

1. Prilozhenie 1.1. Sozдание formy avtorizatsii
2. Создание проекта

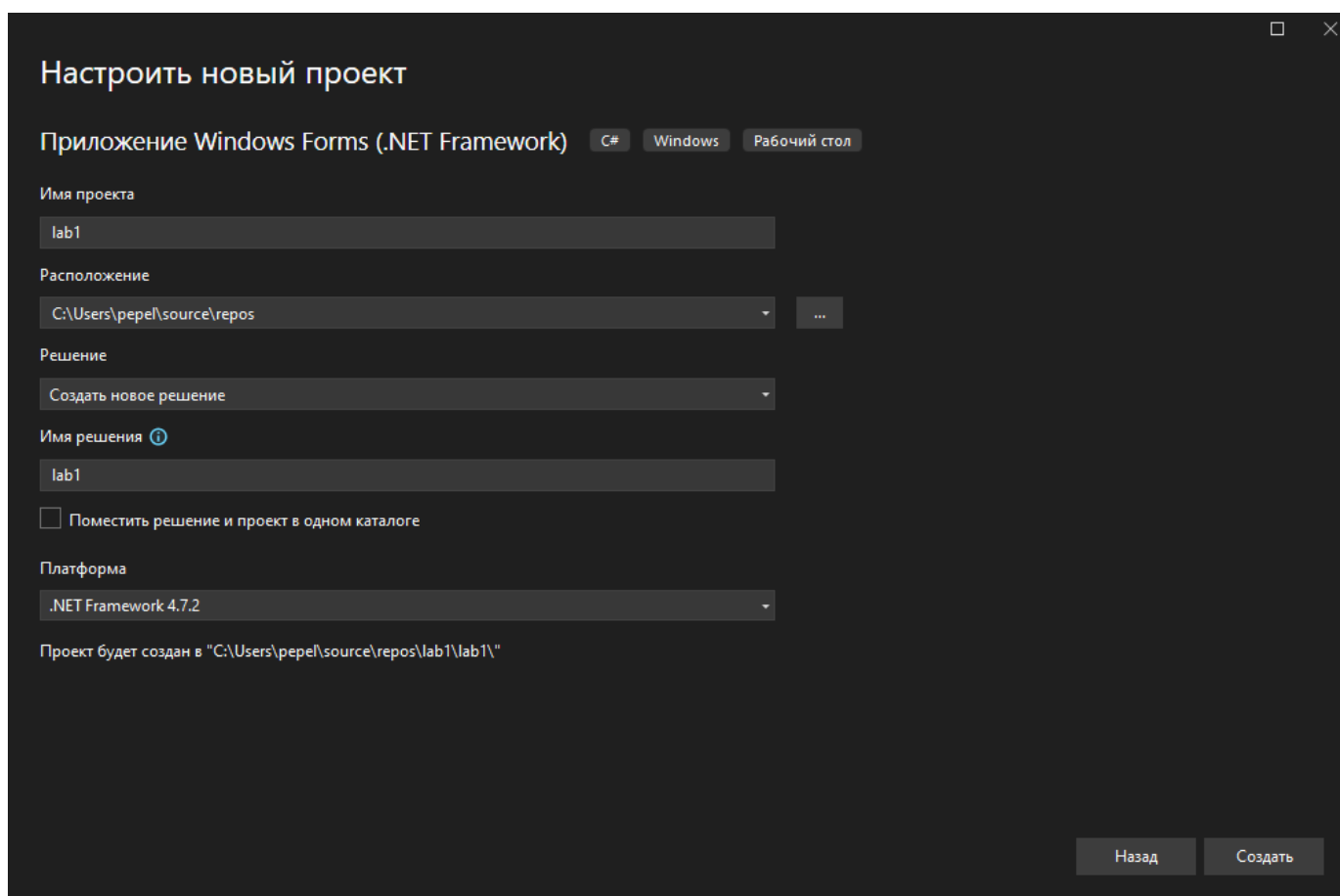


Рисунок 1 - Создание проекта

3. Созданный проект

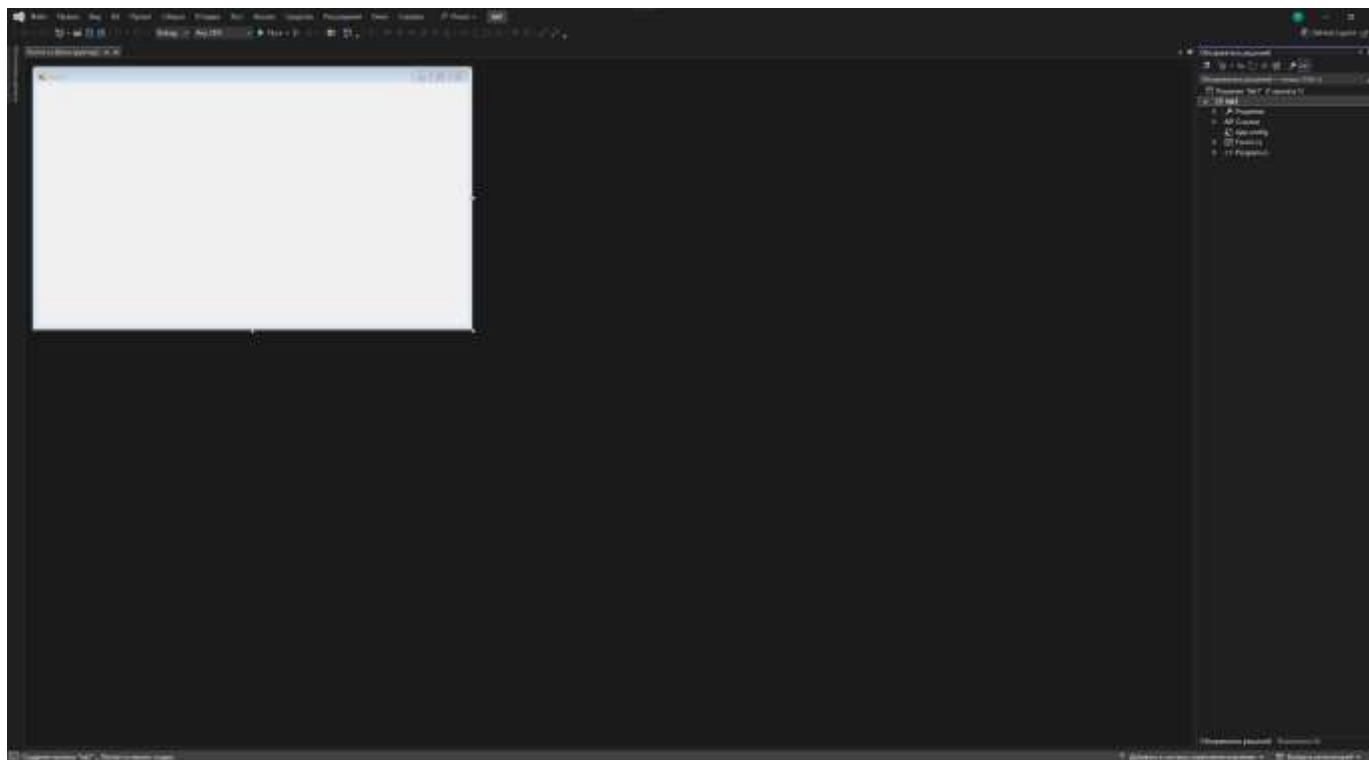


Рисунок 2 - Созданный проект

4. Удаление Form1

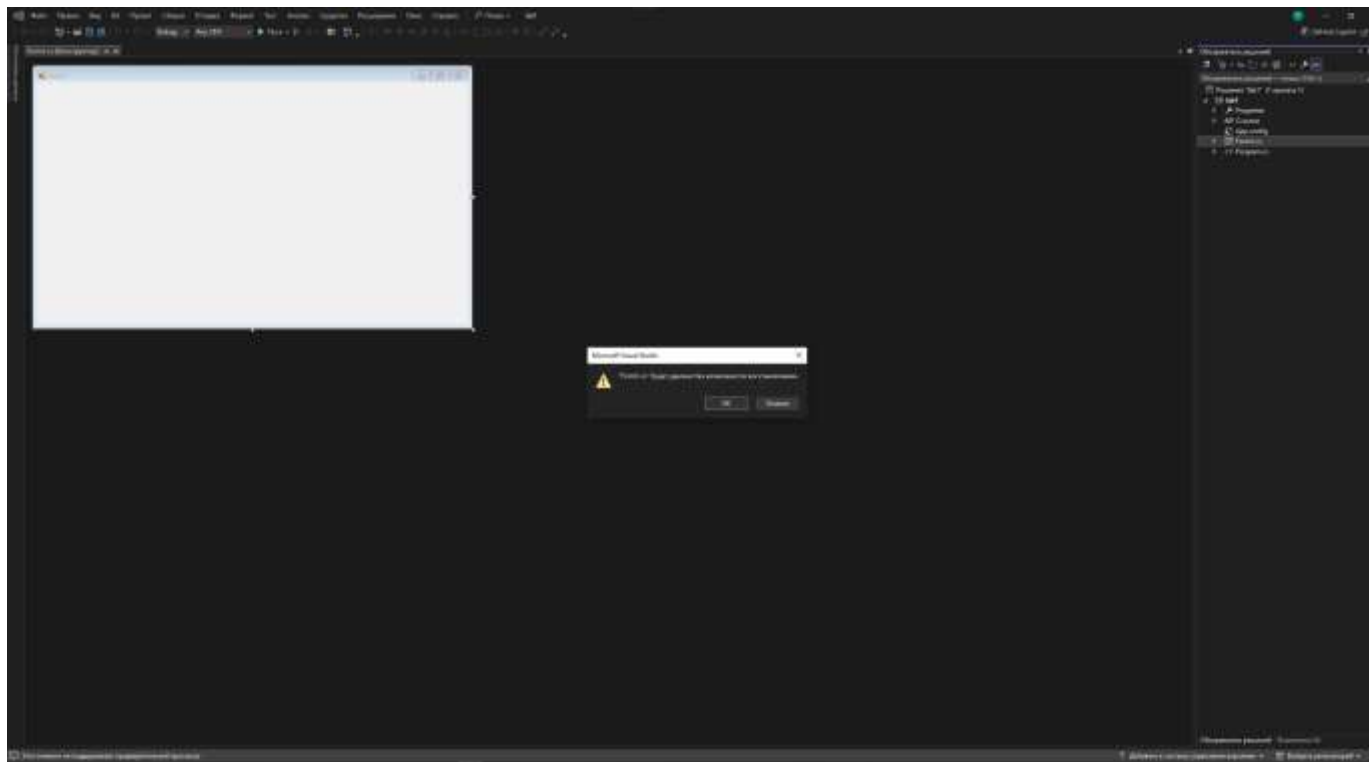


Рисунок 3 - Удаление формы

5. Создание LoginForm

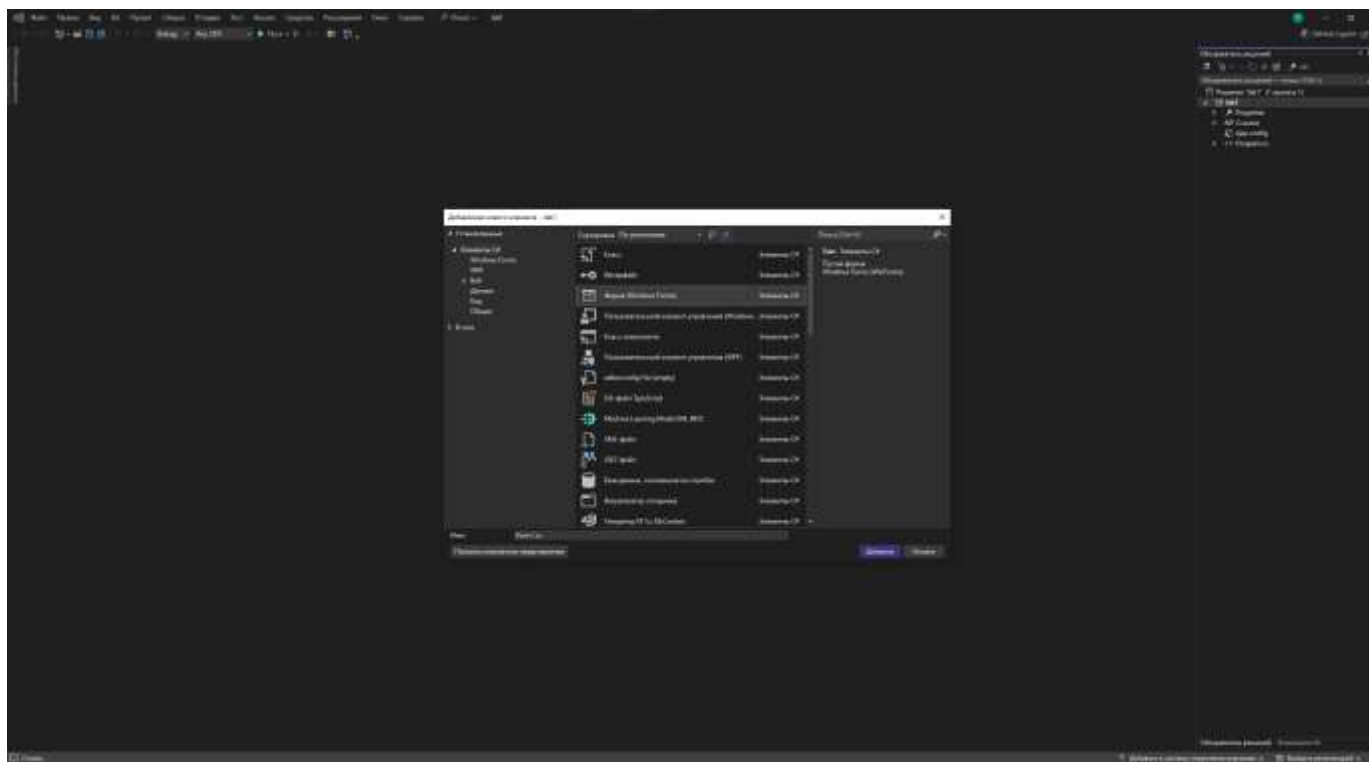


Рисунок 4 - Создание формы

6. Открываем панель элементов

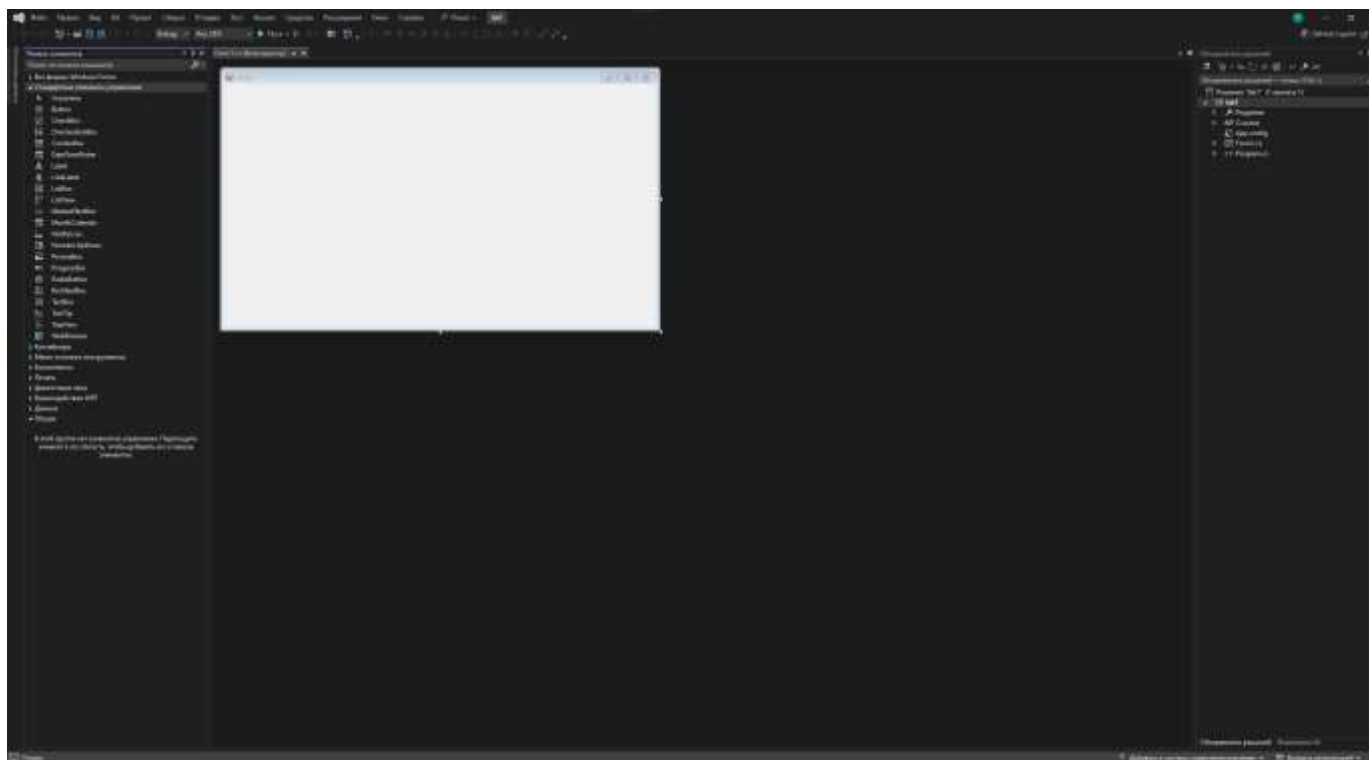


Рисунок 5 - Панель элементов

7. Выбираем в панели элементов элемент panel

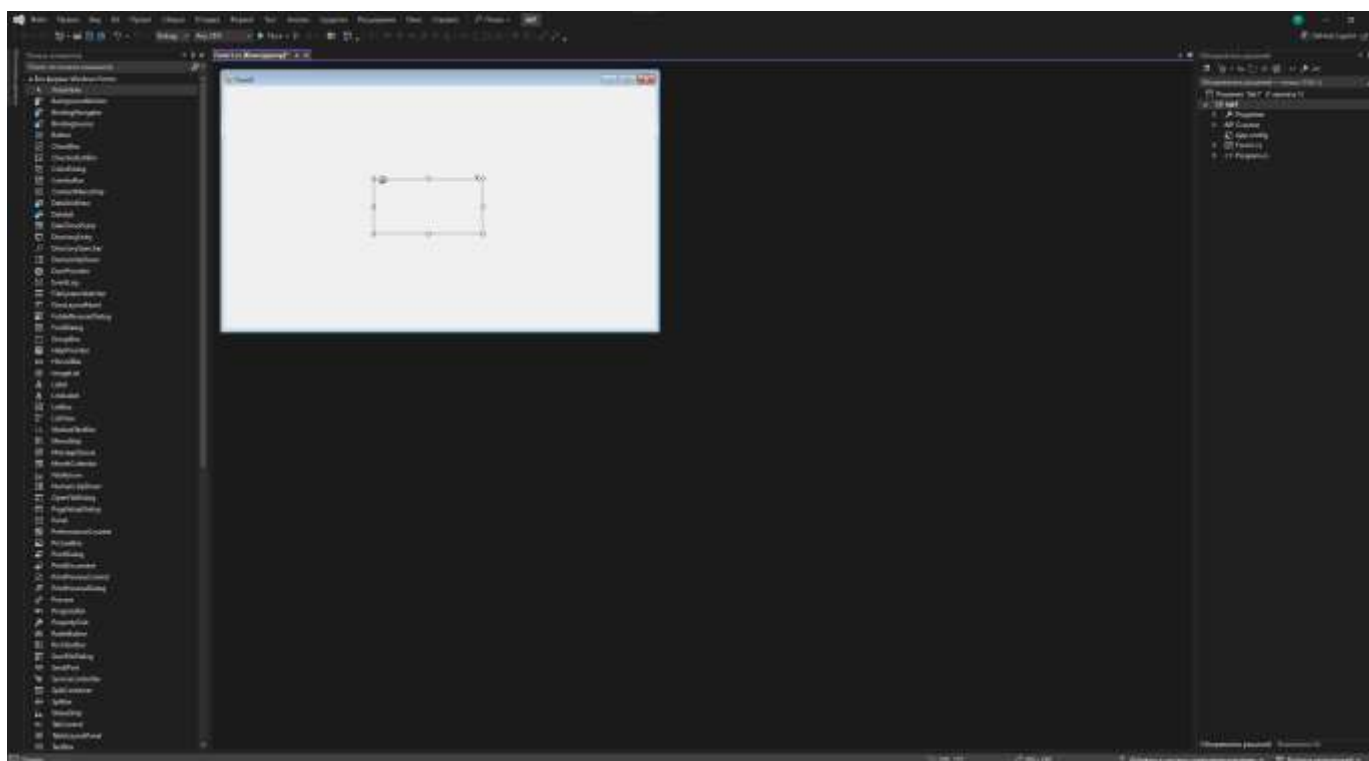


Рисунок 6 - Элемент panel

8. Закрепляем panel в родительском контейнере

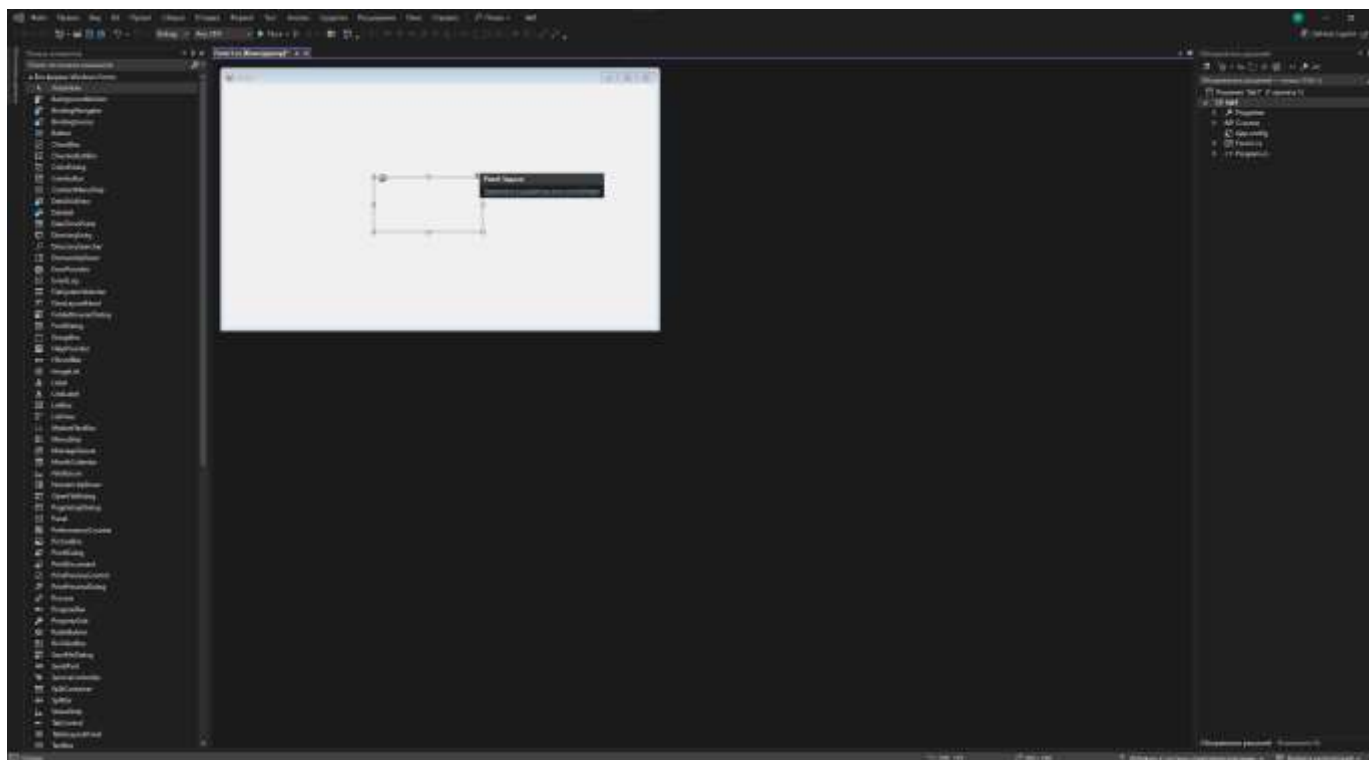


Рисунок 7 – Закрепление panel

