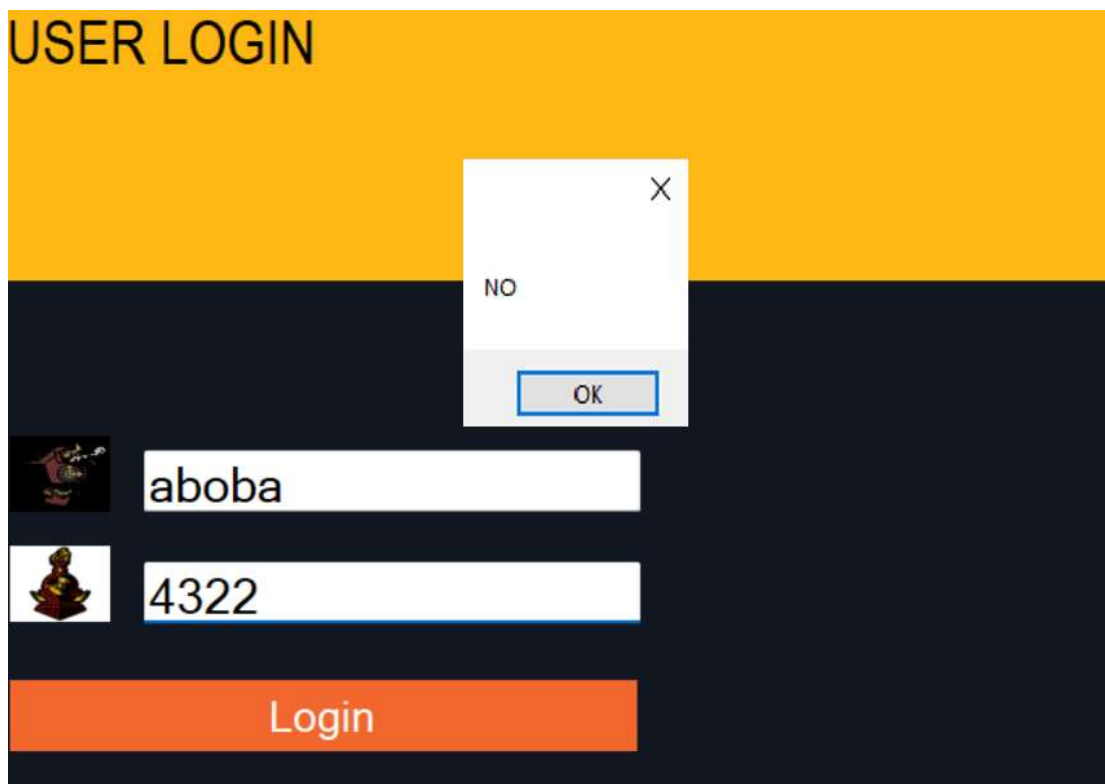


Рисунок 75 - Результат при введении неверных логин и пароль

Добавляем новую форму в проект

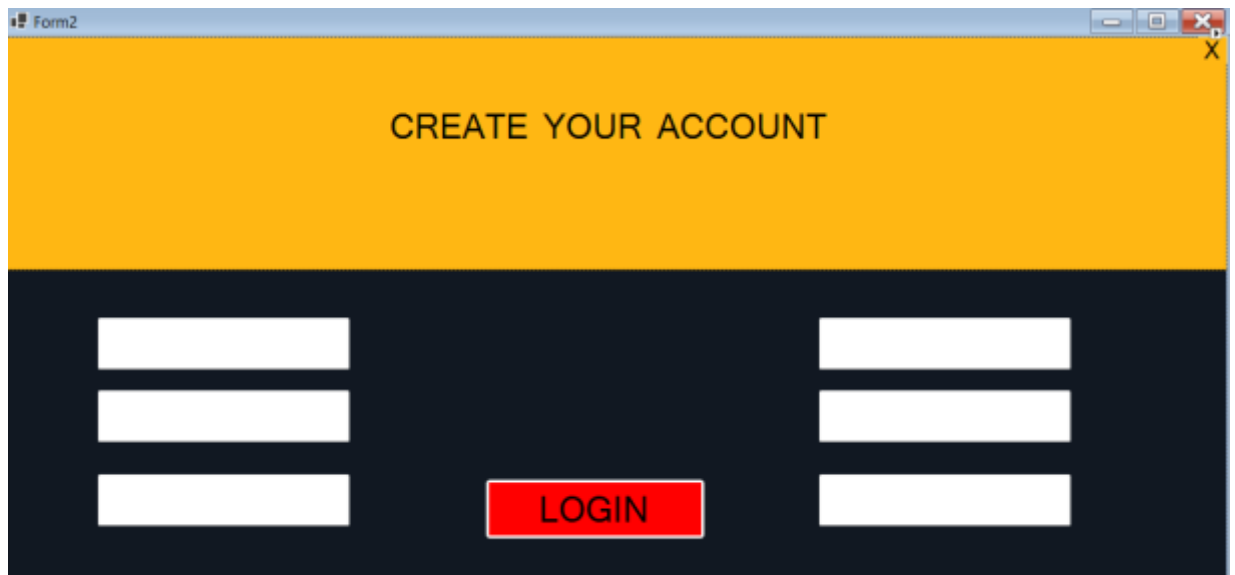


Рисунок 59 - Отредактированная форма RegisterForm

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

```
[STAThread]
Ссылка: 0
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new Form1());
}
```

