Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»
Филиал
«Минский радиотехнический колледж»

Учебный предмет
«Конструирование программ и языки программирования»

Инструкция

по выполнению лабораторной работы №33
«Разработка, отладка и испытание многопользовательского приложения»

Минск 2024 г.

Лабораторная работа № 33

Тема работы: «Разработка, отладка и испытание многопользовательского приложения»

1 Цель работы

Сформировать умения и навыки разработки, отладки и испытание многопользовательского приложения

2 Задание

Реализовать систему авторизации к приложению из лабораторной работы 29 на основе теоретических сведений.

3 Оснащение работы

ПК, среда Visual Studio 2019, MSword.

4 Основные теоретические сведения

Авторизация помогает избежать конфликтов с БД и приложением при использовании программного средства сразу несколькими пользователями.

Для авторизации можно использовать два подхода.

Первый подход заключается в обращении к БД, на сервере которой зарегистрированы пользователи со своими ролями, правами доступа и т.д.

Второй подход более простой – предполагает наличие в БД несвязанной таблицы с пользовательскими данными (логин и пароль), по которым и осуществляется вход или разблокировка функций приложения.

Для разработки реализации многопользовательского приложения для нашего примера воспользуемся вторым вариантом.

Первое, что нужно сделать, добавить в БД новую таблицу Users по следующему скрипту:

```
CREATE TABLE Users (
  id INT IDENTITY PRIMARY KEY,
  username VARCHAR(50) NOT NULL,
  password VARCHAR(50) NOT NULL
);
```

Добавлять таблицу будем посредством SQL-запроса на сервере (рисунки 1-2).

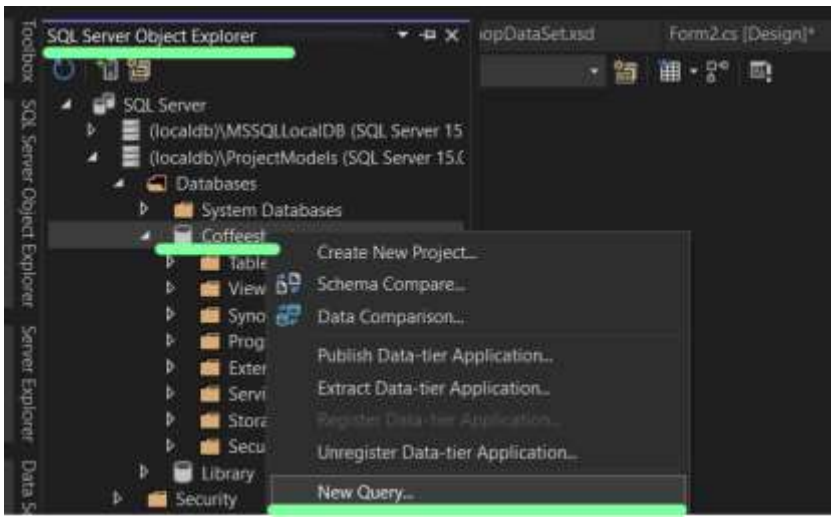

Рисунок 1 – Создание SQL-запроса

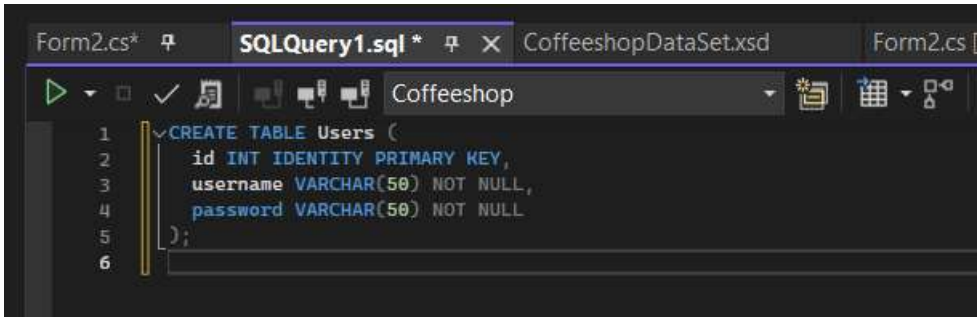

Рисунок 2 – Код SQL-запроса для создания таблицы в БД

Если посмотреть в Обозревателе серверов, после обновления увидим, что таблица успешно добавилась (рисунок 3).