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

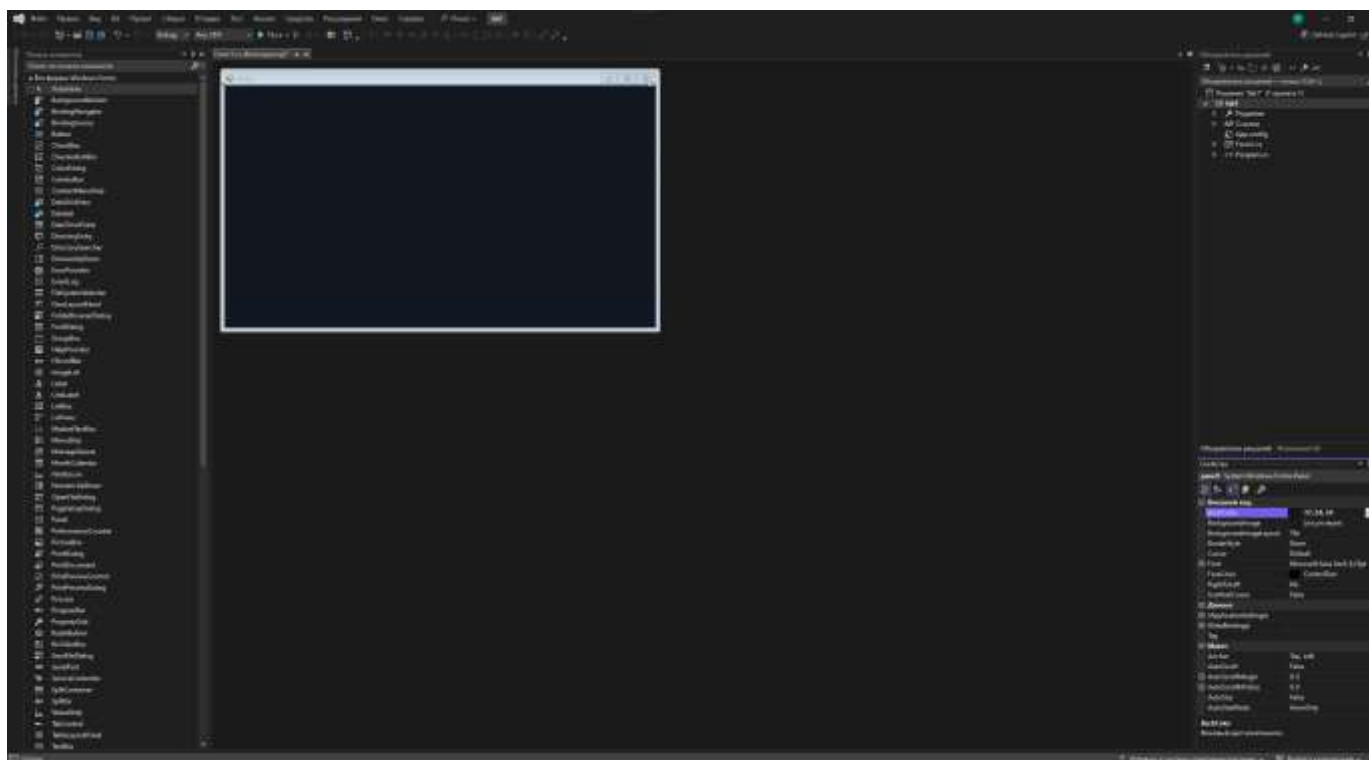


Рисунок 8 - Цвет панели

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

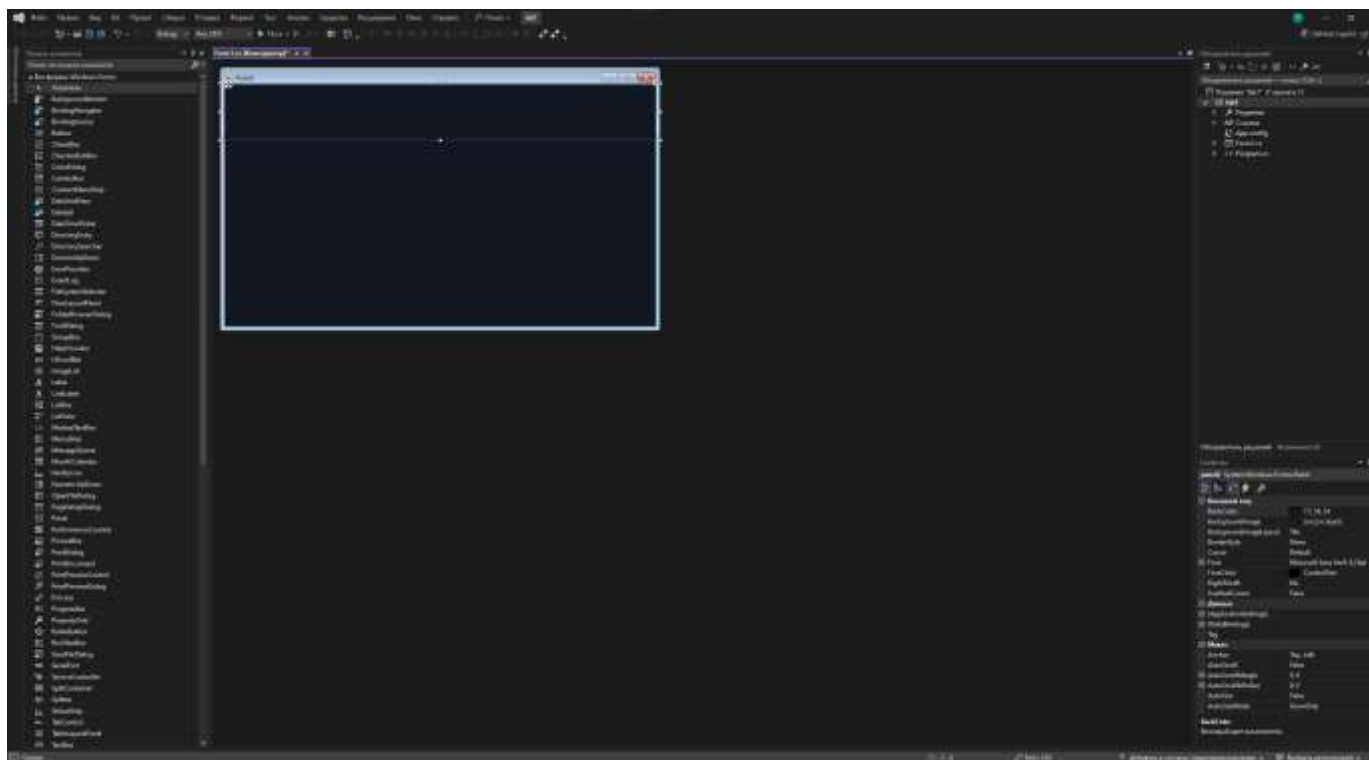


Рисунок 9 - Создание второй панели

11. Меняем значения свойства BackColor элемента Panel на D

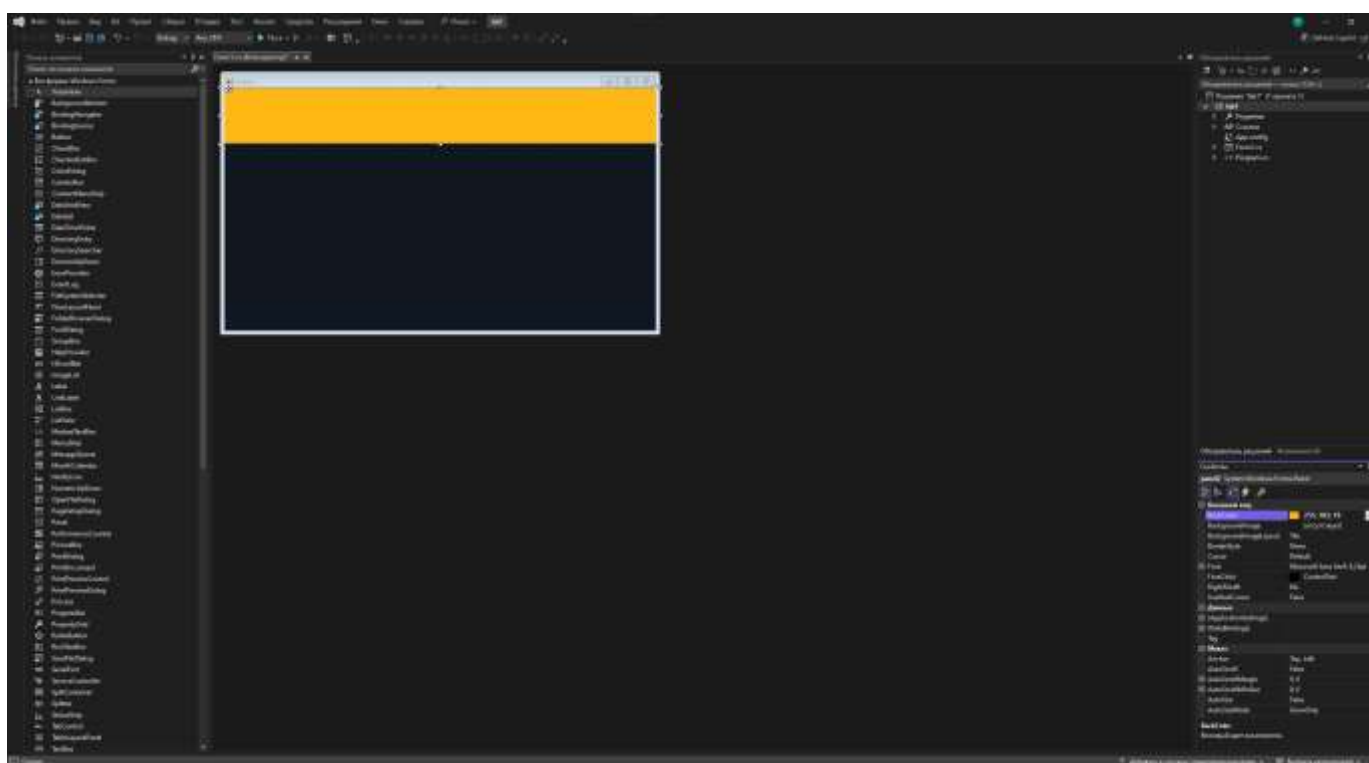


Рисунок 10 - Цвет панели

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

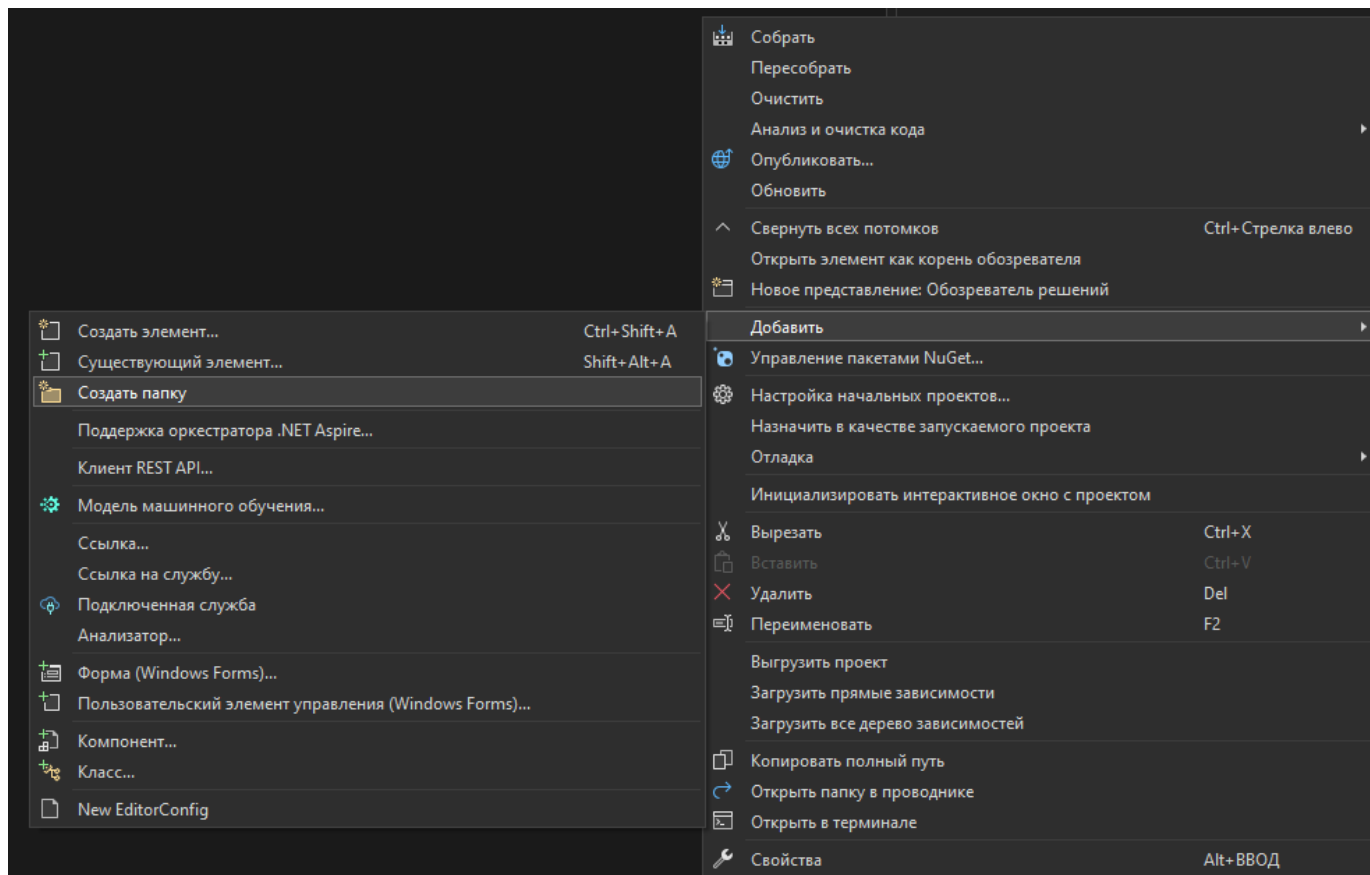


Рисунок 11 - Создание папки

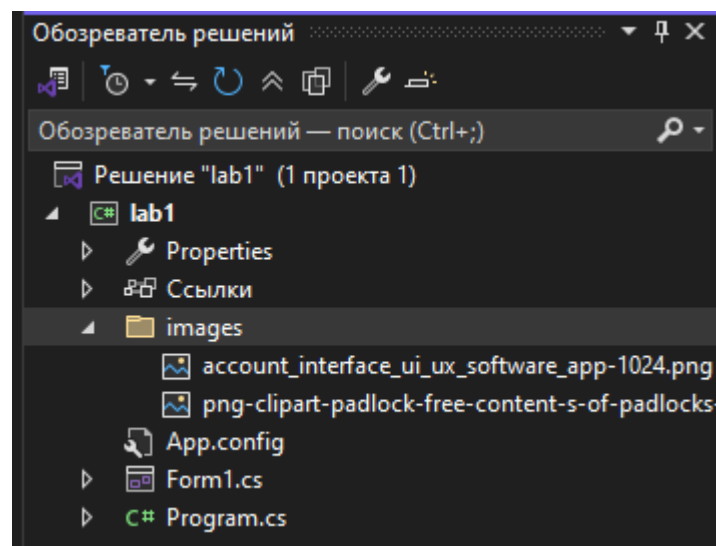


Рисунок 12 - Созданная папка

13. Создаем label и настраиваем шрифт и текст

Font	Arial; 28pt
Name	ab Arial
Size	28
Unit	Point
Bold	False
GdiCharSet	204
GdiVerticalFont	False
Italic	False
Strikeout	False
Underline	False