Рисунок 60 - Program.cs

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

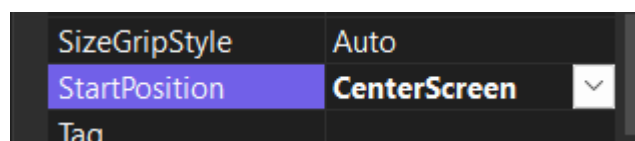


Рисунок 61 – StartPosition

Переименовываем все textbox

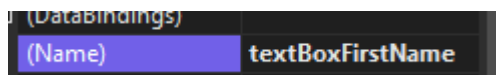


Рисунок 62 - textBoxFirstName

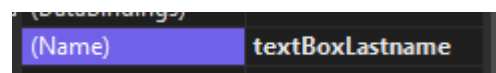


Рисунок 63 - textBoxLastName

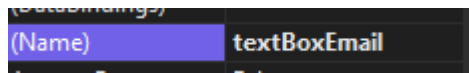


Рисунок 64 - textBoxEmail

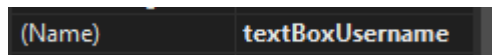


Рисунок 65 - textBoxUsername

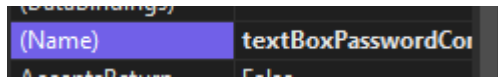


Рисунок 66 - textBoxPasswordConfirm

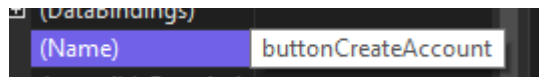
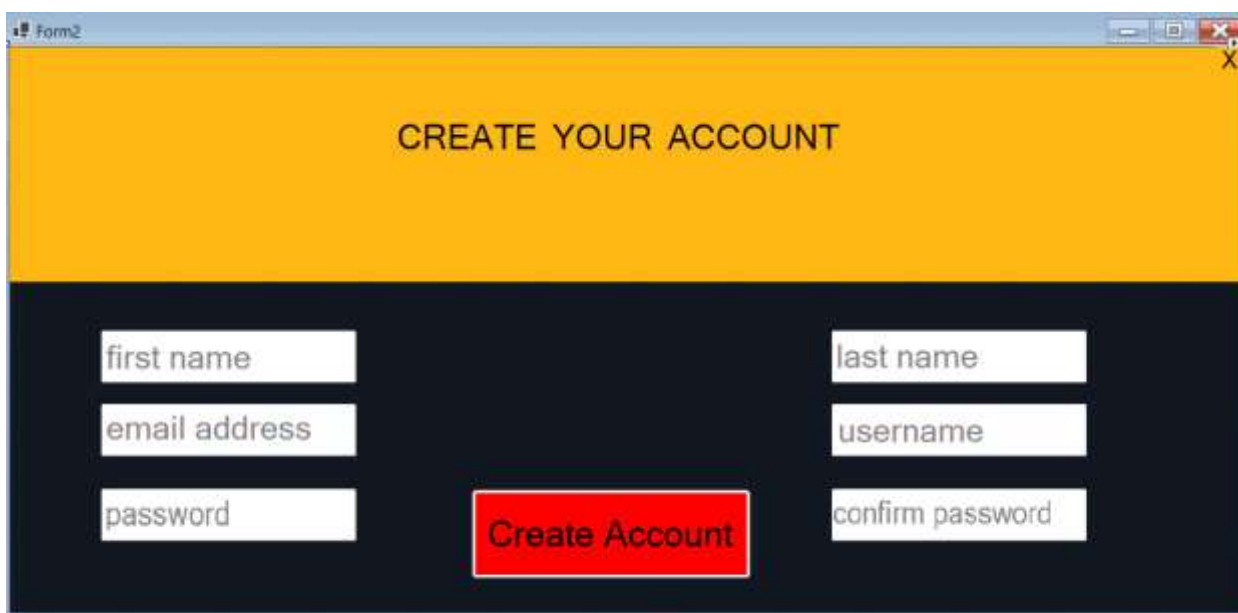


Рисунок 67 – buttonCreateAccountx

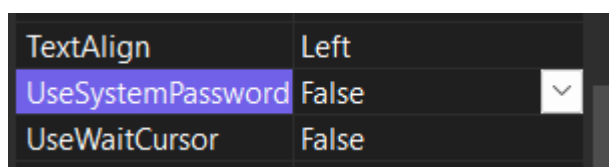
Промежуточный результат



A screenshot of a Windows application window titled 'Form2'. The window has a yellow header bar with the text 'CREATE YOUR ACCOUNT' in black. Below the header is a dark blue area containing several white text input fields. On the left side, there are three fields labeled 'first name', 'email address', and 'password'. On the right side, there are three fields labeled 'last name', 'username', and 'confirm password'. In the center of the dark blue area is a red button with the text 'Create Account' in white. The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

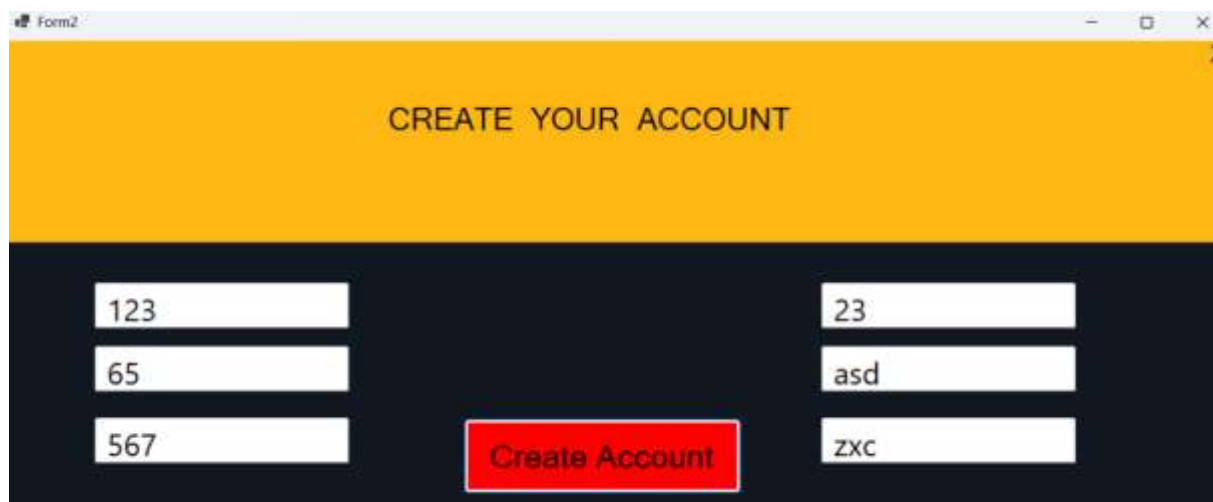
Рисунок 68 - Промежуточный результат

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



A screenshot of the Visual Studio Properties window. The 'TextAlign' property is set to 'Left'. The 'UseSystemPassword' property is highlighted in blue and set to 'False'. The 'UseWaitCursor' property is set to 'False'.