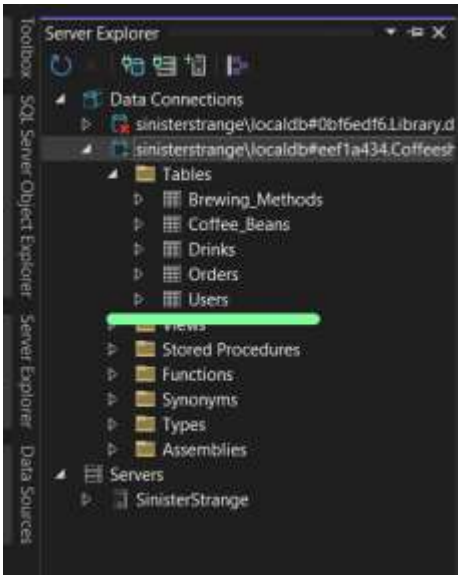

Рисунок 3 – Добавление таблицы

Теперь необходимо заполнить таблицу авторизационными данными для нескольких пользователей.

Для этого выполним следующий запрос:

Insert into Users (username, password) values ('Sinister','2281337'), ('AizenSoloverse','2905');

Т.к. для данной таблицы значения id генерируется автоматически, при вставке данных нам необходимо перечислять наименование полей, в которые нам необходимо вставлять информацию. Кроме этого, данный запрос является примером множественного Insert.

После выполнения запроса можно посмотреть, что в нашей таблице присутствуют два пользователя (рисунок 4).

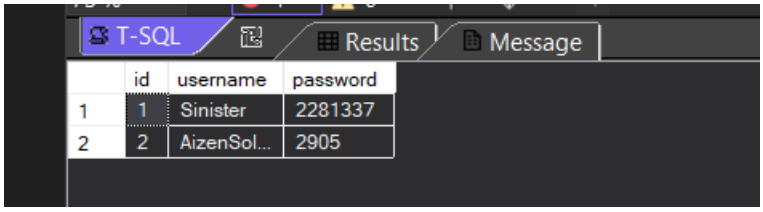

Рисунок 4 – Данные в таблице Users после запроса на вставку строк

Далее обновляем источник данных в Обозревателе серверов.

Затем на форму добавляем две кнопки – Авторизация и Регистрация (рисунок 5).