Рисунок 13 - Настройки шрифта

ImageList	(нет)
RightToLeft	No
Text	USER LOGIN
TextAlign	TopLeft
UseMnemonic	True

Рисунок 14 - Текст label

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

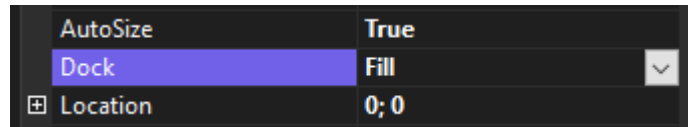


Рисунок 15 - Свойство Dock

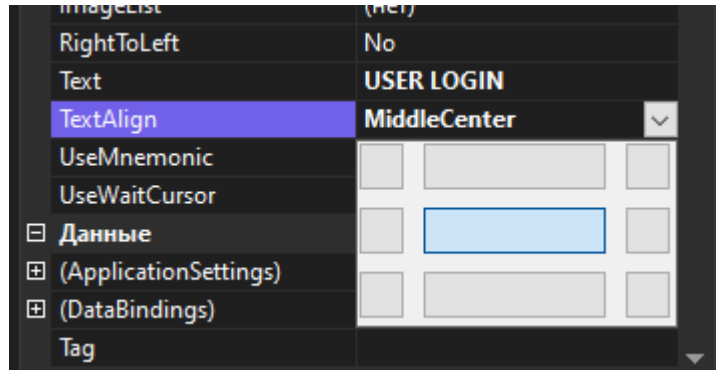


Рисунок 16 - Свойство TextAlign

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

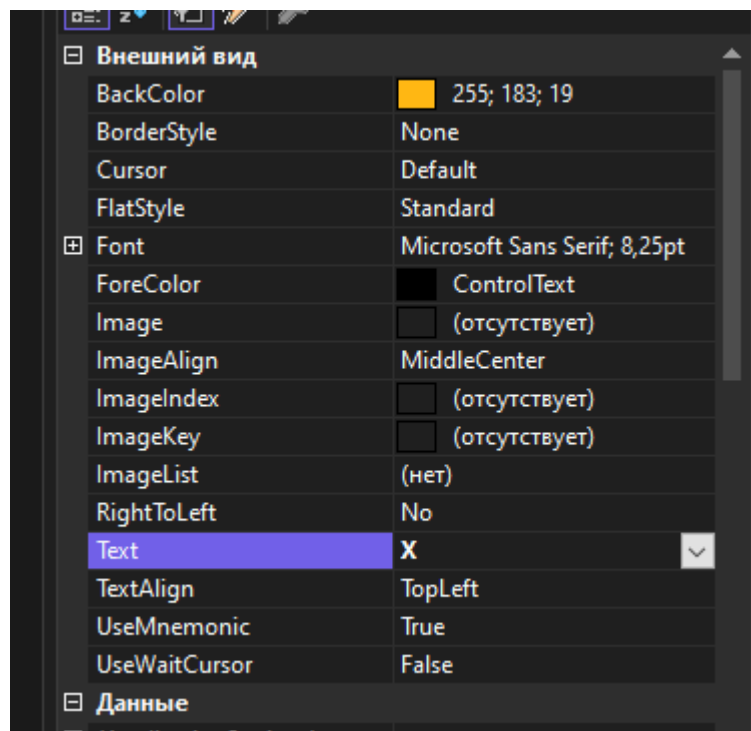


Рисунок 17 - Текст label

Enabled	True
TabIndex	1
UseCompatibleTextRenderin	False
Visible	True
Разработка	
(Name)	labelclose
GenerateMember	True
Locked	False

Рисунок 18 - Имя label

Cursor	Default
FlatStyle	Standard
Font	Microsoft Sans Serif; 12pt
Name	ab Microsoft Sans Serif
Size	12
Unit	Point
Bold	False
GdiCharSet	204
GdiVerticalFont	False
Italic	False

Рисунок 19 - Настройки шрифта

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

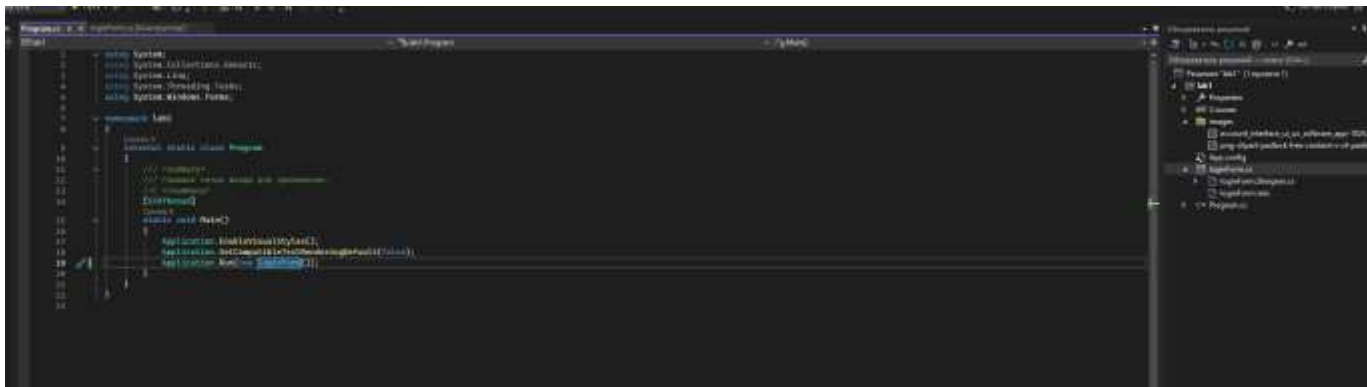


Рисунок 20 - Program.cs

17. Промежуточный результат

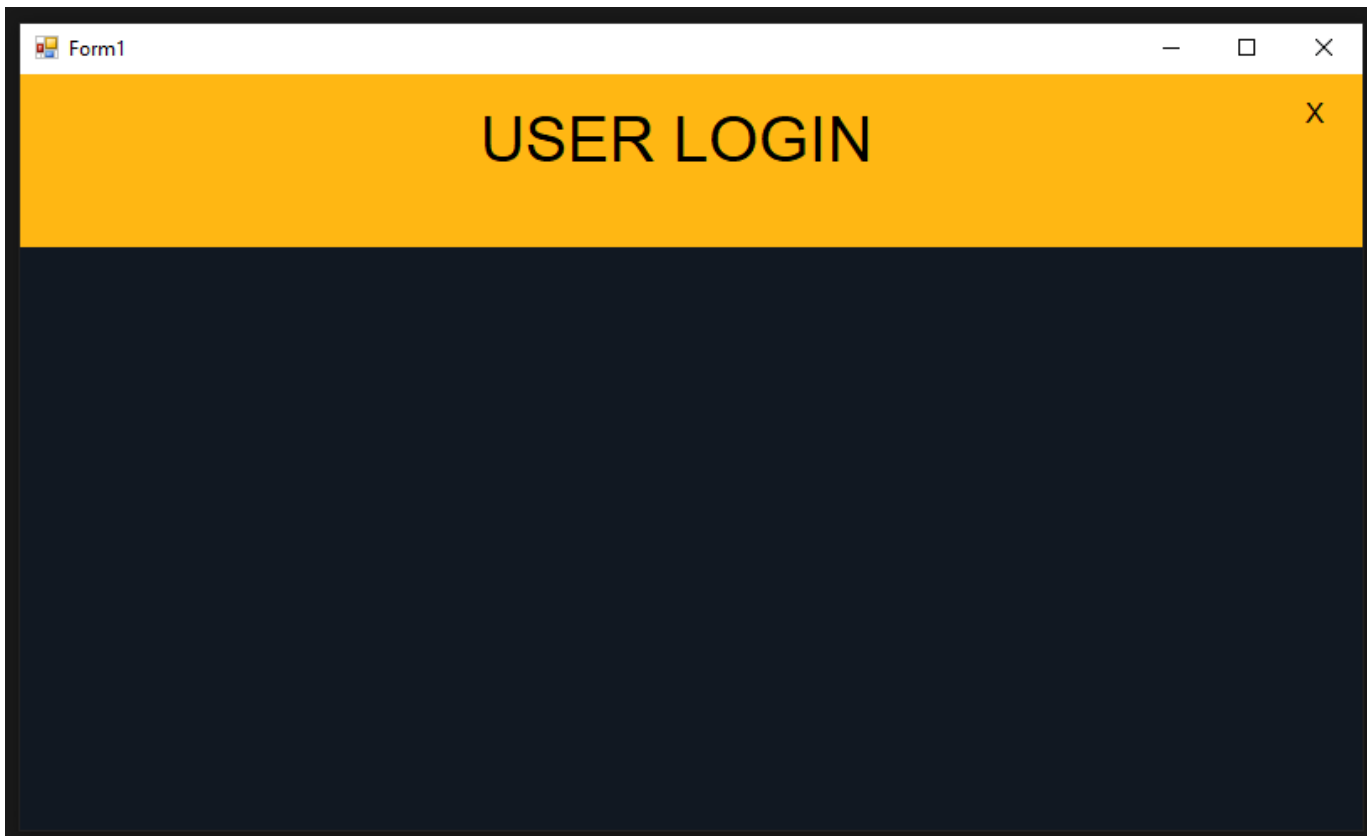


Рисунок 21 - Промежуточный результат

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

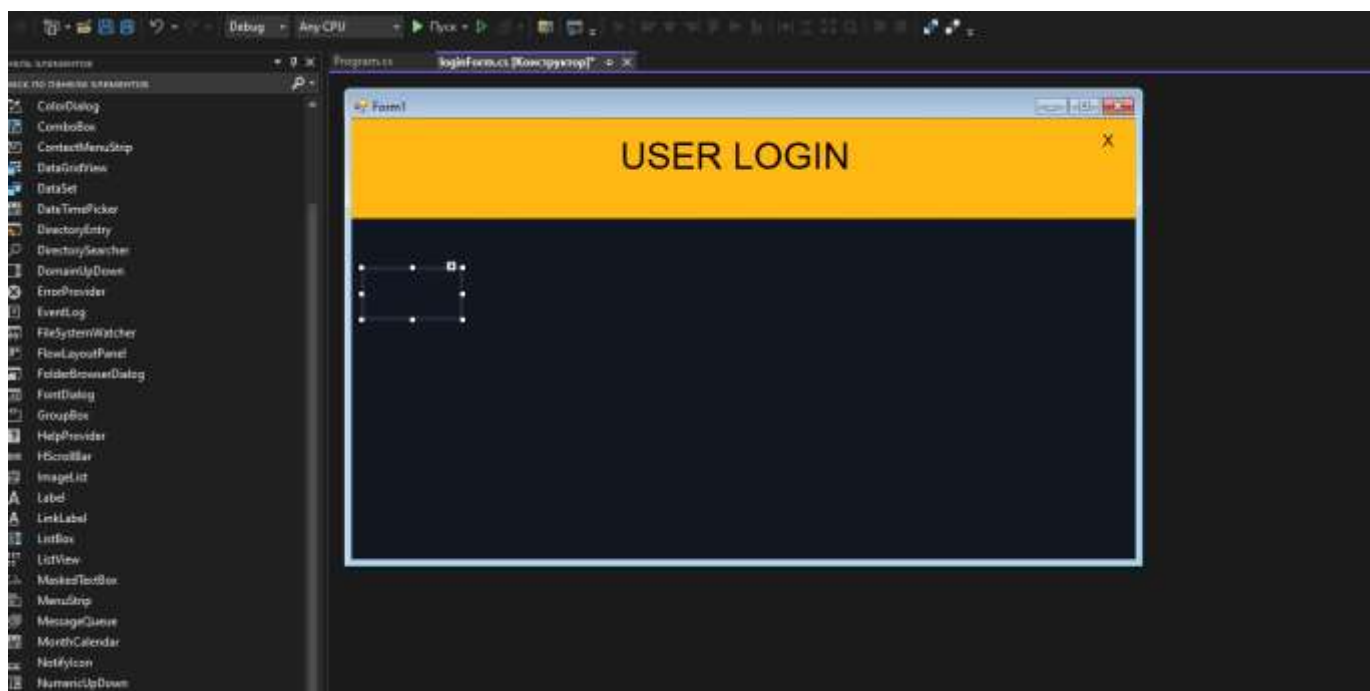


Рисунок 22 - pictureBox

Anchor	Top, Left
Dock	None
Location	12; 154
Margin	3; 3; 3; 3
MaximumSize	0; 0
MinimumSize	0; 0
Padding	0; 0; 0; 0
Size	50; 50
Поведение	
ContextMenuStrip	(нет)
Enabled	True

Рисунок 23 - Размер pictureBox

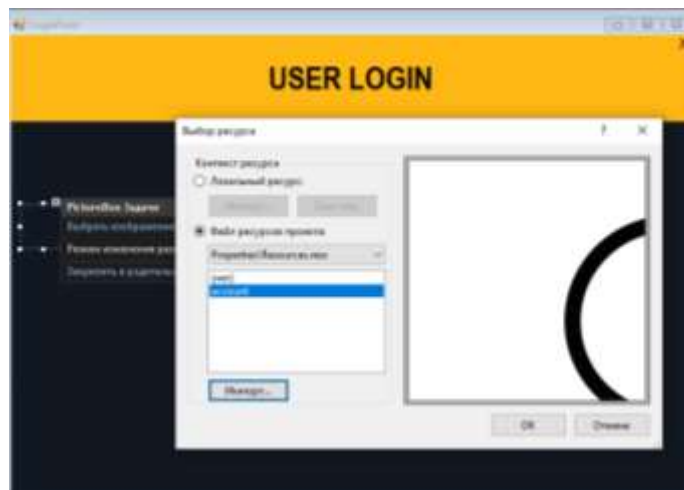


Рисунок 24 - Выбор изображения для pictureBox

19. Готовый pictureBox

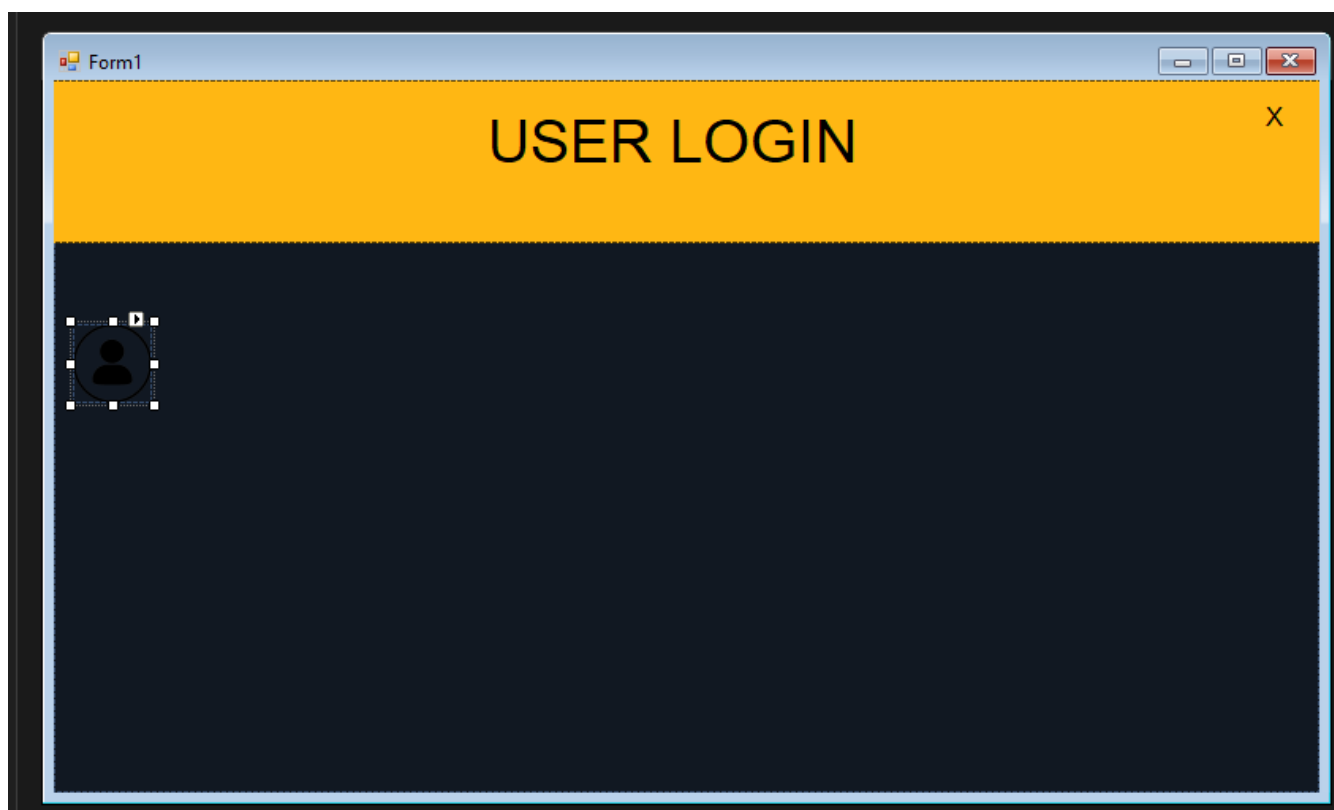


Рисунок 25 - Готовый pictureBox

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

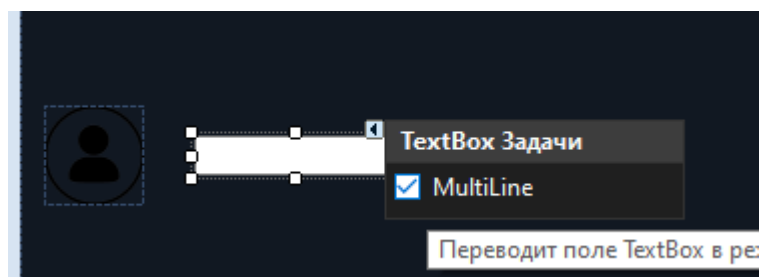


Рисунок 26 - Свойство MultiLine

+	Location	87; 169
+	Margin	3; 3; 3; 3
+	MaximumSize	0; 0
+	MinimumSize	0; 0
+	Size	350; 50
[-]	Поведение	
	AcceptsReturn	False