Рисунок 69 - UseSystemPassword



A screenshot of the same 'Form2' application window. The input fields now contain sample data: 'first name' is '123', 'email address' is '65', 'password' is '567', 'last name' is '23', 'username' is 'asd', and 'confirm password' is 'zxc'. The 'Create Account' button remains in the center. The window controls are still visible in the top right corner.

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 текстовки 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

```
Ссылка: 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;
    }
}

Ссылка: 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

```
Ссылка: 1
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

```
Ссылка: 1
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

```

Ссылка: 1
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

```

Ссылка: 1
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

```

Ссылка: 1
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

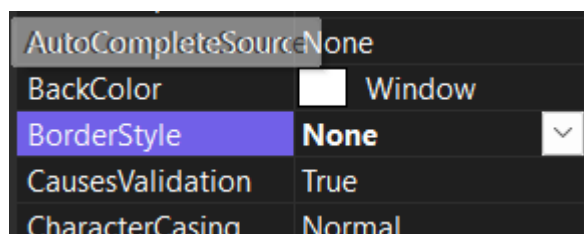
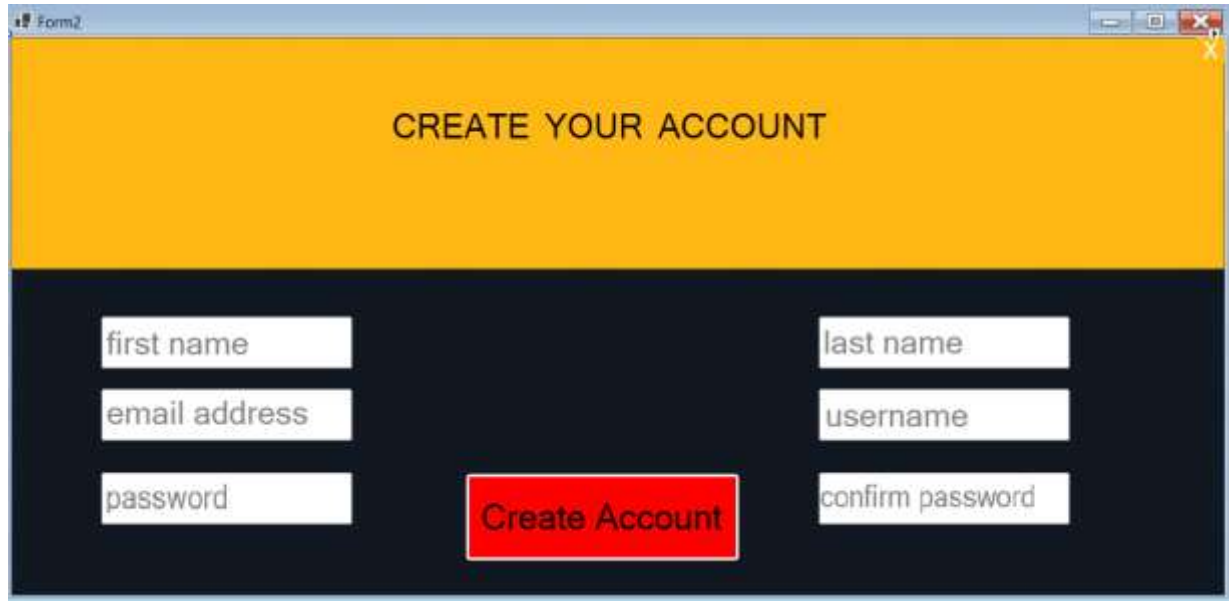


Рисунок 80 - FormBorderStyle

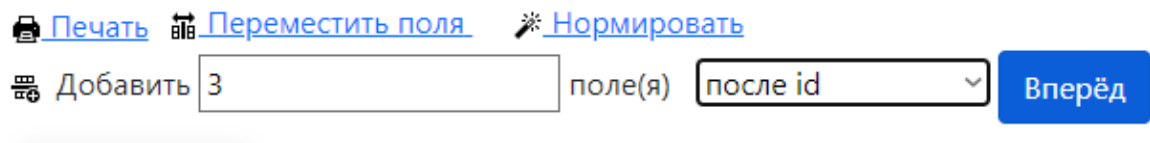
После выполнения всех действий получаем данный результат



The screenshot shows a web browser window titled 'Form2'. The main heading is 'CREATE YOUR ACCOUNT' in white text on a yellow background. Below this, on a dark blue background, are six input fields arranged in two columns. The left column contains 'first name', 'email address', and 'password'. The right column contains 'last name', 'username', and 'confirm password'. A red button with the text 'Create Account' is centered between the two columns.