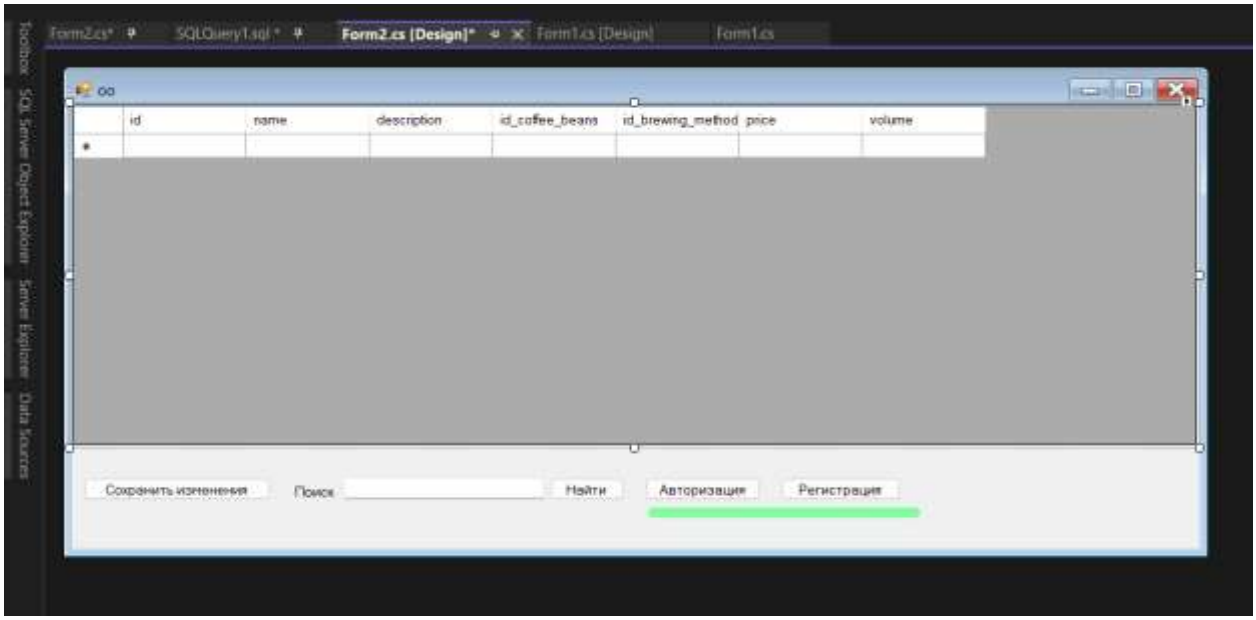

Рисунок 5 – Добавление кнопок для работы с таблицей Users

Для прохождения авторизации понадобится новая форма с полями для ввода данных.

Для добавления новой формы в проект в Обозревателе решения (Solution Explorer) добавим новую форму (рисунки 6-7).