Рисунок 27 - Размеры textbox

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

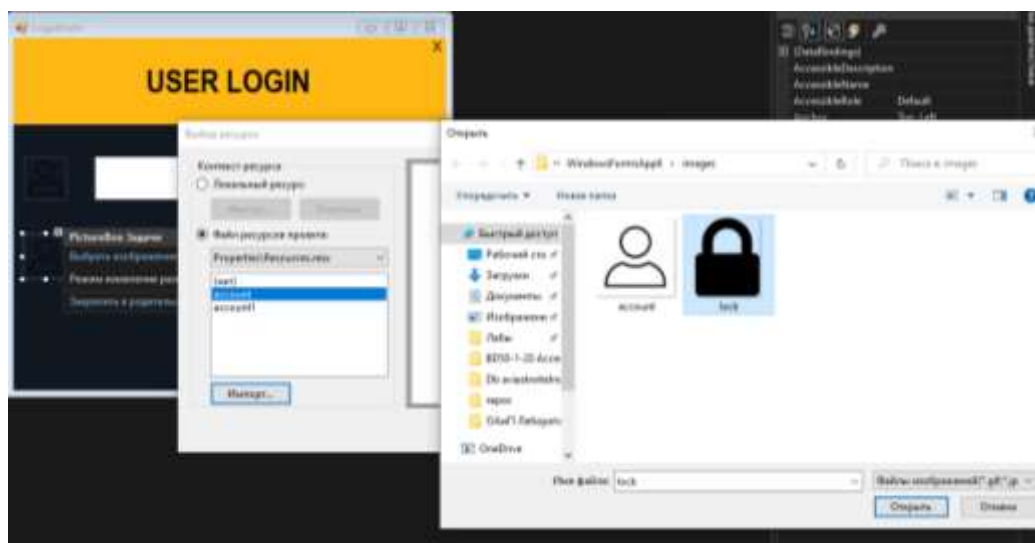


Рисунок 28 - Выбор изображения для pictureBox

22. Промежуточный результат

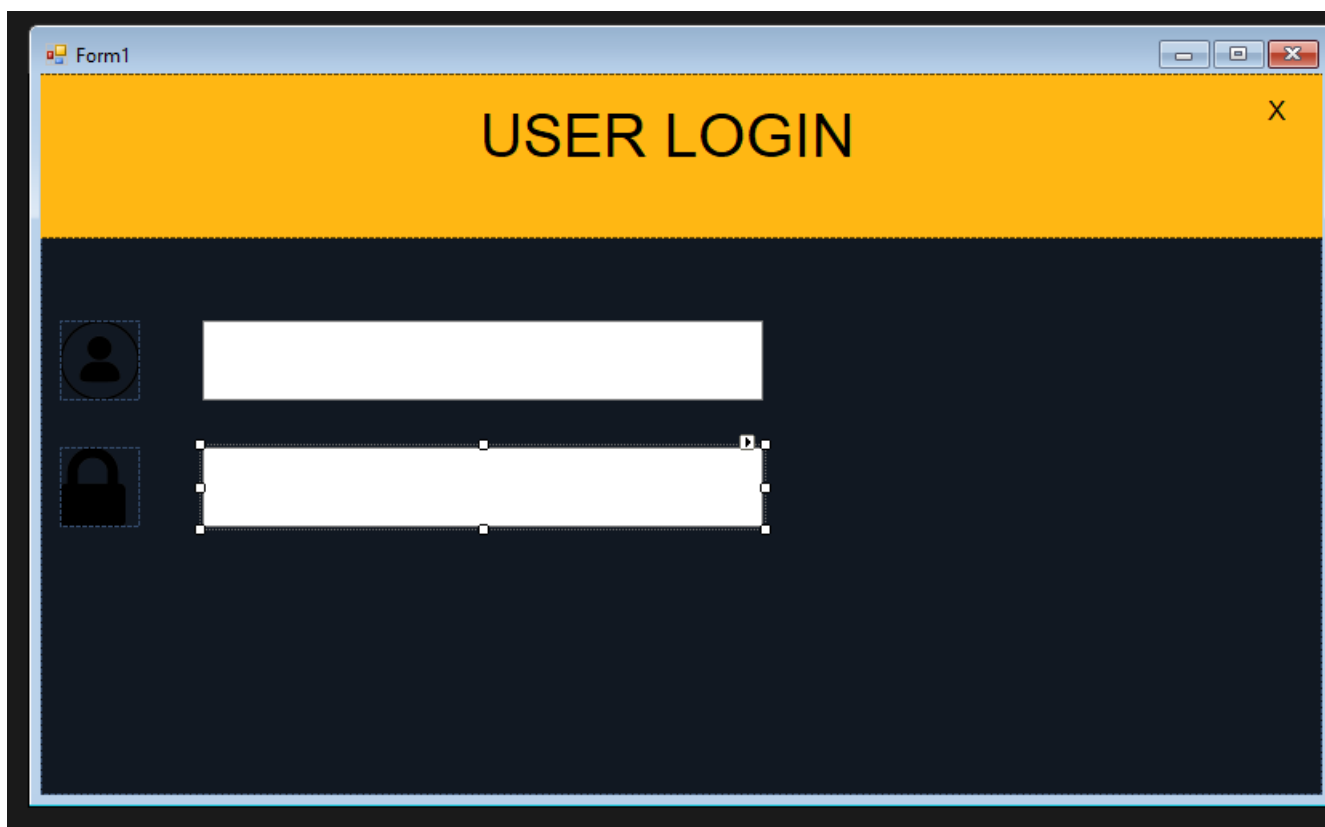


Рисунок 29 - Промежуточный результат

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

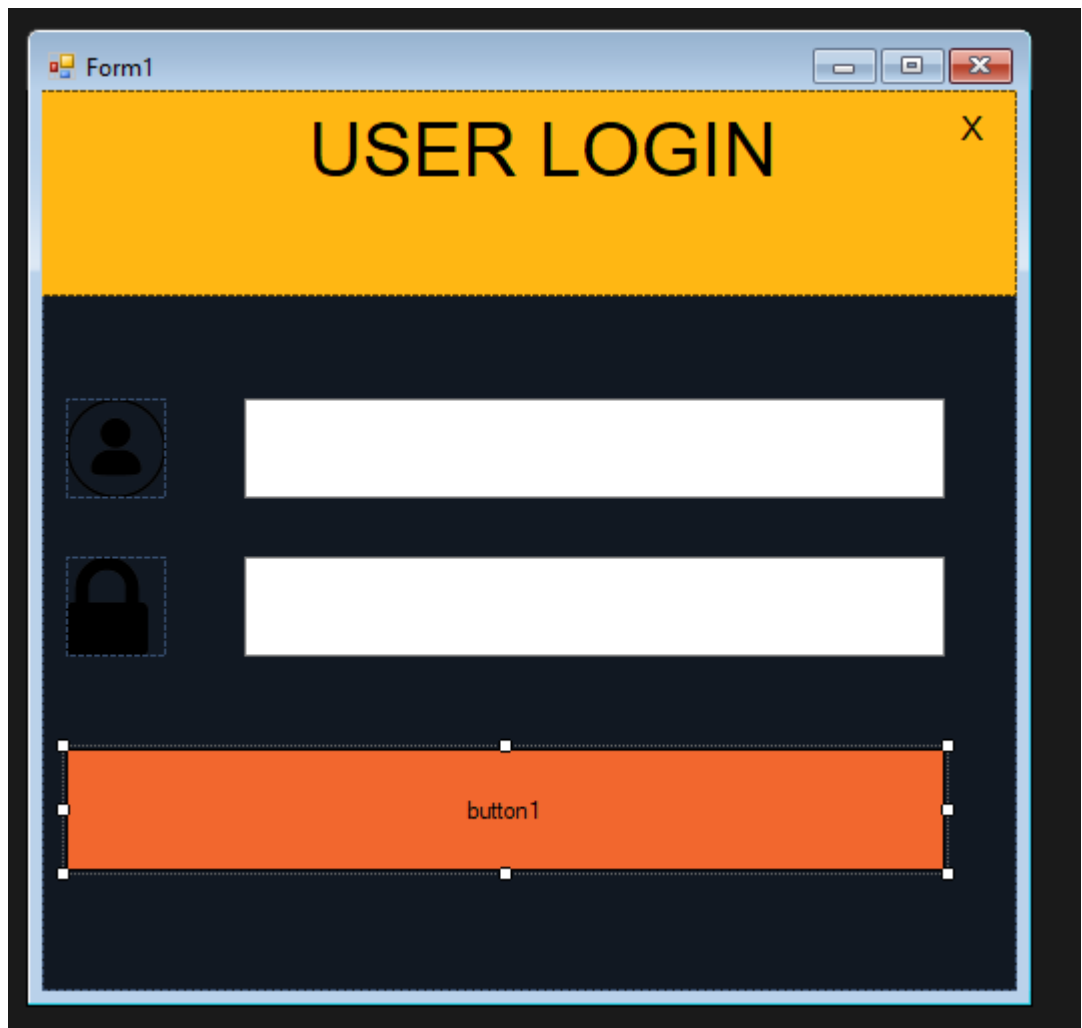


Рисунок 30 - Кнопка

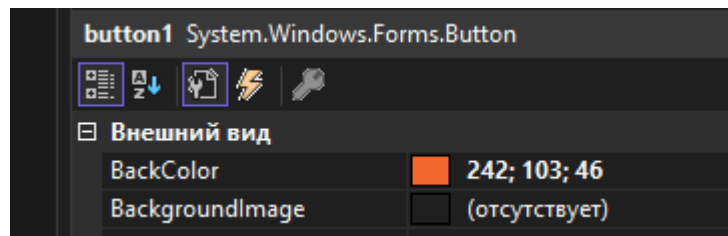


Рисунок 31 - Цвет кнопки

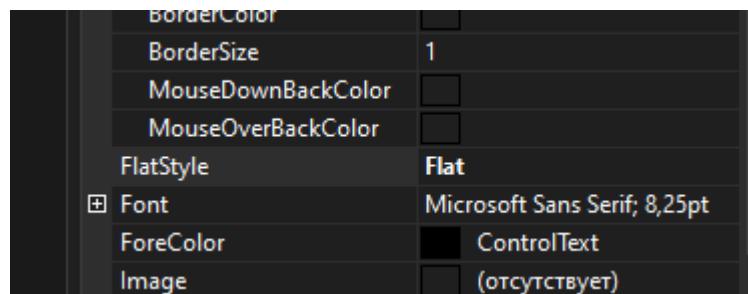


Рисунок 32 - Свойство FlatStyle

24. Настраиваем шрифт кнопки и цвет текста

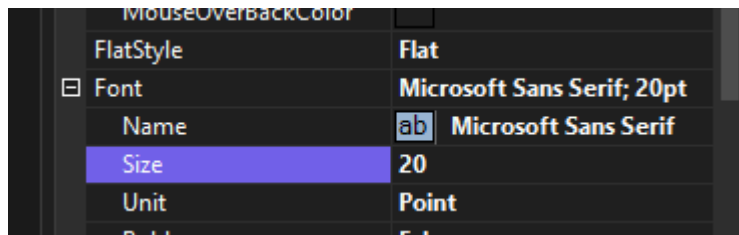


Рисунок 33 - Настройки шрифта

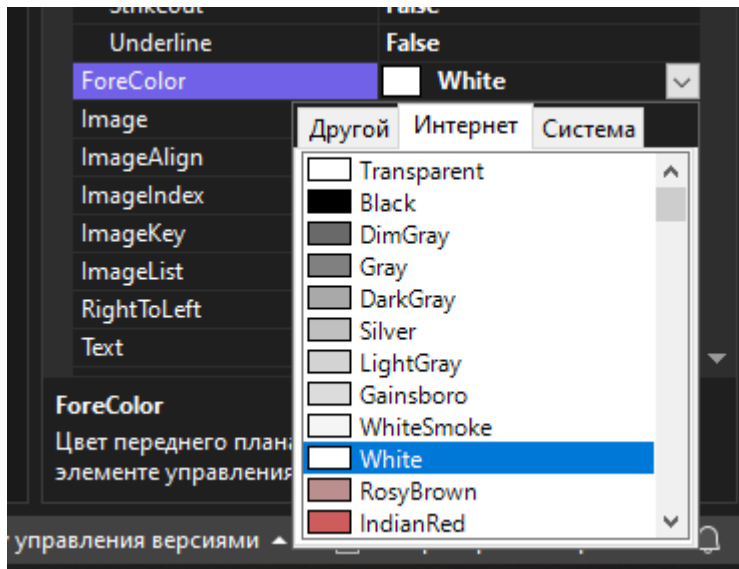


Рисунок 34 - Цвет текста

25. Убираем контур у кнопки

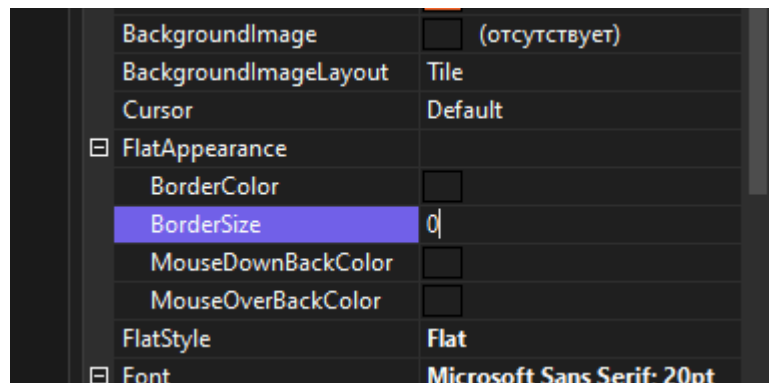


Рисунок 35 - Контур кнопки

26. У текстовых полей настраиваем шрифты

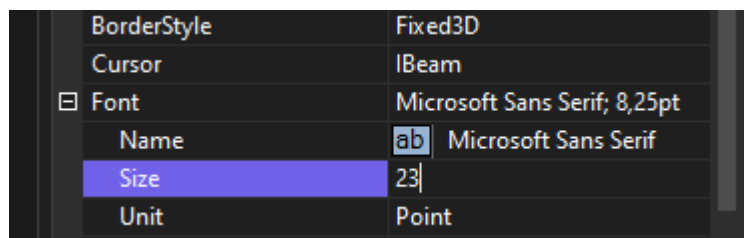


Рисунок 36 - Настройка шрифта

27. Далее переименовываем элементы

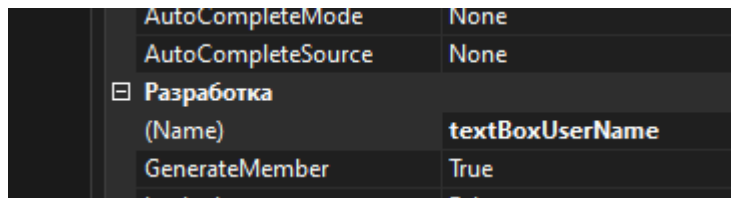


Рисунок 37 - Название textBoxName

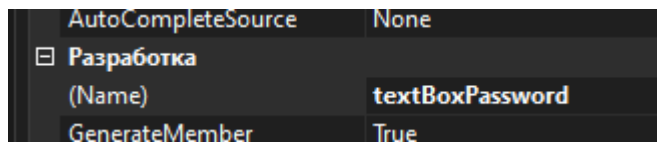


Рисунок 38 - Название textBoxPassword

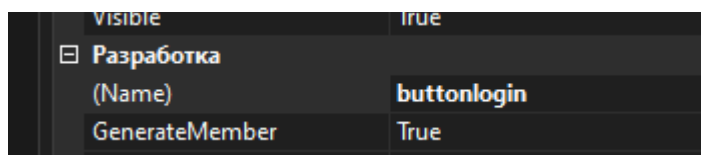


Рисунок 39 - Название buttonLogin

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



Рисунок 40 - Свойство UseSystemPassword

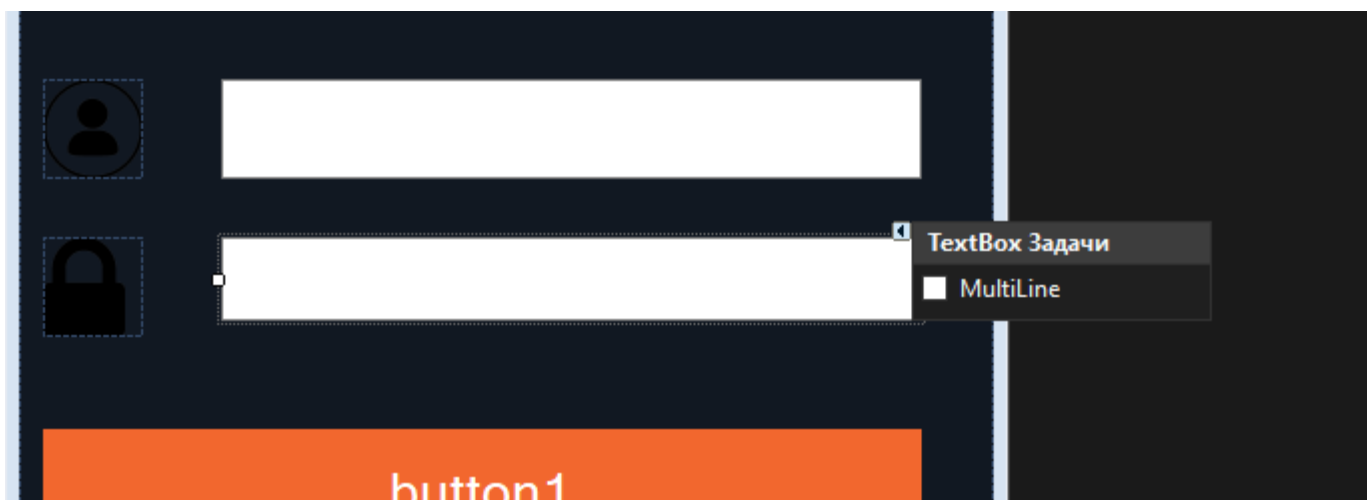
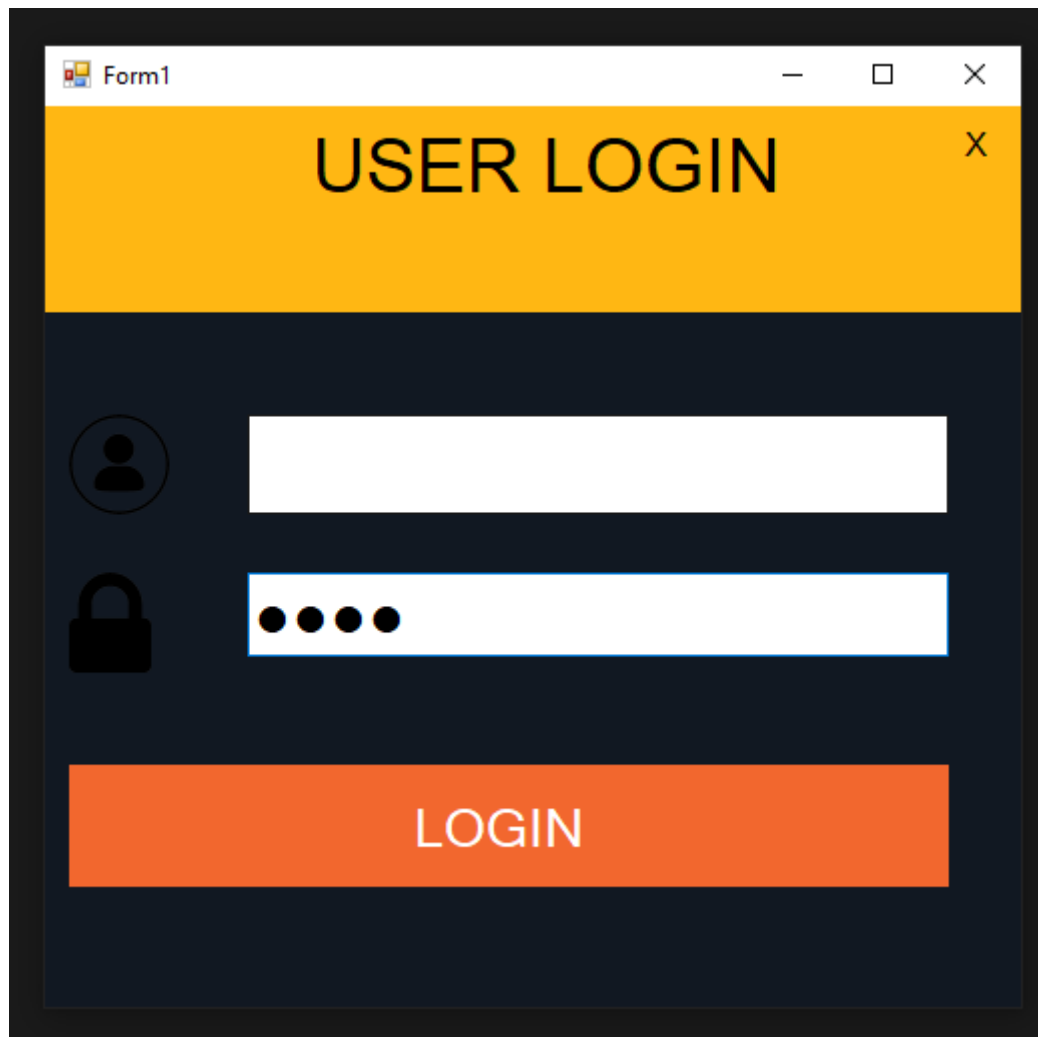


Рисунок 41 - Свойство MultiLine

29. Промежуточный результат



The image shows a screenshot of a Windows application window titled "Form1". The window contains a login form with a yellow header bar at the top that says "USER LOGIN" in black capital letters. Below the header, the form has a dark blue background. On the left side of the form, there are two icons: a person icon for the username field and a padlock icon for the password field. The username field is a white rectangular box, and the password field is a white rectangular box with four black dots representing masked characters. Below these fields is a large orange button with the word "LOGIN" in white capital letters. The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

Рисунок 42 - Промежуточный результат

30. Переходим к коду, задаем textBoxPassword размер

```
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Data;