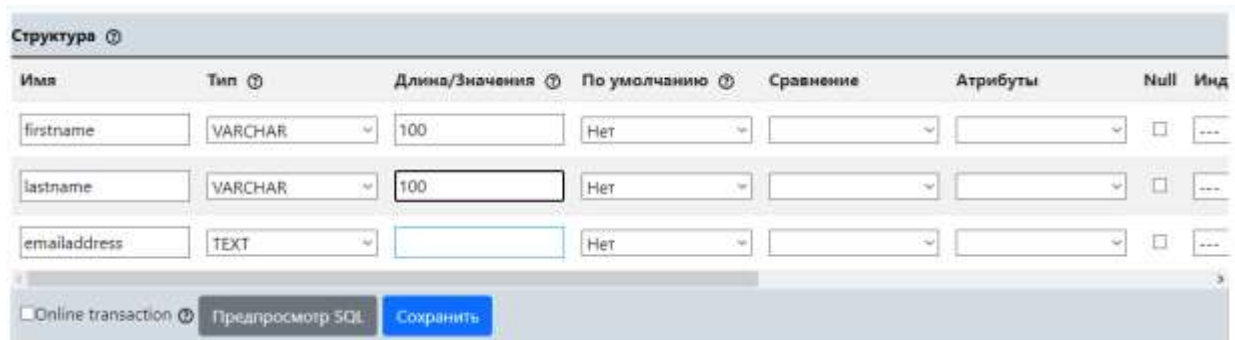
Рисунок 81 - Результат

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



The screenshot shows the top part of the PhpMyAdmin interface. There are three links: 'Печать' (Print), 'Переместить поля' (Move fields), and 'Нормировать' (Normalize). Below these is a section for adding columns: 'Добавить' (Add) followed by a text input containing '3', then 'поле(я)' (field(s)), a dropdown menu with 'после id' (after id) selected, and a blue button labeled 'Вперёд' (Forward).

Рисунок 82 - Добавление столбцов в таблицу



The screenshot shows the 'Структура' (Structure) tab in PhpMyAdmin. It displays a table with three columns: 'firstname' (VARCHAR, 100), 'lastname' (VARCHAR, 100), and 'emailaddress' (TEXT). The table is shown in a grid format with columns for Name, Type, Length/Values, Default, Comparison, Attributes, Null, and Index. At the bottom, there are buttons for 'Online transaction', 'Предпросмотр SQL' (Preview SQL), and 'Сохранить' (Save).

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

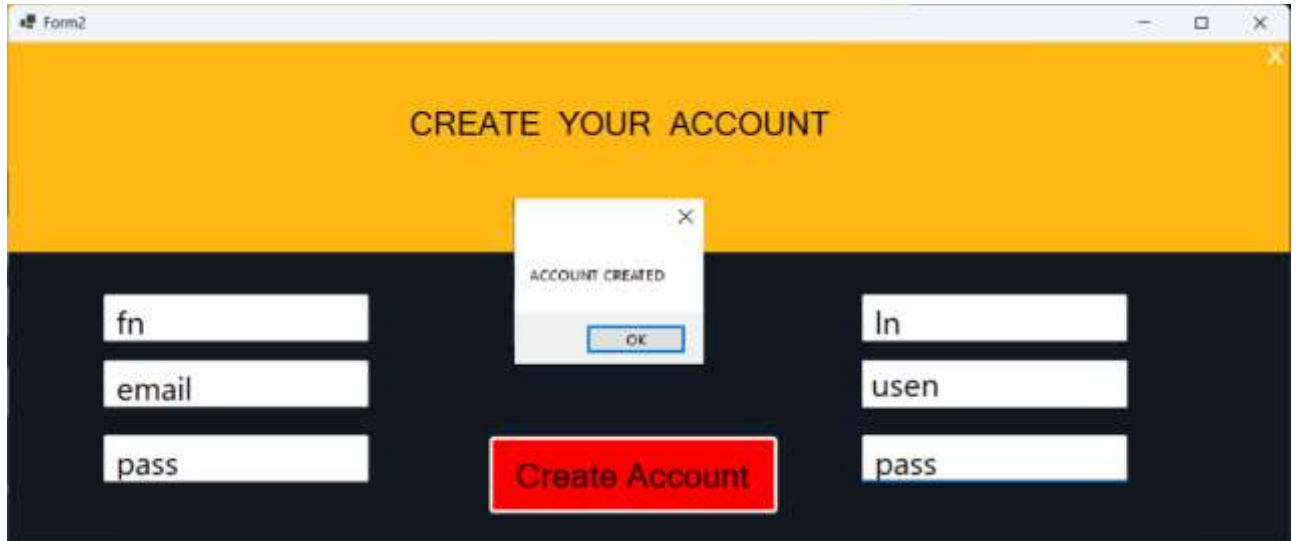


Рисунок 86 - Промежуточный результат

	<u>id</u>	<u>firstname</u>	<u>lastname</u>	<u>emailaddress</u>	<u>username</u>	<u>password</u>
<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