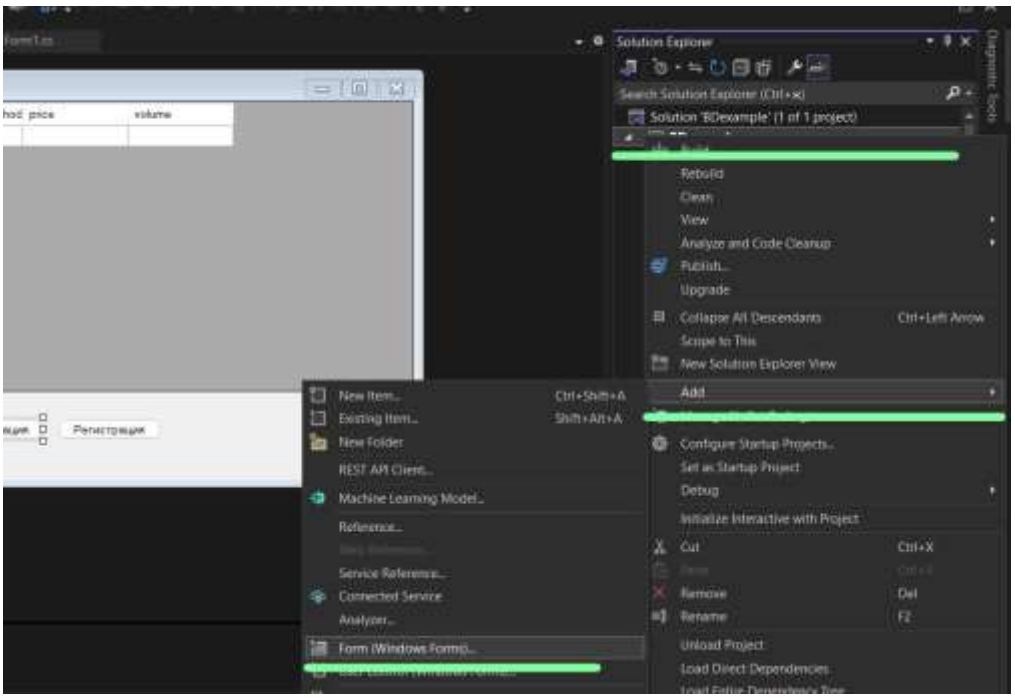

Рисунок 6 – Добавление новой формы к проекту

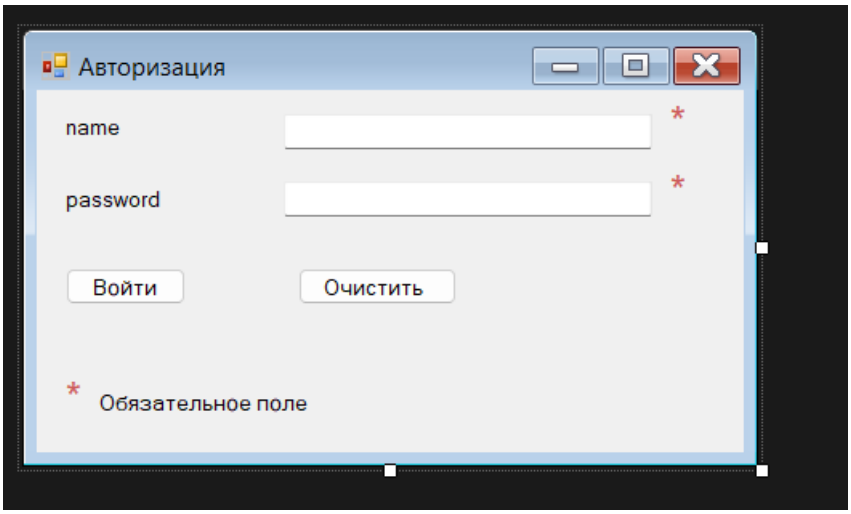

Рисунок 7 – Форма авторизации

Также добавим новую форму и для регистрации пользователя (рисунок 8).

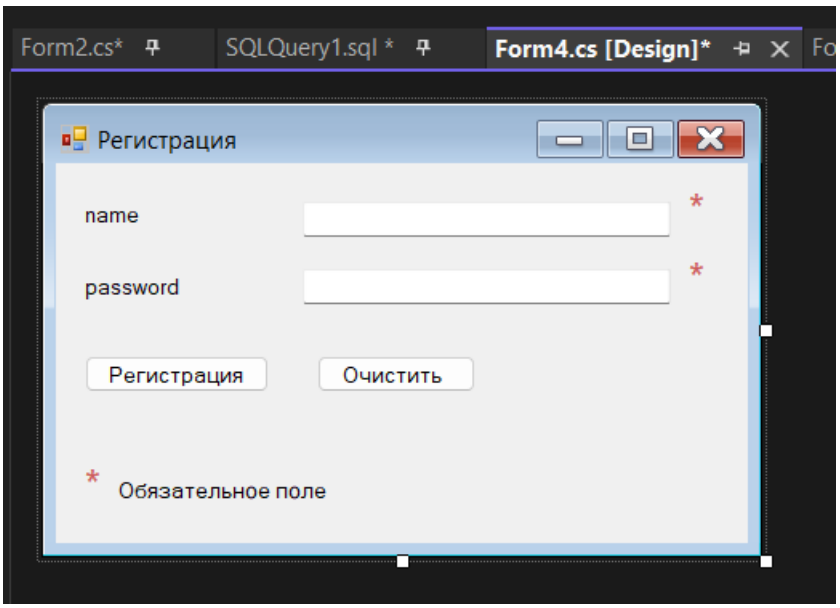

Рисунок 8 – Форма регистрации

Для обеих форм необходимо внести ряд ограничений.

Первое – сокрытие пароля. Для этого можно использовать одно из свойств элемента textbox – UseSystemPasswordChar (рисунок 9). Для данного свойства необходимо установить значение True.