5 using System.Drawing;
6 using System.Linq;
7 using System.Text;
8 using System.Threading.Tasks;
9 using System.Windows.Forms;
10
11 namespace WindowsFormsApp1
12 {
13     public partial class LoginForm : Form
14     {
15         public LoginForm()
16         {
17             InitializeComponent();
18             textBoxPassword.AutoSize = false;
19             textBoxPassword.Size = new Size(textBoxPassword.Size.Width, 30);
20         }
21     }
22 }
```

Рисунок 43 - Размер textbox

Рисунок 44 - Промежуточный результат

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

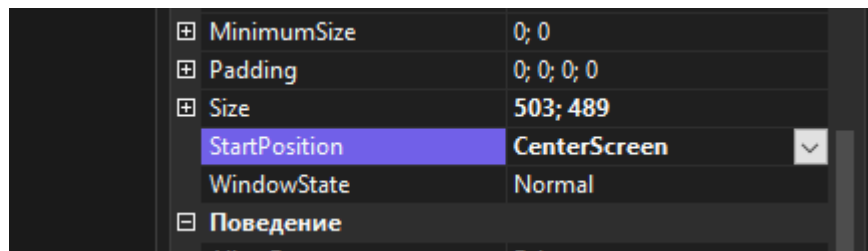


Рисунок 45 - Свойство CenterScreen

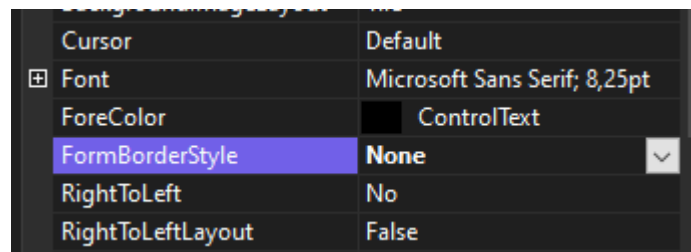


Рисунок 46 - Свойство FormBorderStyle

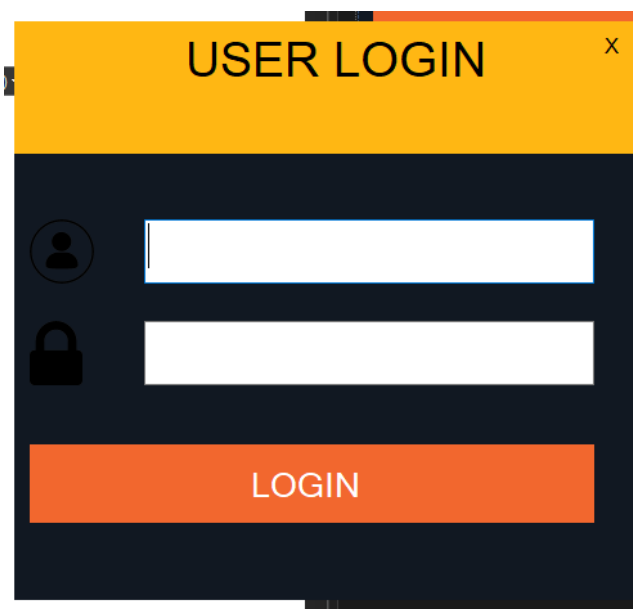


Рисунок 47 - Промежуточный результат

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



Рисунок 48 - Свойство Cursor

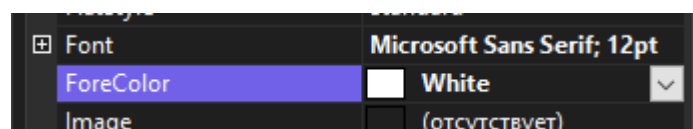


Рисунок 49 - Цвет

FlatStyle	Standard
Font	Arial; 14pt
Name	ab Arial
Size	14
Unit	Point

Рисунок 50 - Настройка шрифта

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

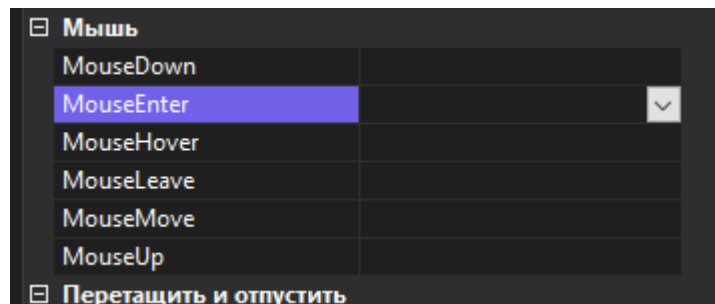


Рисунок 51 - Событие MouseEnter

```

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

```

Рисунок 52 - Код события MouseEnter

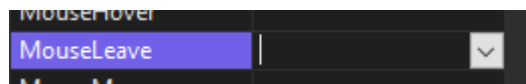


Рисунок 53 - Событие MouseLeave

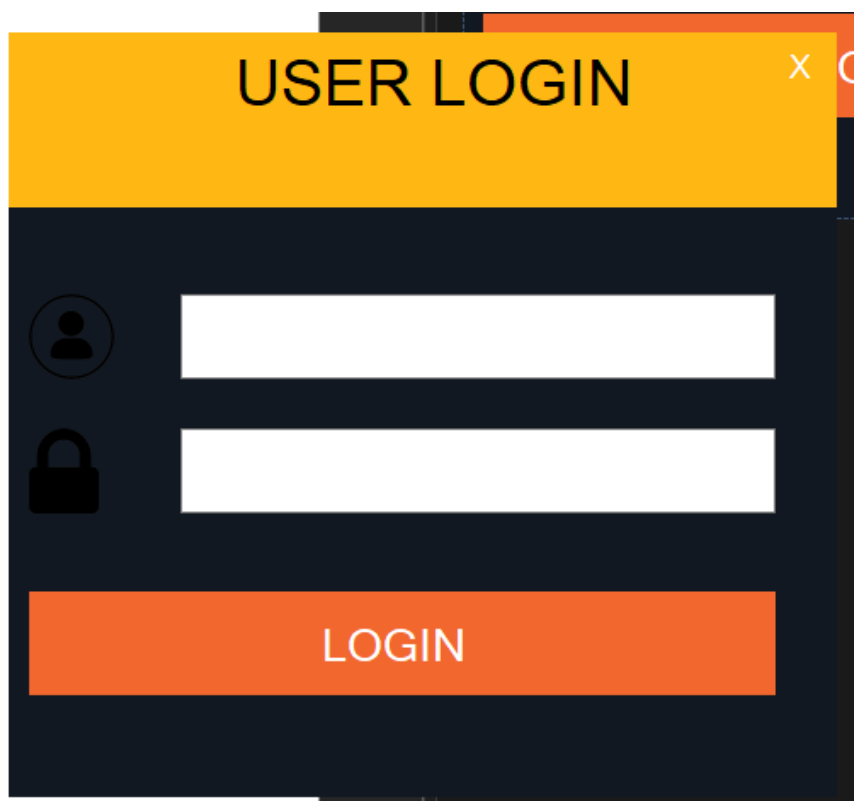
```

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

```

Рисунок 54 – Код события MouseLeave

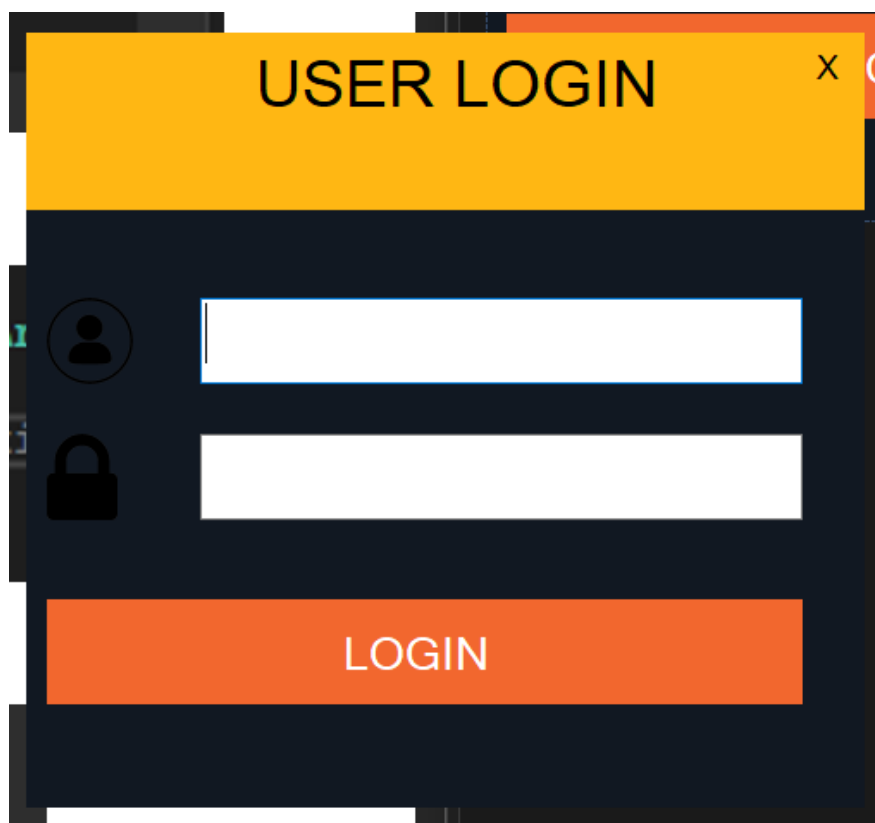
34. Готовая форма



A user login form with a dark blue background. At the top is a yellow title bar with the text "USER LOGIN" and a close button "X". Below the title bar are two white input fields. The first input field 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".

Рисунок 55 -Готовая форма

35. Форма при наведении курсора на кнопку закрыть



A user login form with a dark blue background. At the top is a yellow title bar with the text "USER LOGIN" and a close button "X". Below the title bar are two white input fields. The first input field 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". The close button "X" in the title bar is highlighted with a blue border, indicating it is the active element.