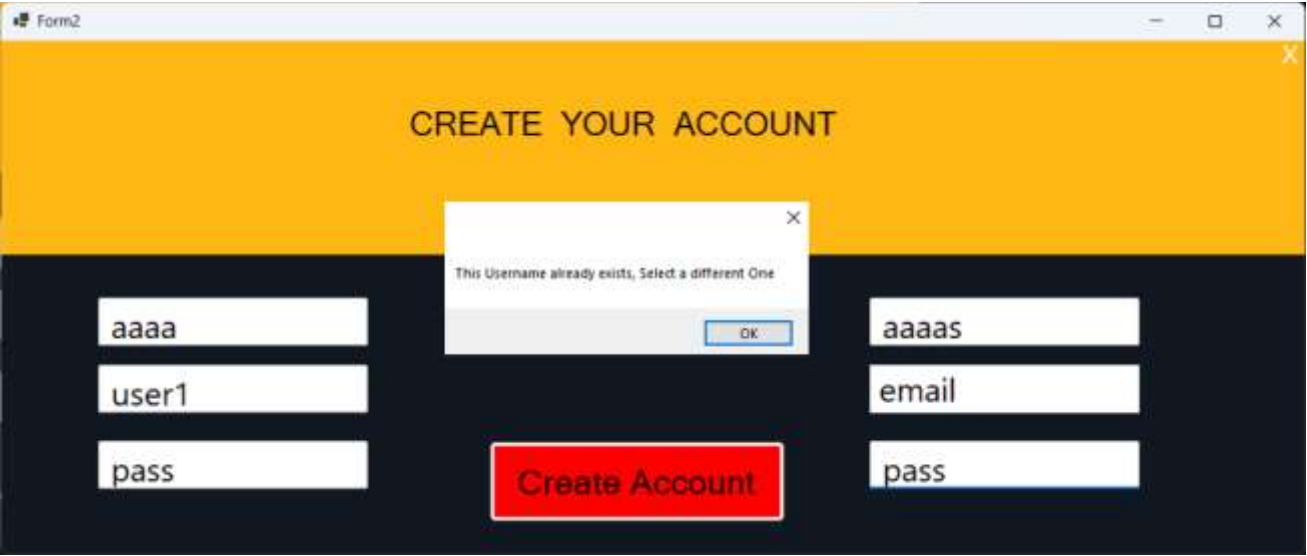


Рисунок 91 - Создание уже существующего пользователя

Проверка создания аккаунта с данными еще существующего пользователя

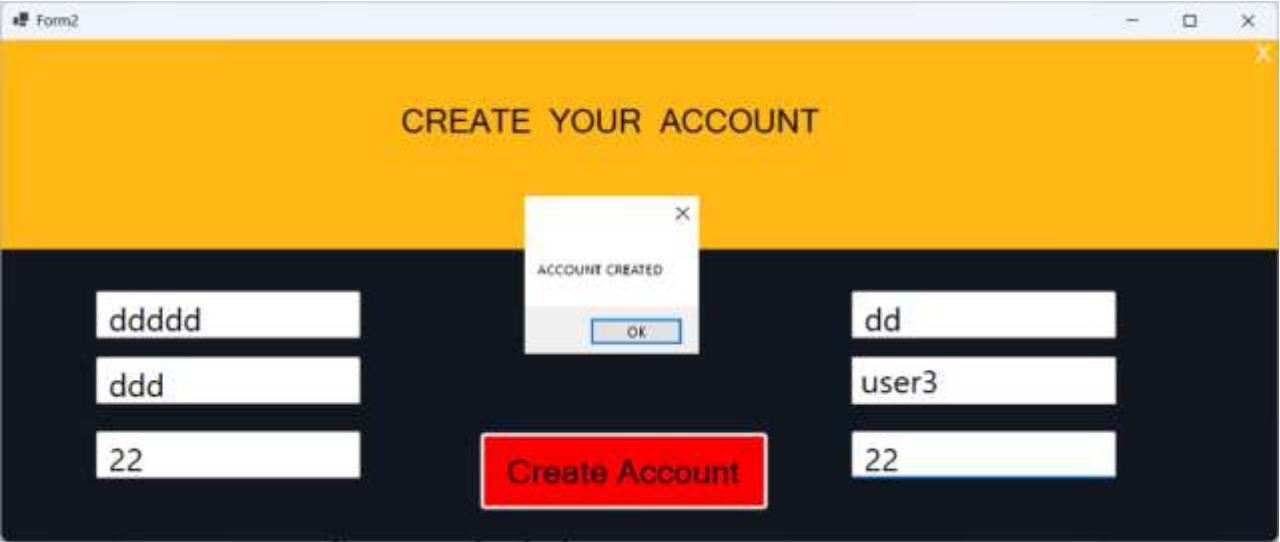


Рисунок 92 - Создание аккаунта

Добавление нового пользователя в базе данных

	<u>id</u>	<u>firstname</u>	<u>lastname</u>	<u>emailaddress</u>	<u>username</u>	<u>password</u>
<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 сообщение «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 screenshot shows a web interface for creating an account. The background is dark blue with a yellow header bar containing the text "CREATE YOUR ACCOUNT". Below the header, there are several input fields: "first name", "email address", "password", "last name", "username", and "confirm password". A prominent red button labeled "Create Account" is centered below the "password" field. A modal dialog box is open in the center of the screen, displaying the message "ENTER YOUR INFORMATION FIRST" and an "OK" button. This indicates that the system is preventing account creation due to empty input fields.