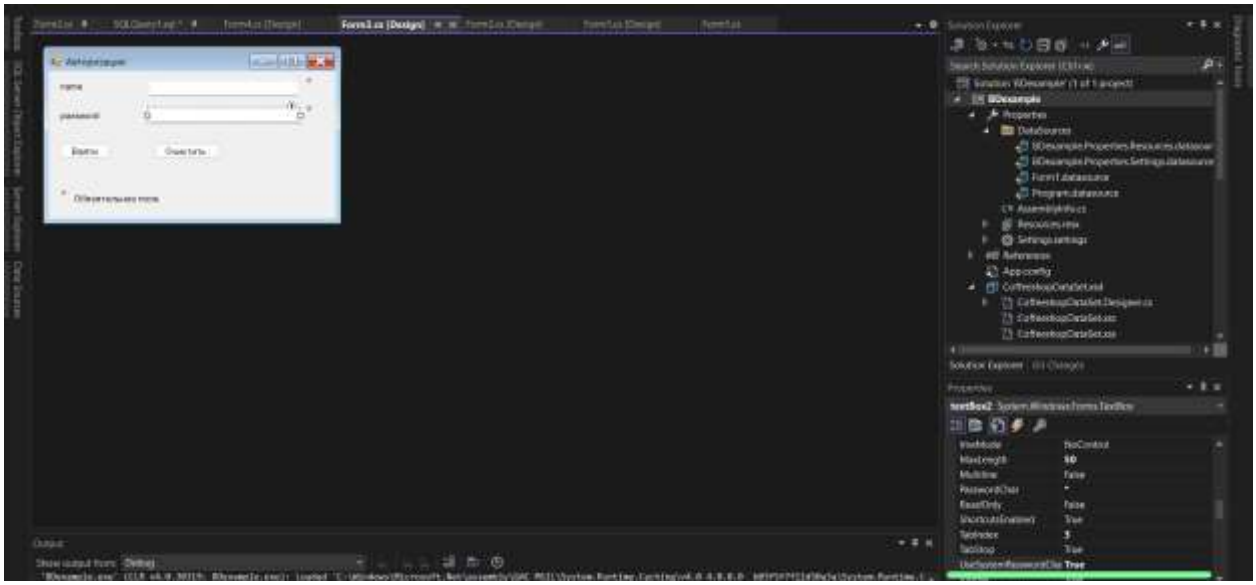

Рисунок 9 – Сокрытие пароля при вводе

Второе – пометка обязательности поля для ввода. За это отвечают компоненты label.

Третье – установка максимального значения количества символов для логина и пароля. В таблице с пользователями у обоих полей стоит ограничение в 50 символов. Соответственно, при вводе значения, превышающего 50 символов, возникнет исключение.

Чтобы выставить максимально допустимое значение для поля, можно воспользоваться свойством textBox – MaxLength (рисунок 10).

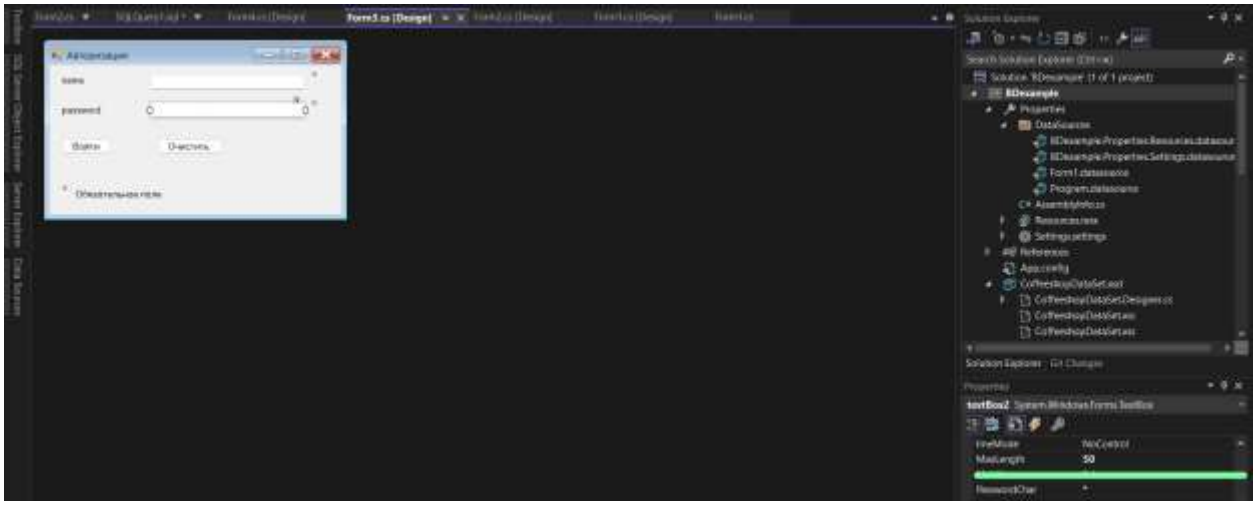

Рисунок 10 – Установка максимальной длины поля

Четвертое – написать сообщения пользователю в случае ошибок и обработать исключения.

Пароль и имя пользователя в базе данных – символьные. Поэтому особых ограничений на ввод данных нет, кроме того, что оба поля обязательны для заполнения (стоит Not null).

Для обработки данных пользователя и проведение аутентификации, в первую очередь заблокируем возможность сохранения изменений и поиска для неавторизированных пользователей на основной форме приложения. Для этого можно воспользоваться свойством кнопок Enabled и установить значение False (рисунок 11).