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

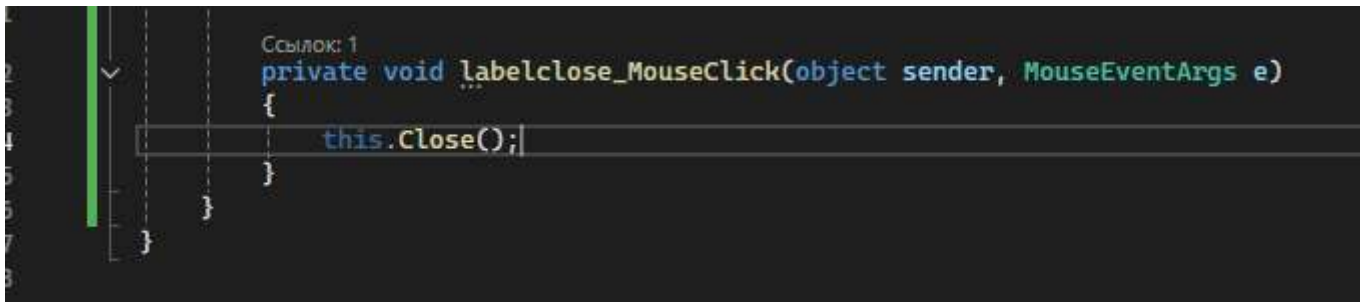


Рисунок 57 - Функция labelClose_Click

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

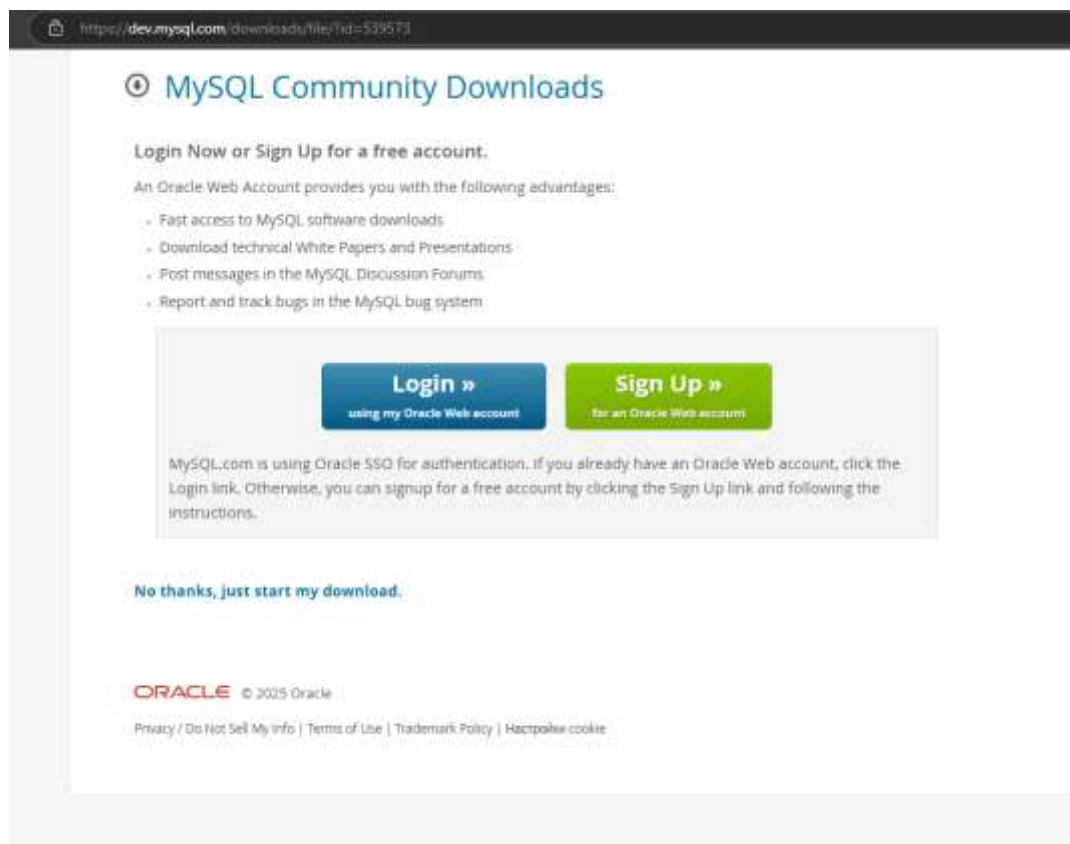


Рисунок 58 - Сайт mysql connector

3. Далее необходимо выполнить установку

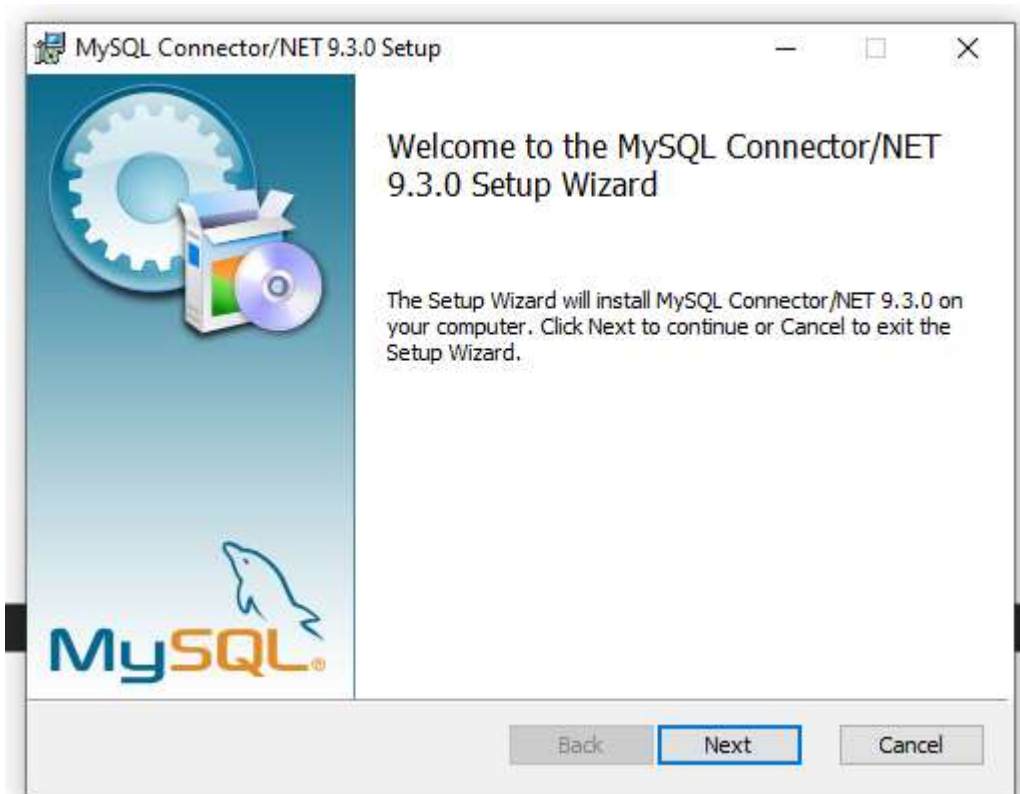


Рисунок 59 - Установка

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

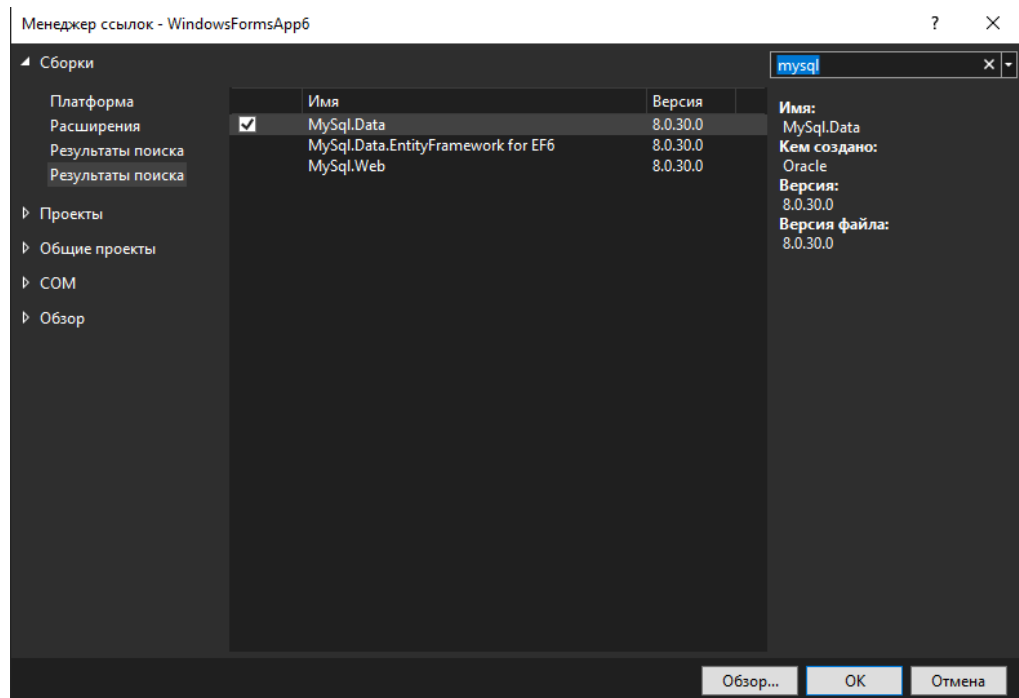


Рисунок 59 - Менеджер ссылок

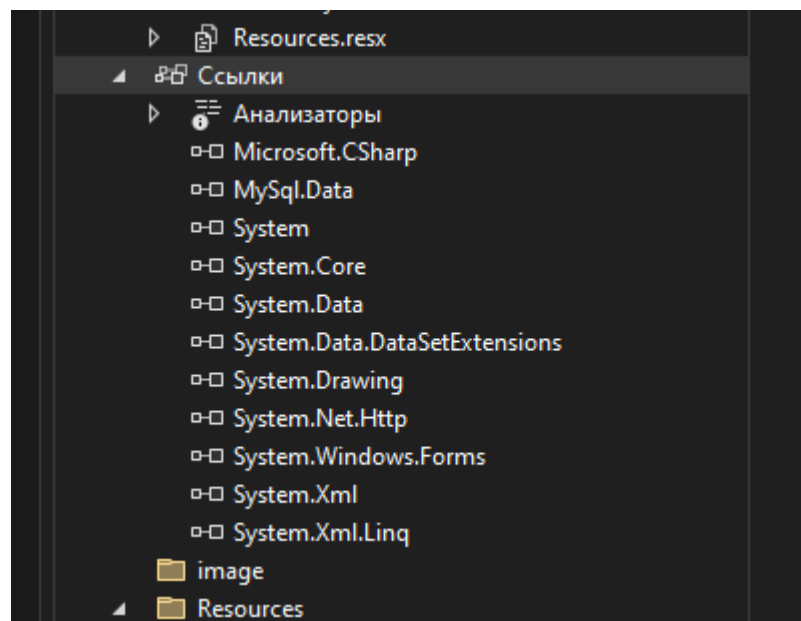


Рисунок 60 - Добавленные ссылки

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

```
using MySql.Data.MySqlClient;
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;

namespace WindowsFormsApp6
{
    Ссылка: 3
    public partial class LoginForm : Form
    {
        MySqlConnection connection = new MySqlConnection();
    }
}
```

Рисунок 61 - Пространство имен

6. Запускаем сервер и заходим в PhpMyAdmin

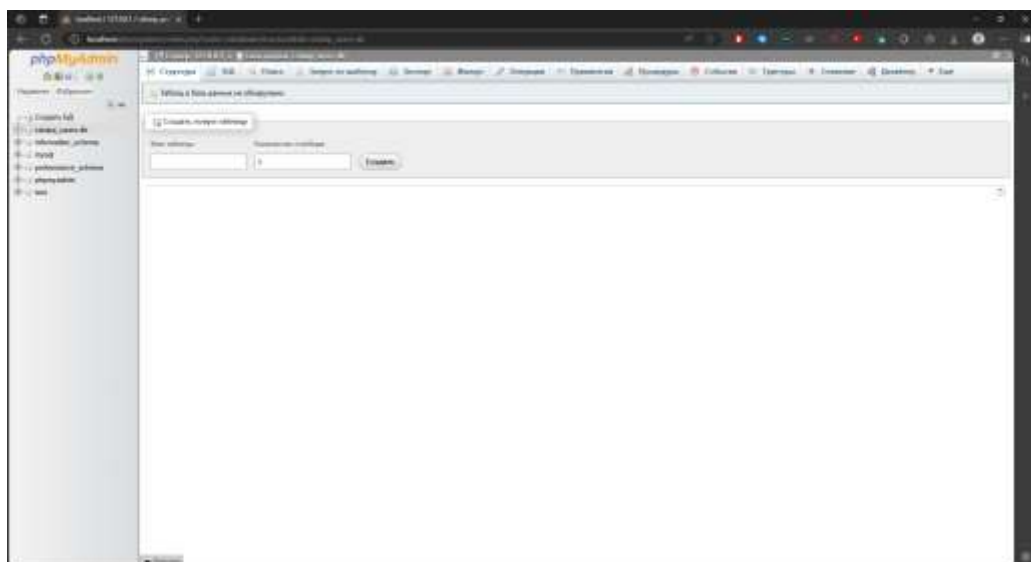


Рисунок 62 - PhpMyAdmin

7. Создаем новую бд под названием csharp_users_db

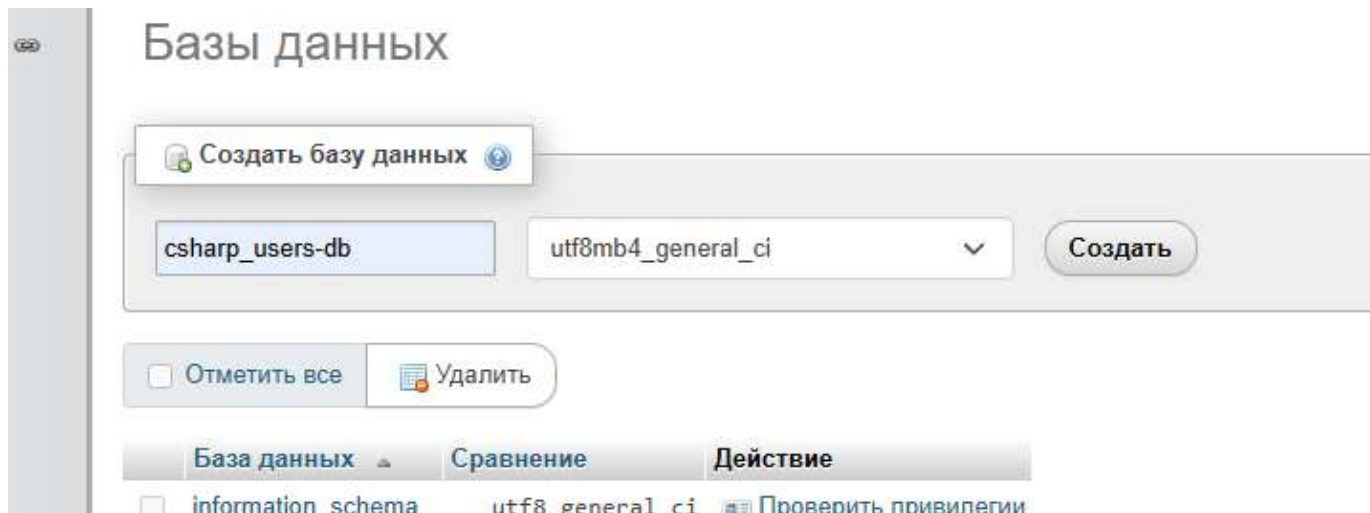


Рисунок 63 - Создание бд

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

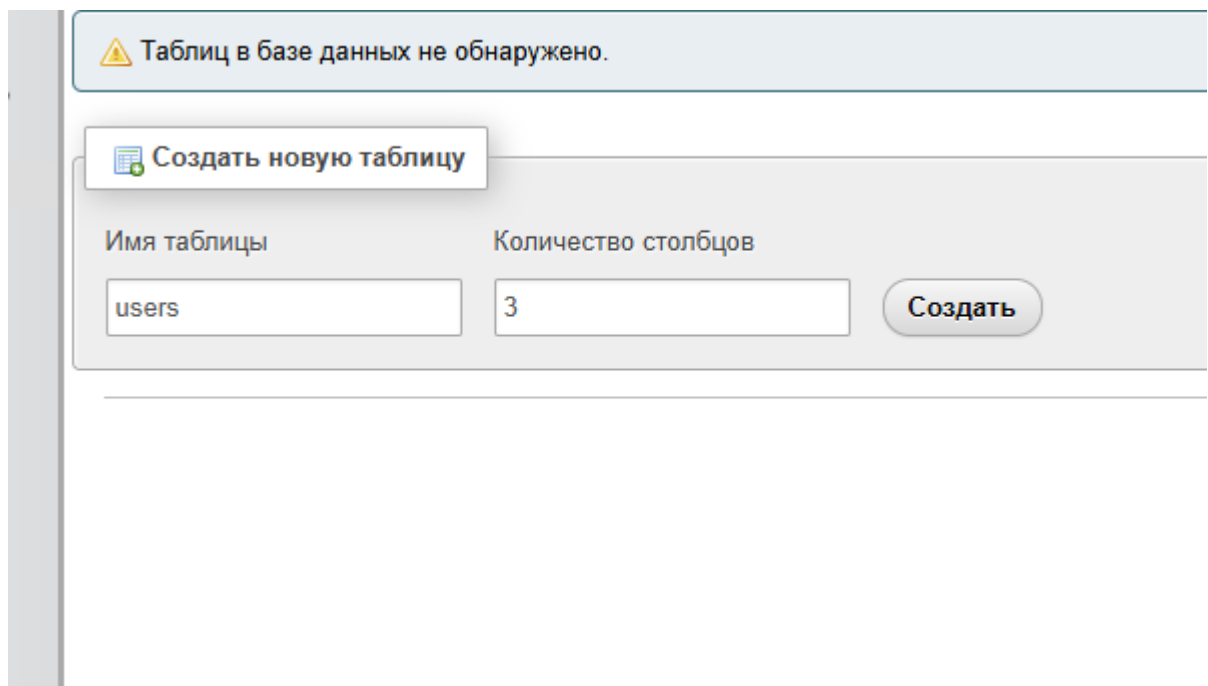


Рисунок 64 - Создание таблицы

9. Заполняем в таблице атрибуты

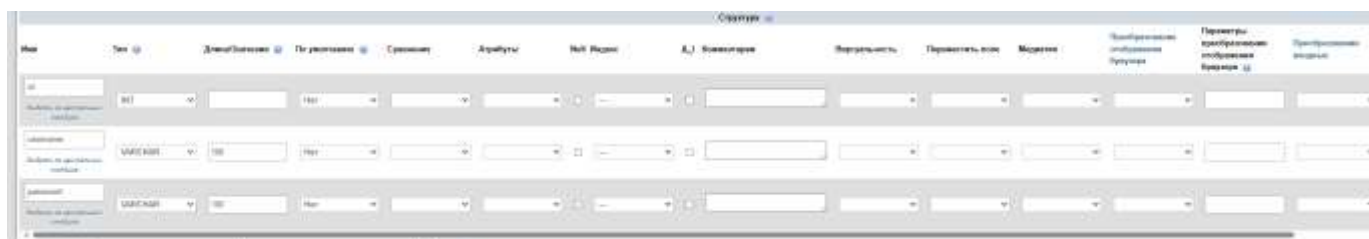


Рисунок 65 - Заполнение таблицы

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

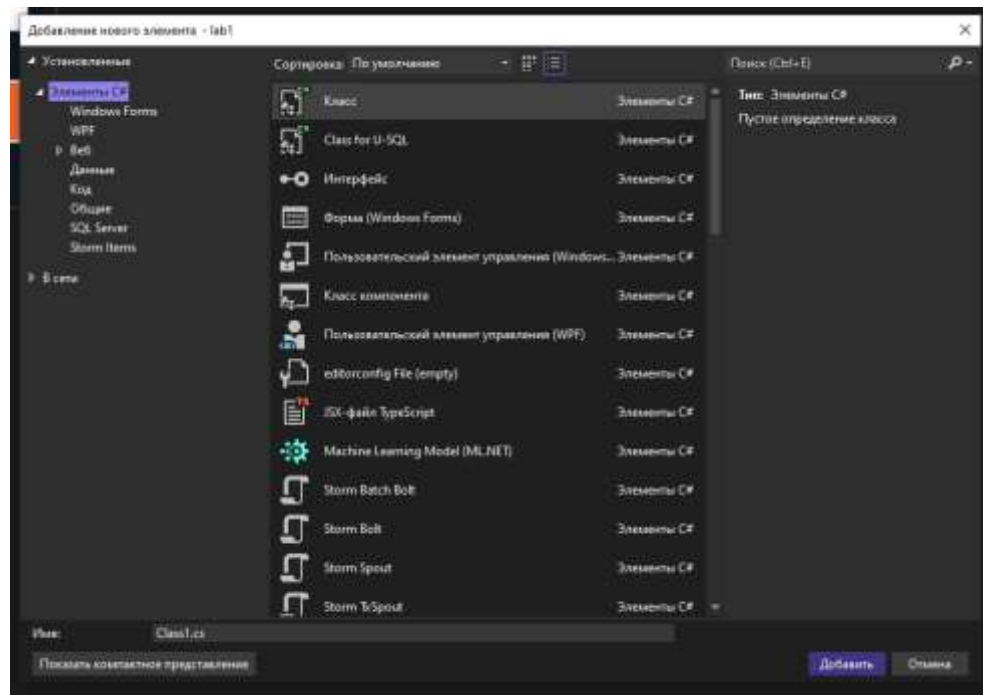


Рисунок 67 - Создание нового класса

12. Копируем наше подключение бд в новый созданный класс

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

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

Рисунок 68 - Созданный класс

13. Пишем функцию для открытия соединений

```

Ссылка: 0
public void openConnection()
{
    if(connection.State == System.Data.ConnectionState.Closed)
    {
        connection.Open();
    }
}

```

Рисунок 69 - Открытие соединения

14. Создаем функцию для закрытия соединения

```

}
Ссылка: 0
public void closeConnection()
{
    if (connection.State == System.Data.ConnectionState.Open)
    {
        connection.Close();
    }
}

```

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' = @user and 'password' = @pass", db.getConnection());

    command.Parameters.Add("@user", 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

Сервер: 127.0.0.1 » База данных: csharp_users-db » Таблица: users

Обзор Структура SQL Поиск Вставить Экспорт Импорт Привилегии Операции

Столбец	Тип	Функция	Null	Значение
id	int(11)			
username	varchar(100)			user1
password	varchar(100)			pass

☒ Игнорировать

Вперёд

Столбец	Тип	Функция	Null	Значение
id	int(11)			