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

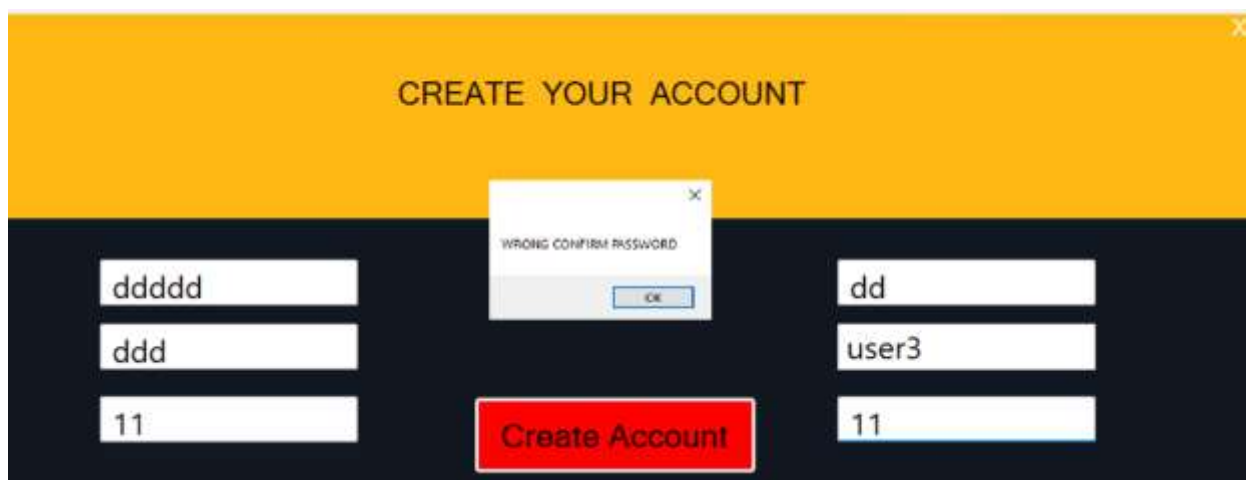


Рисунок 97 – Отрицательный результат проверки пароля

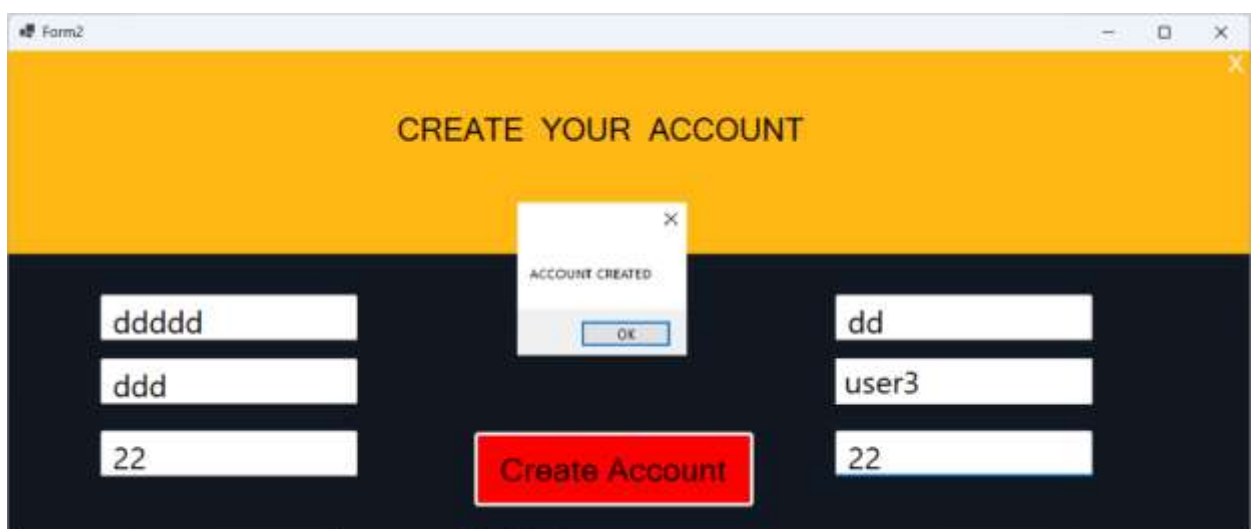


Рисунок 98 – Положительный результат проверки пароля

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 «OK».

```
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

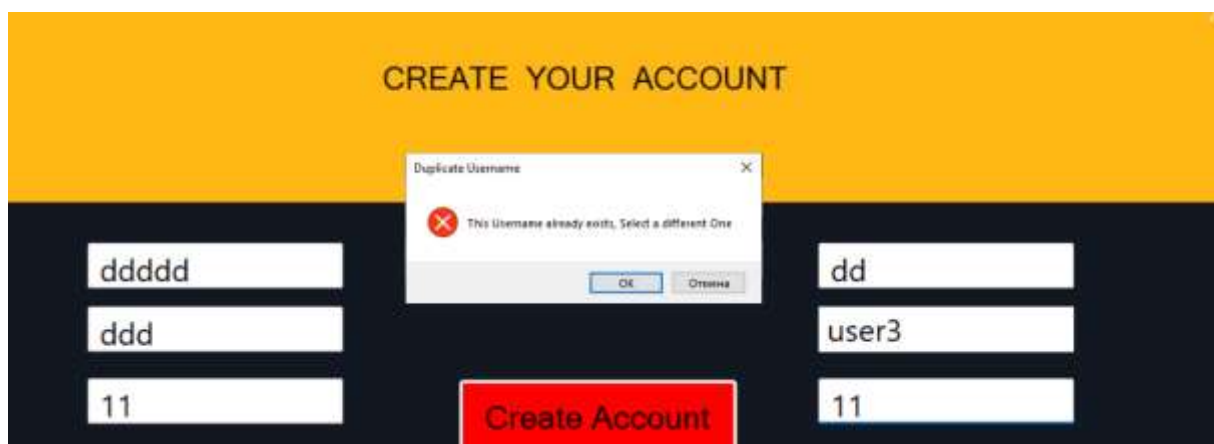


Рисунок 100 - Готовые MessageBox с ошибкой

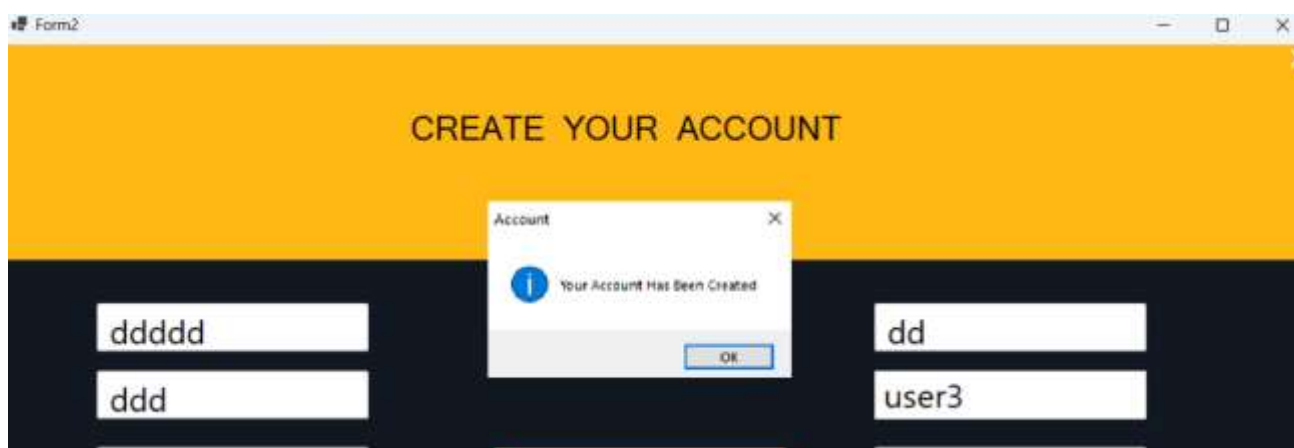


Рисунок 101 - Готовые MessageBox без ошибки

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


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