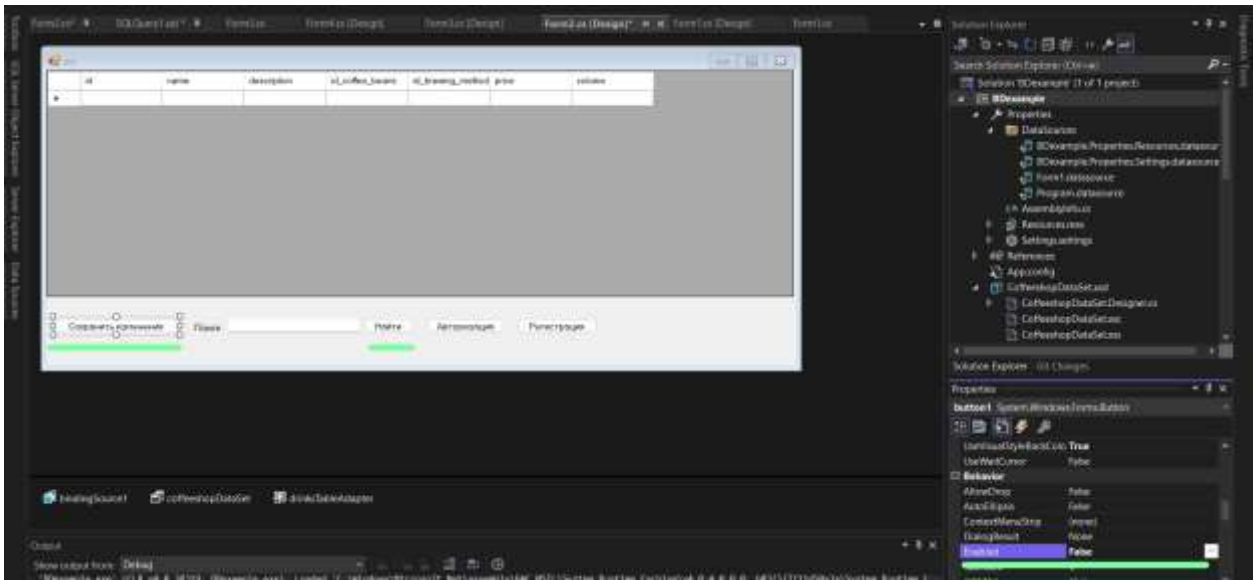

Рисунок 11 – Установка неактивности кнопок

После этого можно начинать писать код для обработки авторизации и взаимодействия между формами. Полный код основной формы представлен ниже.

```
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 System.Data.SqlClient;

namespace BDexample
{

   public partial class Form2 : Form
   {
      public Form2()
      {
         InitializeComponent();
      }

      private void Form2_Load(object sender, EventArgs e)
      {
         // TODO: This line of code loads data into the 'coffeeshopDataSet.Drinks' table. You can move, or remove it, as needed.
         this.drinksTableAdapter.Fill(this.coffeeshopDataSet.Drinks);

      }
      // функции для доступа к кнопкам Сохранить и Найти с формы авторизации.
      public Button Getsavebutton()
      {
         return button1;
      }
      public Button Getsearchbutton()
      {
         return button2;
      }
      private void button1_Click(object sender, EventArgs e)
      {
         try
         {
```

```
                // Сохранение изменений в БД
                this.drinksTableAdapter.Update(coffeeshopDataSet.Drinks);

                // Обновление данных в DataGridView, чтобы отобразить изменения, если они были в БД
                coffeeshopDataSet.Drinks.Clear();
                this.drinksTableAdapter.Fill(coffeeshopDataSet.Drinks);

                MessageBox.Show("Изменения сохранены успешно.", "Сохранение", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Произошла ошибка при сохранении изменений: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private void button2_Click(object sender, EventArgs e)
        {
            string searchQuery = textBox1.Text.Trim();

            // Очистим DataGridView перед новым поиском
            coffeeshopDataSet.Clear();

            // Используем SQL-запрос для поиска в таблице Drinks
            string sqlQuery = "SELECT * FROM Drinks WHERE name LIKE @search OR description LIKE @search";

            using (SqlConnection connection = new SqlConnection("Data Source=(localdb)\\ProjectModels;Initial Catalog=Coffeeshop;Integrated Security=True"))
            {
                connection.Open();

                using (SqlDataAdapter adapter = new SqlDataAdapter(sqlQuery, connection))
                {
                    adapter.SelectCommand.Parameters.AddWithValue("@search", "%" + searchQuery + "%");

                    adapter.Fill(coffeeshopDataSet, "Drinks");
                }
            }
```

```
            // Обновим DataGridView
            dataGridView1.DataSource = coffeeshopDataSet.Tables["Drinks"];
        }

        private void textBox1_TextChanged(object sender, EventArgs e)
        {
            if (string.IsNullOrWhiteSpace(textBox1.Text))
            {
                // Очистка TextBox, поэтому отменяем результат поиска и отображаем все данные
                coffeeshopDataSet.Clear();
                using (SqlConnection connection = new SqlConnection("Data Source=(localdb)\\ProjectModels;Initial Catalog=Coffeeshop;Integrated Security=True"))
                {
                    connection.Open();

                    // Загружаем все данные из таблицы Drinks
                    string sqlQuery = "SELECT * FROM Drinks";
                    using (SqlDataAdapter adapter = new SqlDataAdapter(sqlQuery, connection))
                    {
                        adapter.Fill(coffeeshopDataSet, "Drinks");
                    }
                }

                // Обновляем DataGridView
                dataGridView1.DataSource = coffeeshopDataSet.Tables["Drinks"];
            }
            else
            {
                // TextBox не пуст, выполняем поиск
                button2_Click(sender, e);
            }
        }
        //обеспечиваем открытие формы авторизации
        private void button3_Click(object sender, EventArgs e)
        {
            Form3 form3 = new Form3(this); // Передаем ссылку на Form2
            form3.Show();
        }
    }
}
```