Рисунок 73 - Внесения данных в бд

Тестируем результат

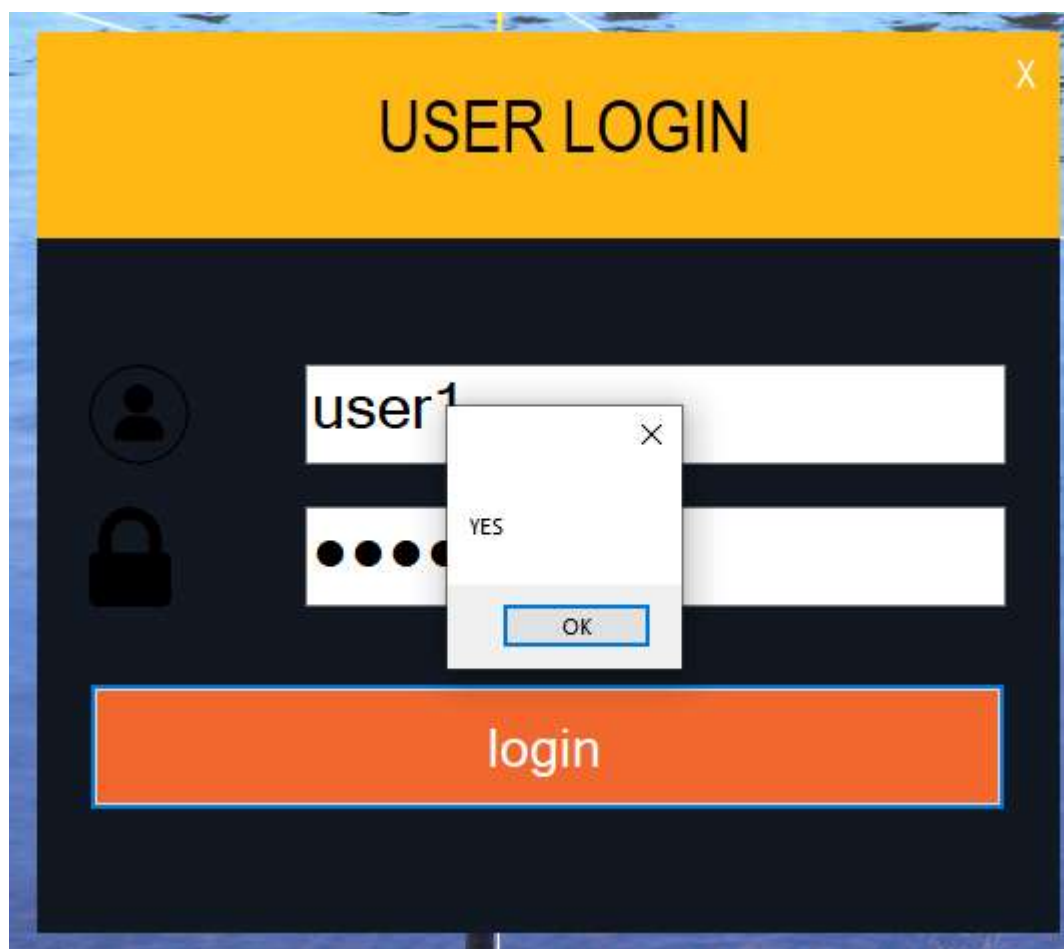


Рисунок 74 - Результат при введении верных логин и пароль

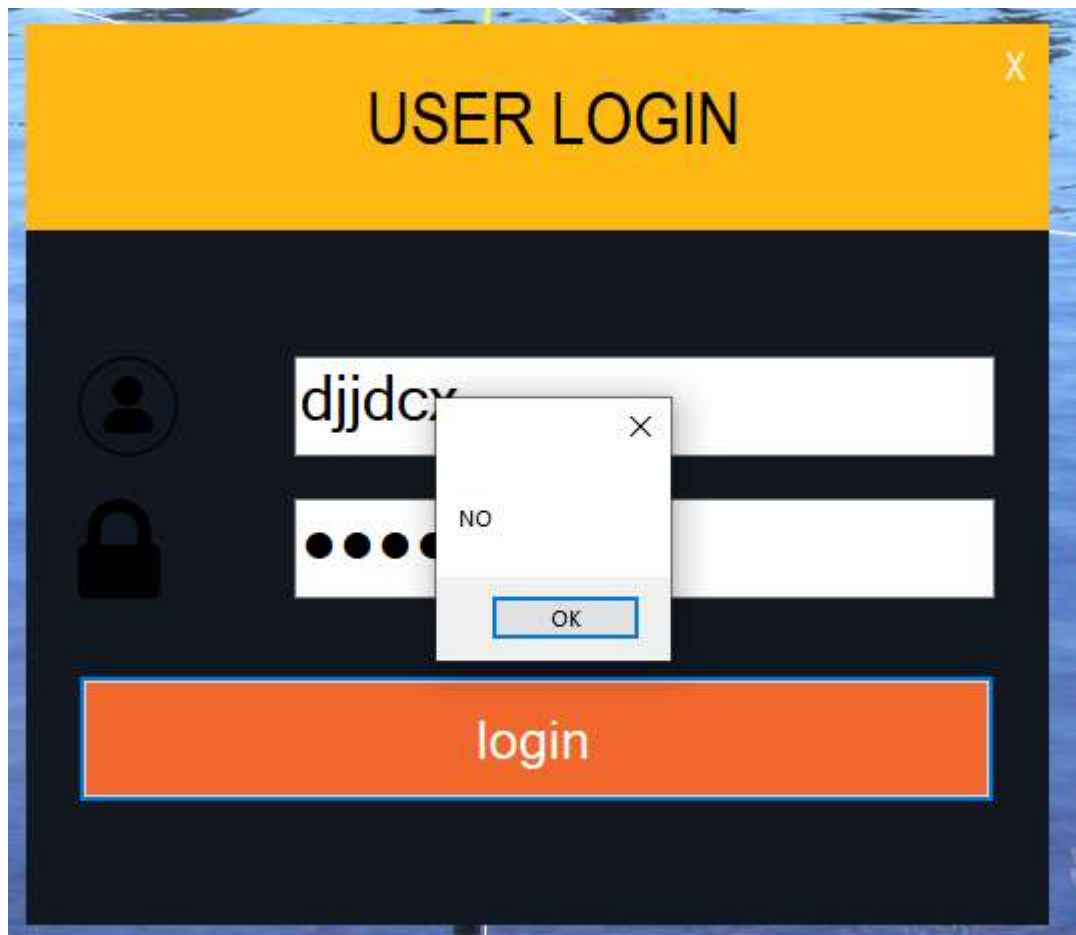


Рисунок 75 - Результат при введении неверных логин и пароль

Добавляем новую форму в проект

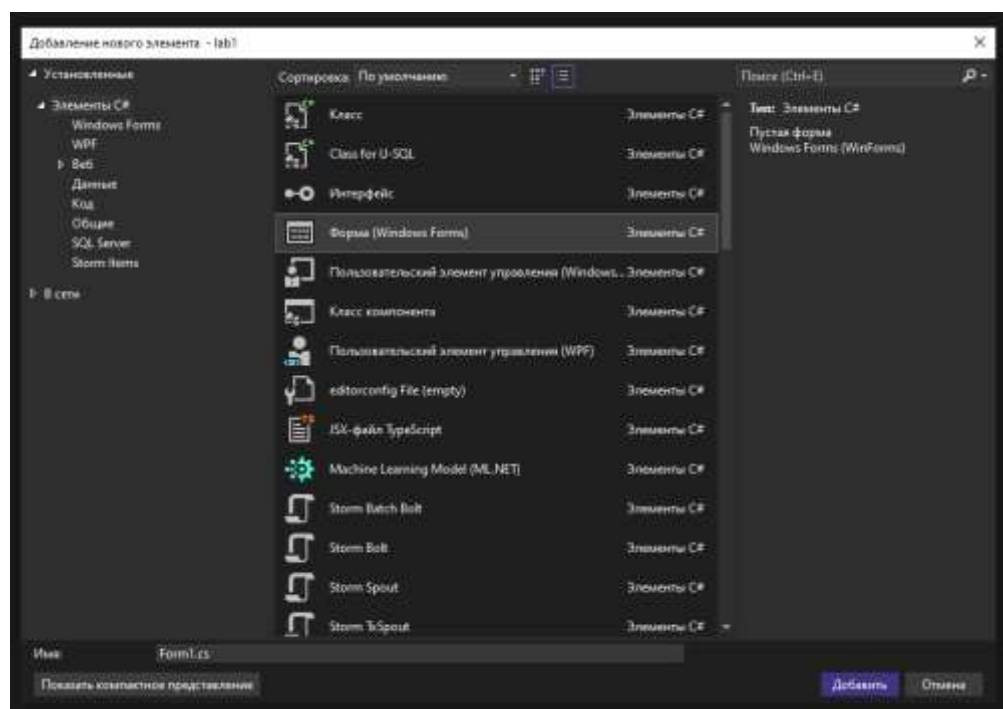


Рисунок 57 - Новая форма

Копируем содержимое LoginForm на новую созданную форму RegisterForm

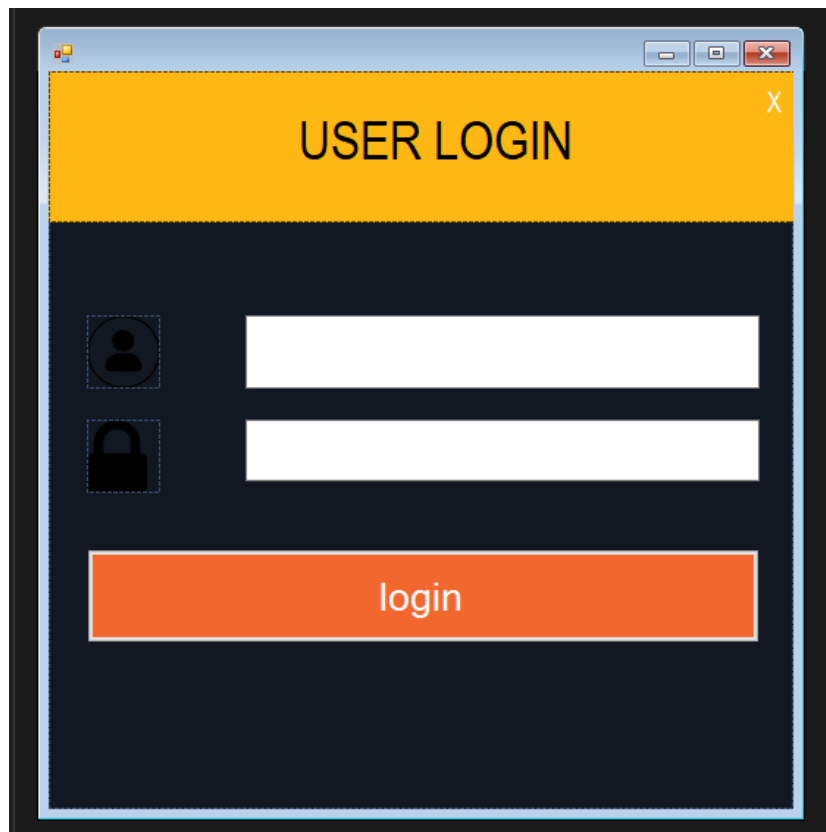


Рисунок 58 - Скопированная LoginForm

Редактируем, добавляем textbox

A screenshot of a Windows-style application window titled 'CREATE ACCOUNT'. The window has a yellow header bar with the title and a close button (X). The main area has a dark blue background. There are six white text input fields arranged in a single column. Below the input fields is a large orange button with the text 'login' in white.

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

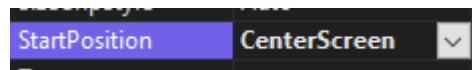


Рисунок 61 – StartPosition

Переименовываем все textbox

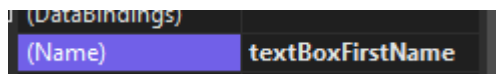


Рисунок 62 - textBoxFirstName

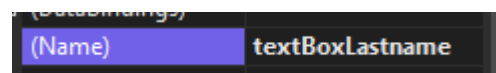


Рисунок 63 - textBoxLastName

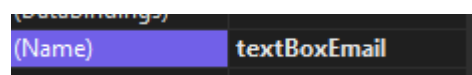


Рисунок 64 - textBoxEmail

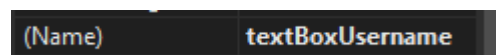


Рисунок 65 - textBoxUsername

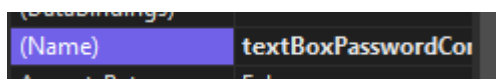


Рисунок 66 - textBoxPasswordConfirm

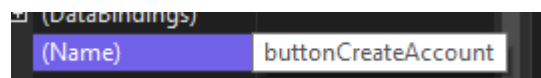


Рисунок 67 – buttonCreateAccountx

Промежуточный результат

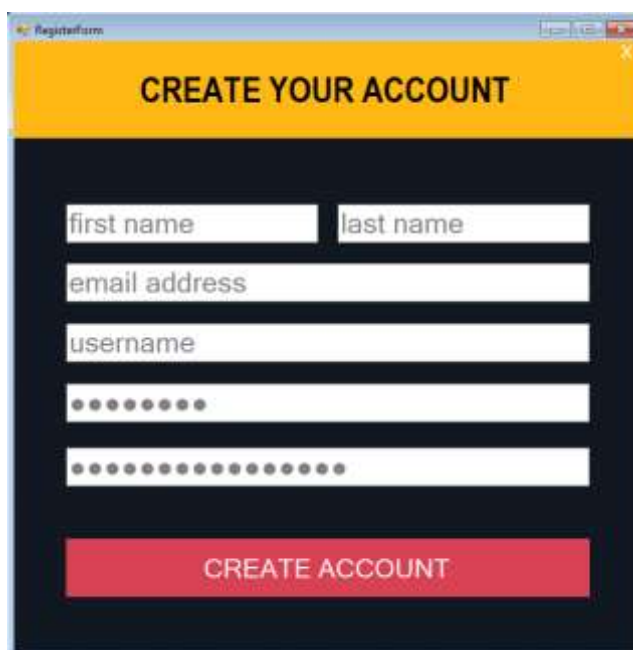
A screenshot of a Windows application window titled 'RegisterForm'. The window has a yellow header bar with the text 'CREATE YOUR ACCOUNT'. Below the header, there is a dark blue background with several white text input fields. The fields are labeled 'first name', 'last name', 'email address', 'username', 'password', and 'confirm password'. The 'password' and 'confirm password' fields are currently masked with black dots. At the bottom of the form, there is a red button with the text 'CREATE ACCOUNT'.

Рисунок 68 - Промежуточный результат

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

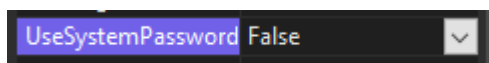


Рисунок 69 - UseSystemPassword

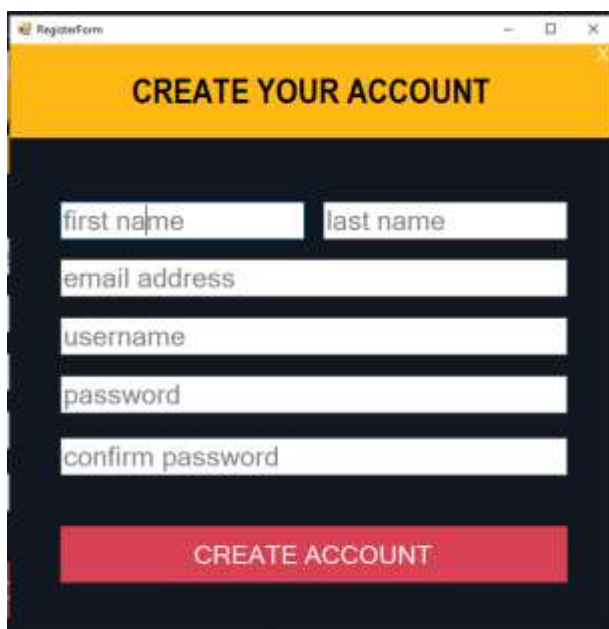
A screenshot of the same 'CREATE YOUR ACCOUNT' form as in Figure 68. However, the 'password' and 'confirm password' fields are now visible, showing the text 'password' and 'confirm password' respectively. The 'CREATE ACCOUNT' button remains at the bottom.

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

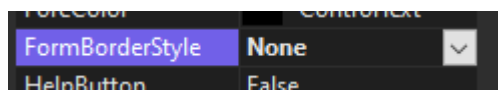


Рисунок 80 - FormBorderStyle

После выполнения всех действий получаем данный результат

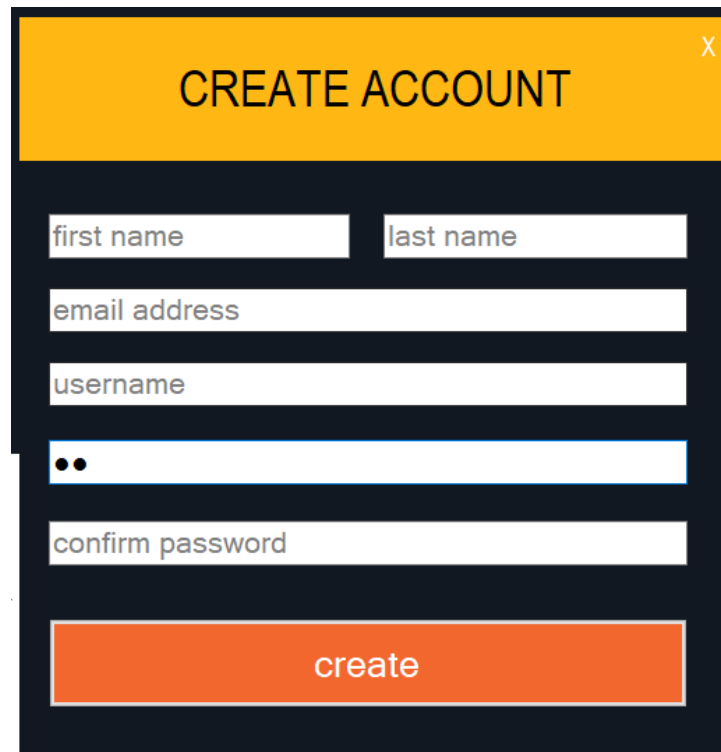


Рисунок 81 - Результат

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

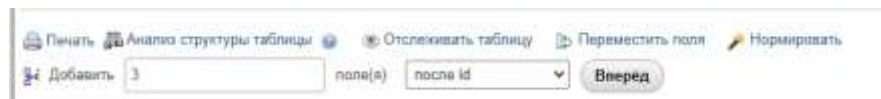


Рисунок 82 - Добавление столбцов в таблицу

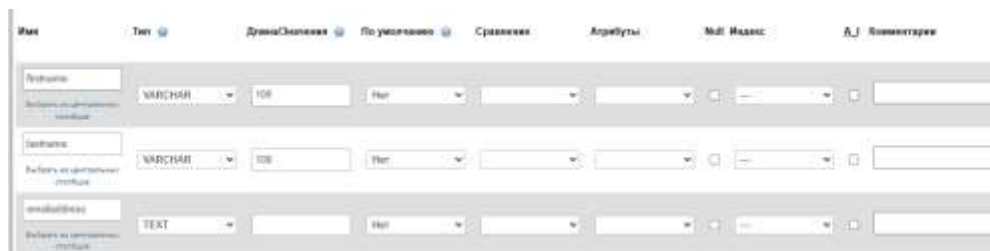


Рисунок 83 - Структура таблицы

В пространство имен подключаем MySQL

```

51
1      using System;
2      using System.Collections.Generic;
3      using System.ComponentModel;
4      using System.Data;
5      using System.Drawing;
6      using System.Linq;
7      using System.Text;
8      using System.Threading.Tasks;
9      using System.Windows.Forms;
10     using MySql.Data.MySqlClient;
11

```

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»

```

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;

namespace AccountCreation
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void buttonCreateAccount_Click(object sender, EventArgs e)
        {
            string db = "db";
            MySqlCommand command = new MySqlCommand("INSERT INTO users (first_name, last_name, email_address, username, password) VALUES (@fn, @ln, @email, @username, @password)", db.getConnection());

            command.Parameters.AddWithValue("fn", textBox1.Text);
            command.Parameters.AddWithValue("ln", textBox2.Text);
            command.Parameters.AddWithValue("email", textBox3.Text);
            command.Parameters.AddWithValue("username", textBox4.Text);
            command.Parameters.AddWithValue("password", textBox5.Text);

            db.openConnection();

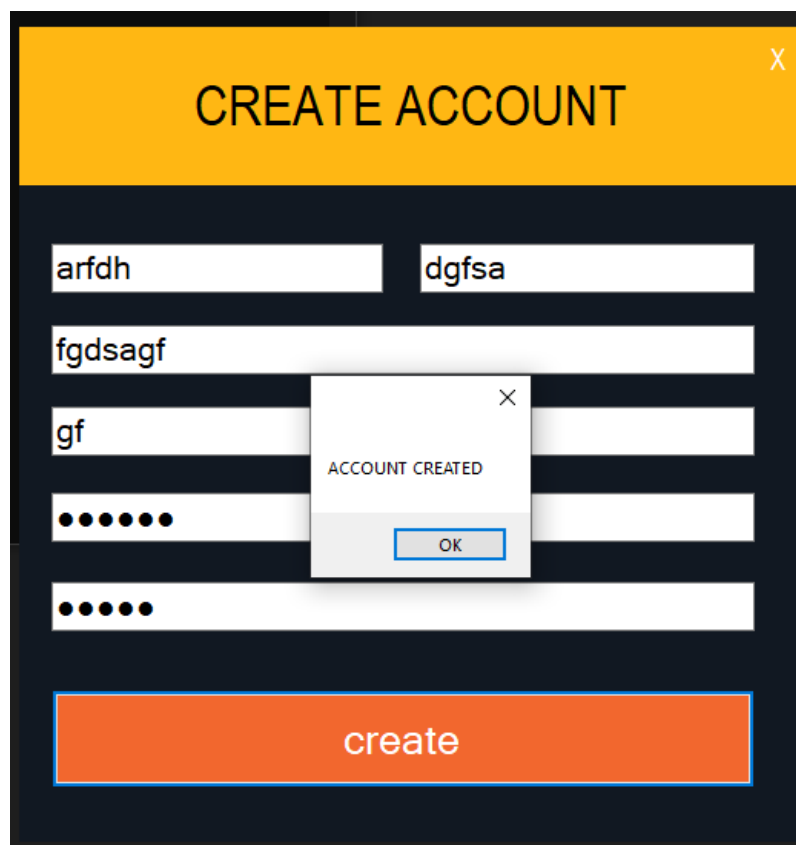
            if (command.ExecuteNonQuery() > 0)
            {
                MessageBox.Show("Account created");
            }
            else
            {
                MessageBox.Show("Error");
            }

            db.closeConnection();
        }
    }
}

```

Рисунок 85 - Функция кнопки создания аккаунта

Промежуточный результат



The image shows a 'CREATE ACCOUNT' form with a dark blue background and an orange header. The form contains several input fields: 'firstname' (value: arfdh), 'lastname' (value: dgfsa), 'emailaddress' (value: fgdsagf), 'username' (value: gf), and 'password' (masked with dots). A modal dialog box is centered over the form, displaying 'ACCOUNT CREATED' and an 'OK' button. At the bottom of the form is a large orange 'create' button.