Рисунок 102 - Доработка MessageBox

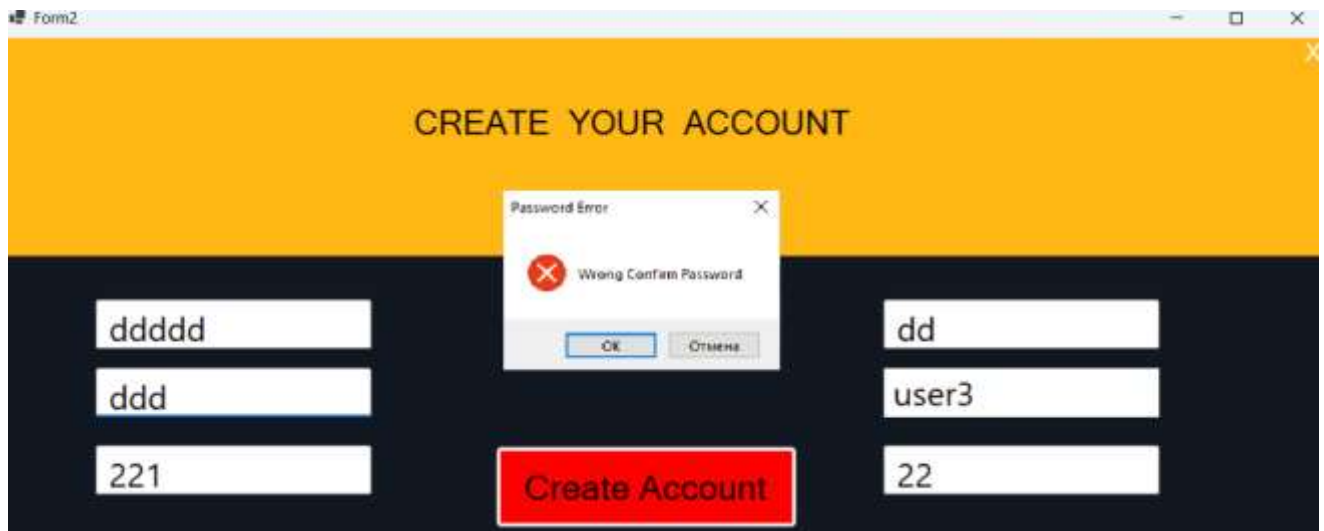


Рисунок 103 – Готовый MessageBox с ошибкой

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

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

Рисунок 104 - Доработка MessageBox

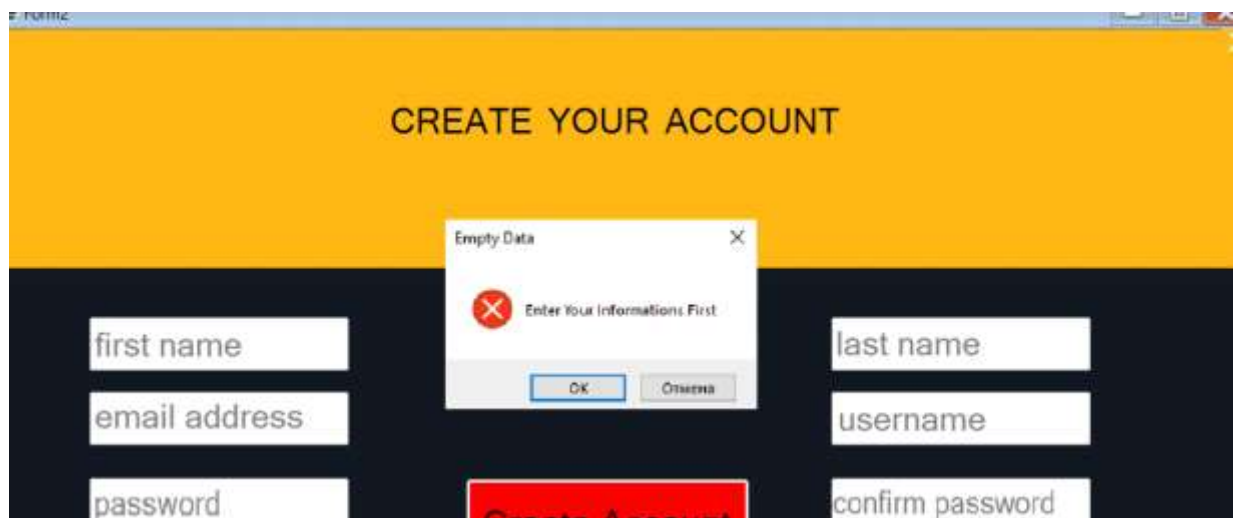


Рисунок 105 - Готовый MessageBox с ошибкой

Так же добавляем в месседжбоксы 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.

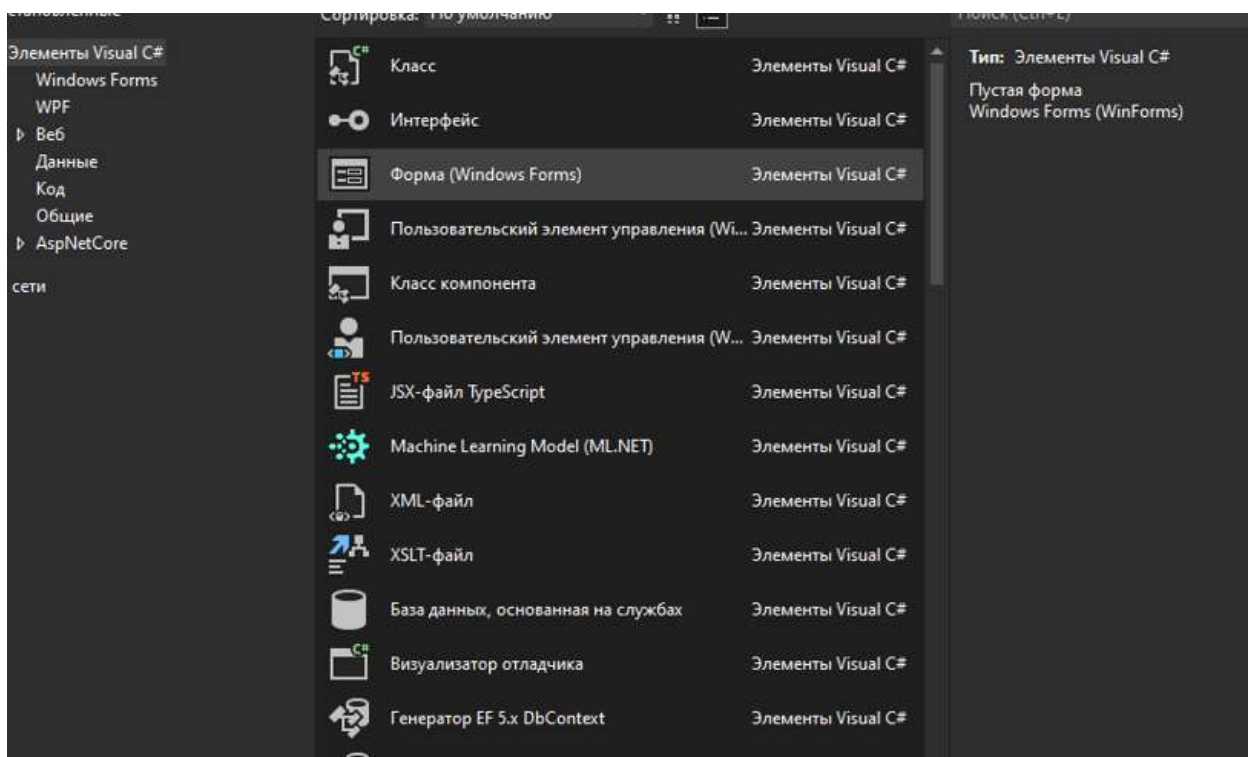


Рисунок 107 - Создание новой формы

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

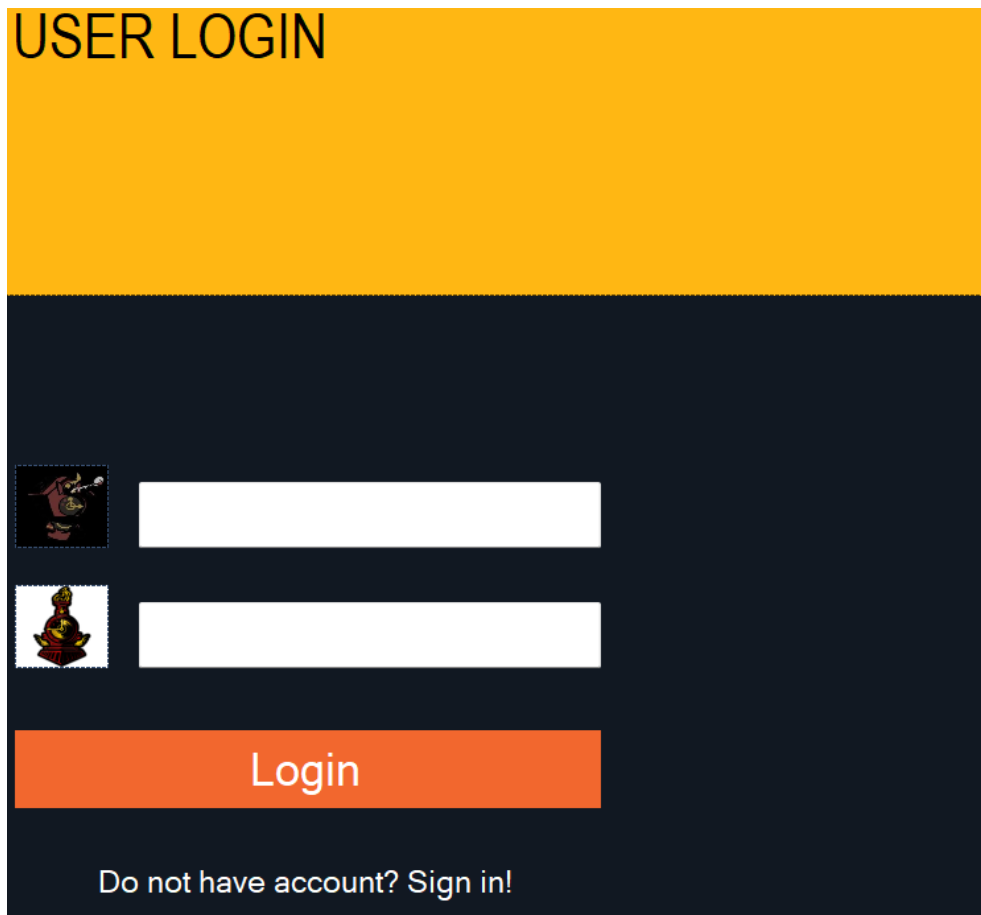