В форме с авторизацией код будет выглядеть следующим образом:

```
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 System.Data.SqlClient;
namespace BDexample
{
   public partial class Form3 : Form
   {
      private Form2 form2; // Добавляем поле для хранения ссылки на Form2
      public class UserManager
      {
         private string connectionString = "Data Source=(localdb)\\ProjectModels;Initial Catalog=Coffeeshop;Integrated Security=True";

         public bool AuthenticateUser(string username, string password)
         {
            //SQL-запрос для проверки учетных данных
            string sqlQuery = "SELECT COUNT(*) FROM Users WHERE username = @username AND password = @password";

            using (SqlConnection connection = new SqlConnection(connectionString))
            {
               connection.Open();

               using (SqlCommand command = new SqlCommand(sqlQuery, connection))
               {
                  // Параметризация запроса
                  command.Parameters.AddWithValue("@username", username);
                  command.Parameters.AddWithValue("@password", password);

                  // Проверка существования пользователя с заданными учетными данными
```

```
                    int userCount = (int)command.ExecuteScalar();

                    // Если userCount больше нуля, значит, пользователь существует и пароль верен
                    return userCount > 0;
                }
            }
        }
    }
    public Form3(Form2 form2)
    {
        InitializeComponent();
        this.form2 = form2; // Сохраняем ссылку на Form2
        this.Load += Form3_Load; // Подписываемся на событие Load
    }
    //авторизация
    private void button1_Click(object sender, EventArgs e)
    {
        string login = textBox1.Text; // поле для ввода логина
        string password = textBox2.Text; // поле для ввода пароля
        UserManager userManager = new UserManager(); // создаем объект класса для проведения дальнейшей авторизации
       bool res= userManager.AuthenticateUser(login, password);
        if (res==true)  // если пользователь есть в БД
        {
            MessageBox.Show($"Добрый день, {login} ");
            Button savebtn = form2.Getsavebutton();
            Button searchbutton = form2.Getsearchbutton();
            //делаем кнопки Найти и Сохранить с основной формы активными
            savebtn.Enabled = true;
            searchbutton.Enabled = true;
            this.Close();

        }
        else // если пользователя нет в БД
        {
            MessageBox.Show("Введены неверные данные. Попробуйте еще раз");
            button2_Click(sender, e);

        }
    }

    private void Form3_Load(object sender, EventArgs e)
```

```
            {
                textBox1.Focus(); // Устанавливаем фокус на TextBox при открытии формы авторизации

            }

            private void button2_Click(object sender, EventArgs e)
            {

                textBox1.Clear();
                textBox2.Clear();
                textBox1.Focus();
            }
        }
    }
```

Далее разработаем проведение регистрации пользователя.

В регистрации пользователя существует один момент, на который стоит обратить внимание и обработать. Это ситуация связана с созданием нового пользователя с данными, которые уже есть в таблице БД.