Рисунок 86 - Промежуточный результат

id	firstname	lastname	emailaddress	username	password
0	arfdh	dgfsa	fgdsagf	gf	123123

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 fields (dots) for email and phone number. A red "CREATE ACCOUNT" button is at the bottom. A modal dialog box is displayed in the center, containing the text "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, but with different data: first name "dddd", last name "dd", password "ddd", and username "user3". The masked fields for email and phone number now contain two dots each. A red "CREATE ACCOUNT" button is at the bottom. A modal dialog box is displayed in the center, containing the text "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 "username" field, displaying the message "ENTER YOUR INFORMATION FIRST".

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

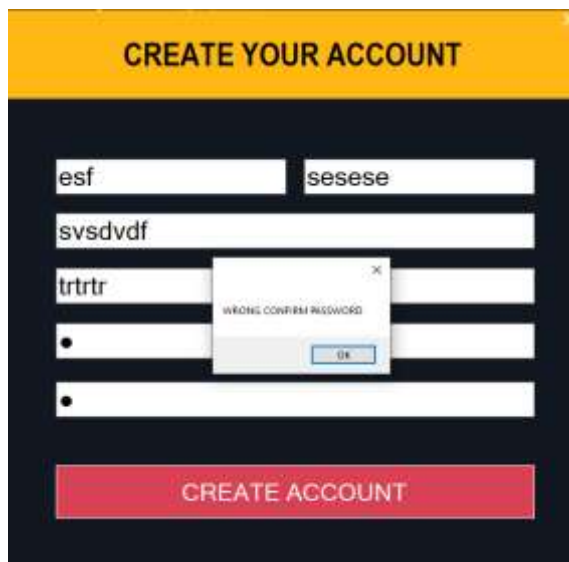


Рисунок 97 – Отрицательный результат проверки пароля

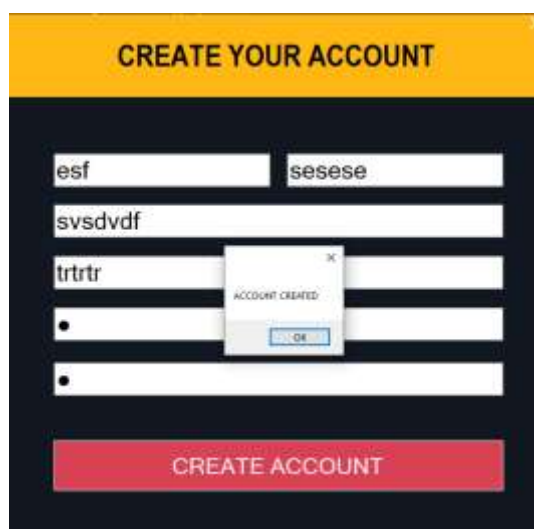


Рисунок 98 – Положительный результат проверки пароля

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

Рисунок 99 - Доработка MessageBox

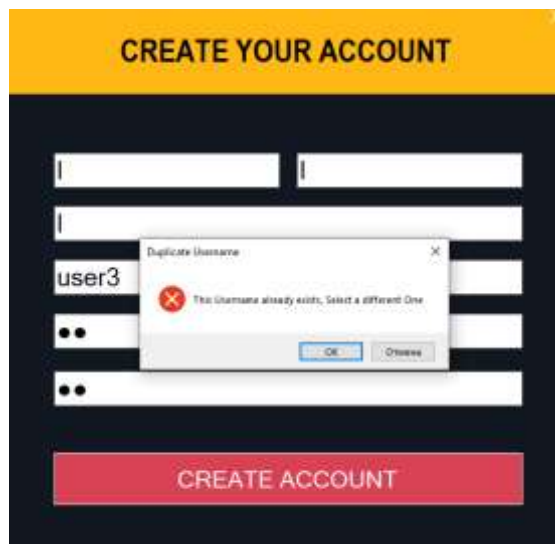


Рисунок 100 - Готовые MessageBox с ошибкой

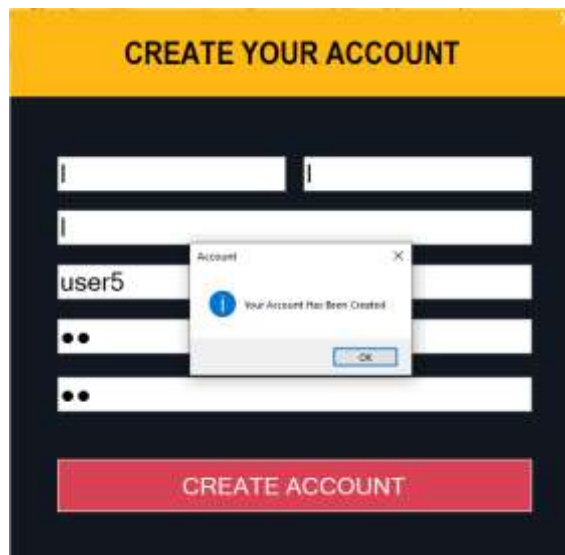


Рисунок 101 - Готовые MessageBox без ошибки

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

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

Рисунок 102 - Доработка MessageBox

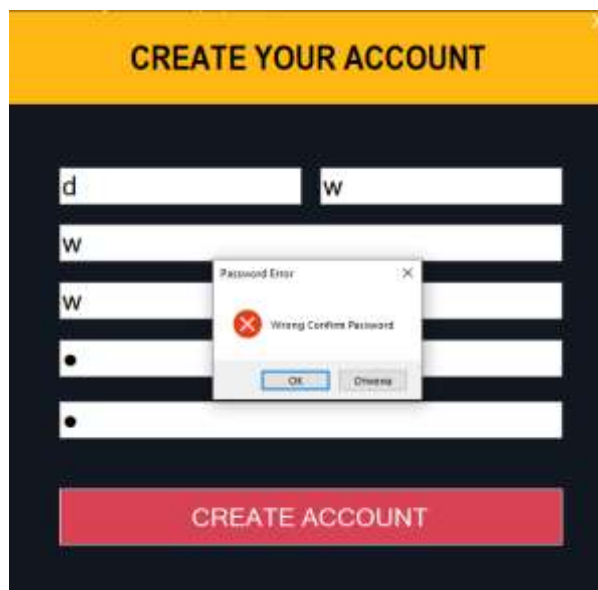


Рисунок 103 – Готовый MessageBox с ошибкой

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

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

Рисунок 104 - Доработка MessageBox

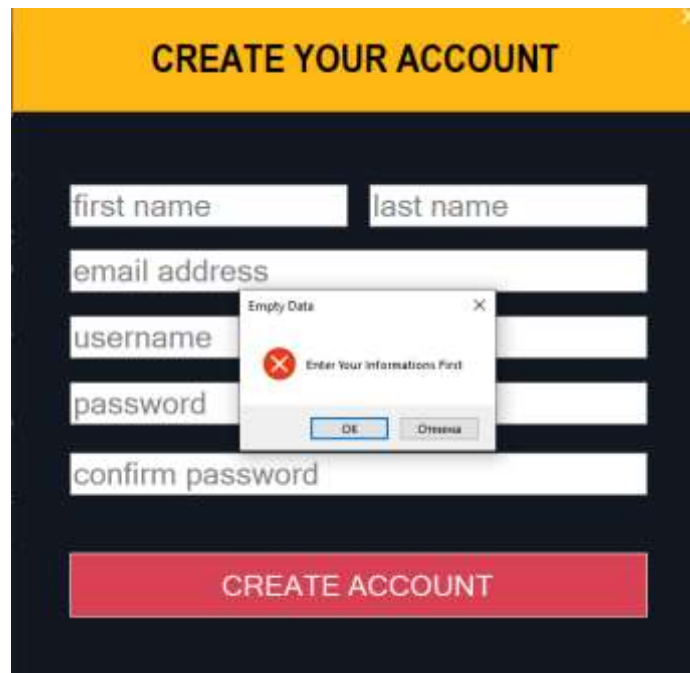


Рисунок 105 - Готовый MessageBox с ошибкой

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

```

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

    DataTable table = new DataTable();

    MySqlDataAdapter adapter = new MySqlDataAdapter();

    MySqlCommand command = new MySqlCommand("SELECT * FROM `users` WHERE `username` = @usn and `password` = @pass", db.getConnection());

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

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

    if(table.Rows.Count > 0)
    {
        MessageBox.Show("YES");
    }
    else
    {
        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.

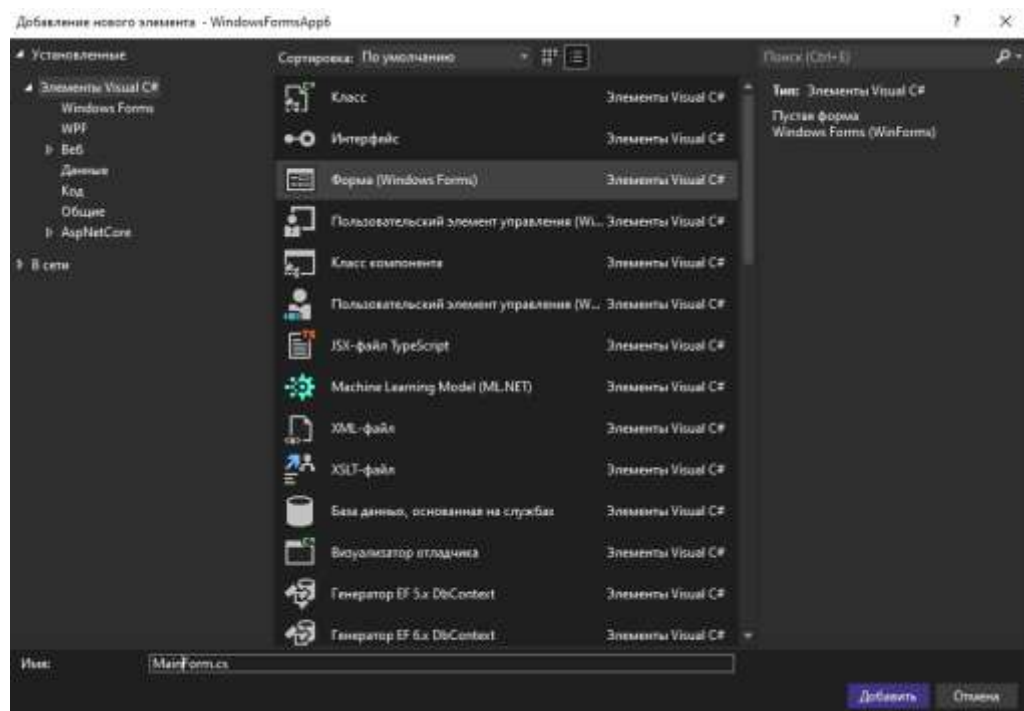


Рисунок 107 - Создание новой формы

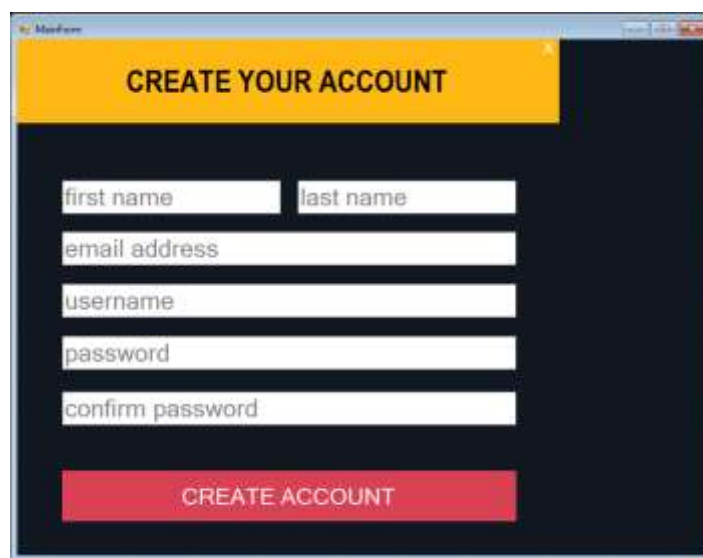


Рисунок 108 - Новая форма

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

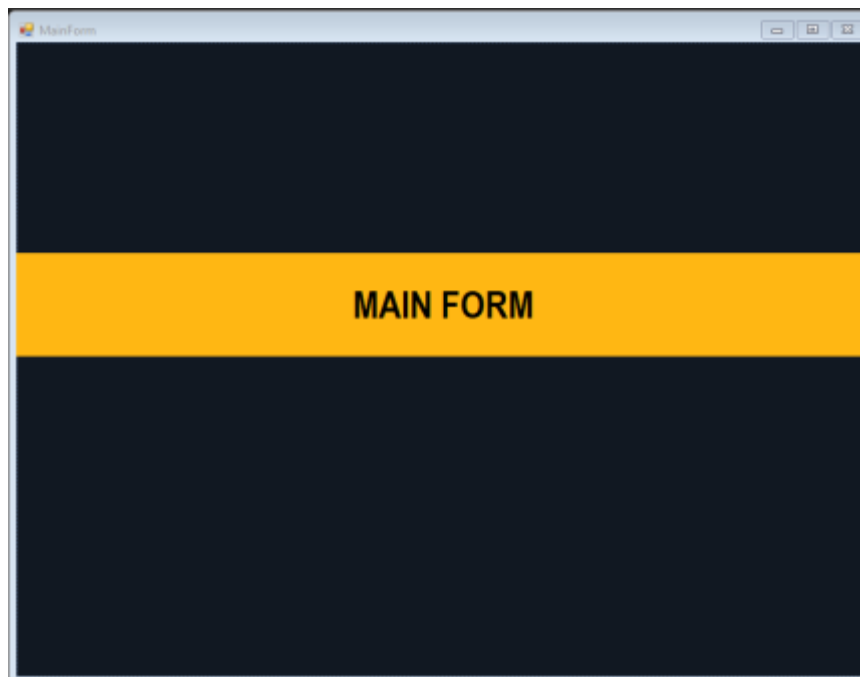


Рисунок 109 - Отредактированная форма

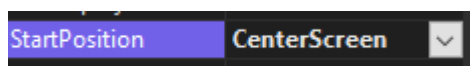


Рисунок 110 - Свойство StartPosition

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

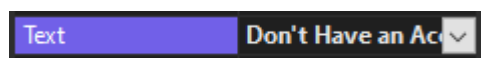


Рисунок 111 - Текст label

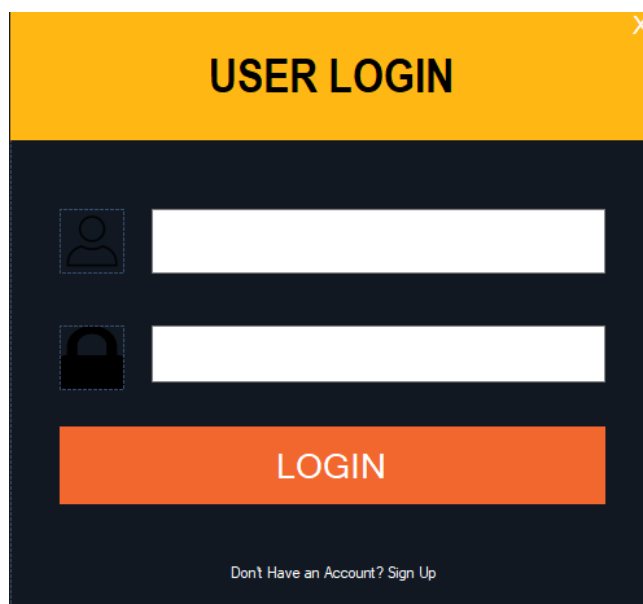


Рисунок 112 - Отредактированная LoginForm

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.

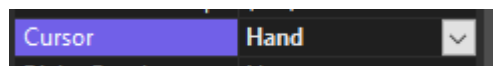


Рисунок 116 - Свойство Cursor

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»

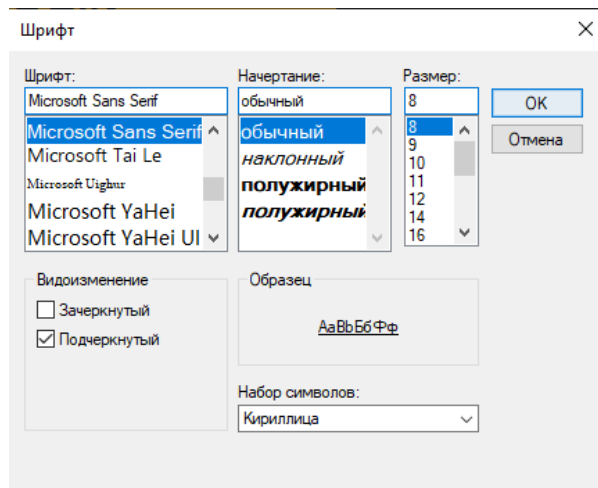


Рисунок 118 - Настройки шрифта

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

Рисунок 119 - RegisterForm

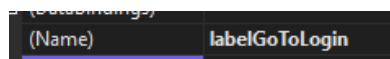


Рисунок 120 - Название label

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();
}
else

```

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

Prilozhenie 1.2. Validatsiya koda

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.