Рисунок 112 - Отредактированная LoginForm

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»

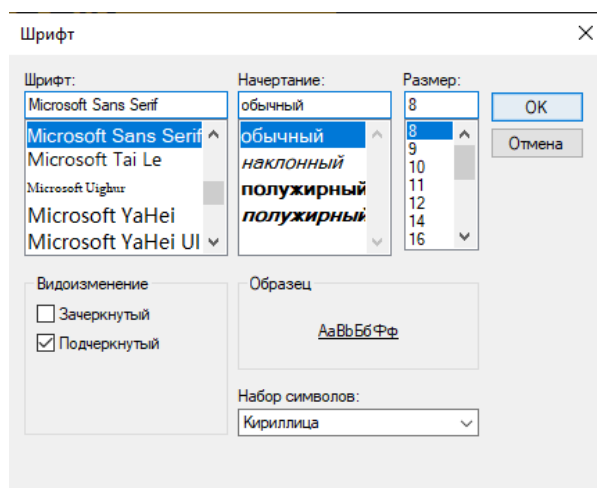


Рисунок 118 - Настройки шрифта

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

Рисунок 119 - RegisterForm

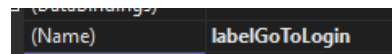


Рисунок 120 - Название label

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.

```

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 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

Prilozhenie 1.2.

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.

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