Код для третьей формы будет выглядеть следующим образом:

```
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 System.Data.SqlClient;
namespace BDexample
{
    public partial class Form4 : Form
    {
        public class UserManager
        {
            private string connectionString = "Data Source=(localdb)\\ProjectModels;Initial Catalog=Coffeeshop;Integrated Security=True";

            public bool RegisterUser(string username, string password)
            {
                // Проверяем, существует ли пользователь с таким именем
                if (IsUserExists(username))
                {
```

```
                // Пользователь с таким именем уже существует
                return false;
            }

            // Если пользователя с таким именем нет, регистрируем нового пользователя
            string sqlQuery = "INSERT INTO Users (username, password) VALUES (@username, @password)";

            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                connection.Open();

                using (SqlCommand command = new SqlCommand(sqlQuery, connection))
                {
                    // Параметризация запроса
                    command.Parameters.AddWithValue("@username", username);
                    command.Parameters.AddWithValue("@password", password);

                    // Выполняем запрос
                    int rowsAffected = command.ExecuteNonQuery();

                    // Если добавление прошло успешно (была добавлена хотя бы одна строка), возвращаем true
                    return rowsAffected > 0;
                }
            }
        }

        private bool IsUserExists(string username)
        {
            // Проверяем, существует ли пользователь с таким именем
            string sqlQuery = "SELECT COUNT(*) FROM Users WHERE username = @username";

            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                connection.Open();
```

```
                using (SqlCommand command = new SqlCommand(sqlQuery, connection))
                {
                    // Параметризация запроса
                    command.Parameters.AddWithValue("@username", username);

                    // Получаем количество пользователей с заданным именем
                    int userCount = (int)command.ExecuteScalar();

                    // Если userCount больше нуля, значит, пользователь существует
                    return userCount > 0;
                }
            }
        }
    }
    public Form4()
    {
        InitializeComponent();
    }
    // автофокус на поле
    private void Form4_Load(object sender, EventArgs e)
    {
        textBox1.Focus(); // Устанавливаем фокус на TextBox при открытии формы регистрации
    }
    // очистка полей
    private void button2_Click(object sender, EventArgs e)
    {
        textBox1.Clear();
        textBox2.Clear();
        textBox1.Focus();
    }
    // регистрация
    private void button1_Click(object sender, EventArgs e)
    {
        string login = textBox1.Text;
        string password = textBox2.Text;
        UserManager userManager = new UserManager();
        bool res = userManager.RegisterUser(login, password);
        if (res==true) {
            MessageBox.Show("Вы успешно зарегистрированы. Пожалуйста, пройдите авторизацию");
            this.Close();
```

```
                }
                else
                {
                    MessageBox.Show("Пользователь с таким именем уже зарегистрирован. Введите другое имя пользователя или пройдите авторизацию");
                    button2_Click(sender, e);
                }

            }
        }
    }
```

5. Порядок выполнения работы

1. Выделить ключевые моменты задачи.
2. Построить алгоритм и теоретическую объектную модель решения задачи.
3. Запрограммировать полученные алгоритмы и объектную модель.

6. Форма отчета о работе

Лабораторная работа № ___

Номер учебной группы _____

Фамилия, инициалы учащегося _____

Дата выполнения работы _____

Тема работы: _____

Цель работы: _____

Оснащение работы: _____

Результат выполнения работы: _____

7. Контрольные вопросы и задания

1. Какое свойство отвечает за неактивность элемента?
2. Приведите пример реализации запроса на множественный Insert.
3. Что означает запись not null при создании таблицы БД?
4. Как называются строки данного вида: Data Source=(localdb)\\ProjectModels;Initial Catalog=Coffeeshop;Integrated Security=True"?

8. Рекомендуемая литература

1. Рихтер, Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C# / Дж. Рихтер. СПб. : Изд-во Питер, 2021. 896 с.

2. Прайс, М. Дж. C# 10 и .NET 6. Современная кросс-платформенная разработка / М. Дж. Прайс. СПб : Изд-во Питер, 2023. 848 с.

3. Васильев, А.Н. Программирование на C# для начинающих. Особенности языка / А.Н. Васильев. М. : Эксмо, 2022. 528 с.

4. Фримен, А. ASP.NET Core 3 с примерами на C# для профессионалов / А. Фримен. СПб. : Изд-во Вильямс, 2021. 1184 с.