МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Криворізький національний університет

Кафедра моделювання та програмного забезпечення

Звіт

з лабораторної роботи №5

РОЗРОБКА ДОДАТКІВ ОБ’ЄКТНОГО УПРАВЛІННЯ БАЗАМИ ДАНИХ

Студент групи ІПЗ-21-2

Губарєв Р.В.

+380980190289

Викладачі

Козиков А. В.

Гриценко А. М.

Кривий Ріг

2022

1. **Короткі теоретичні відомості про СУБД та їх реалізацію за допомогою IDE**

СУБД (Система управління базами даних) є програмними засобами, які дозволяють зберігати, керувати та маніпулювати даними в базах даних. Вони забезпечують структуроване зберігання інформації, а також надають можливості для ефективного пошуку, оновлення та аналізу даних.

Інтегроване середовище розробки (IDE) є програмним інструментарієм, який надає розробникам зручне середовище для написання, відладки і тестування програмного коду. IDE може також містити інструменти для реалізації СУБД і розробки баз даних.

СУБД можуть бути реалізовані за допомогою IDE, які надають засоби для створення та управління базами даних. Такі IDE можуть мати графічний інтерфейс користувача, що спрощує створення таблиць, введення даних, виконання запитів і аналізу результатів.

За допомогою IDE розробники можуть створювати схеми баз даних, визначати таблиці, поля та зв'язки між ними. Вони також можуть створювати запити для отримання, оновлення або видалення даних. IDE надають засоби для відладки SQL-запитів, а також можливості для виконання аналітичних запитів та створення звітів.

Деякі популярні IDE для реалізації СУБД включають:

1. MySQL Workbench: Інтегроване середовище розробки для MySQL, яке надає інструменти для моделювання баз даних, створення SQL-запитів і адміністрування сервера баз даних MySQL.
2. Oracle SQL Developer: Інструмент для розробки та адміністрування баз даних Oracle. Він надає можливості для створення та виконання SQL-запитів, моделювання баз даних і управління об'єктами бази даних.
3. Microsoft SQL Server Management Studio (SSMS): IDE для роботи з базами даних Microsoft SQL Server. Він дозволяє створювати та змінювати об'єкти бази даних, виконувати запити і адмініструвати сервер.
4. PostgreSQL Studio: Інструмент для розробки та управління базами даних PostgreSQL. Він надає можливості для створення таблиць, індексів, виконання запитів і адміністрування бази даних.
5. **Короткі теоретичні відомості про візуальні об’єкти (компонентів), що використовуються.**

**MenuStrip -** Створює меню, що настроюється.

**DataGridView -** Елемент керування DataGridView надає таблицю, що настроюється, для відображення даних. Клас DataGridView дозволяє налаштовувати комірки, рядки, стовпці та межі.

**Button -** Запускає, зупиняє чи перериває процес.

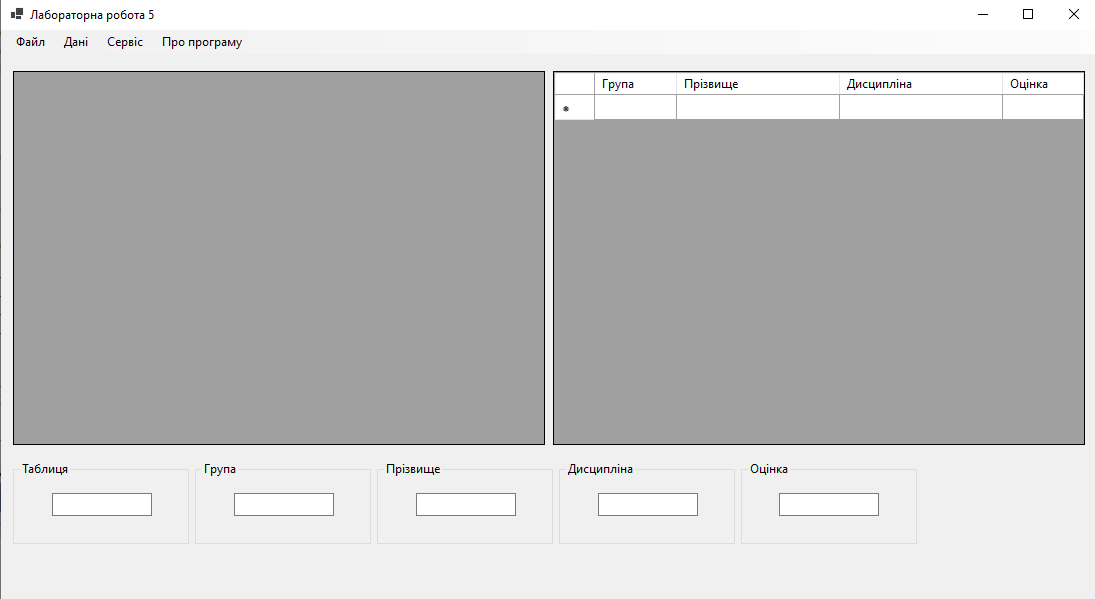
**Panel -** Групують набір елементів керування на кадрі без міток з можливістю прокручування.

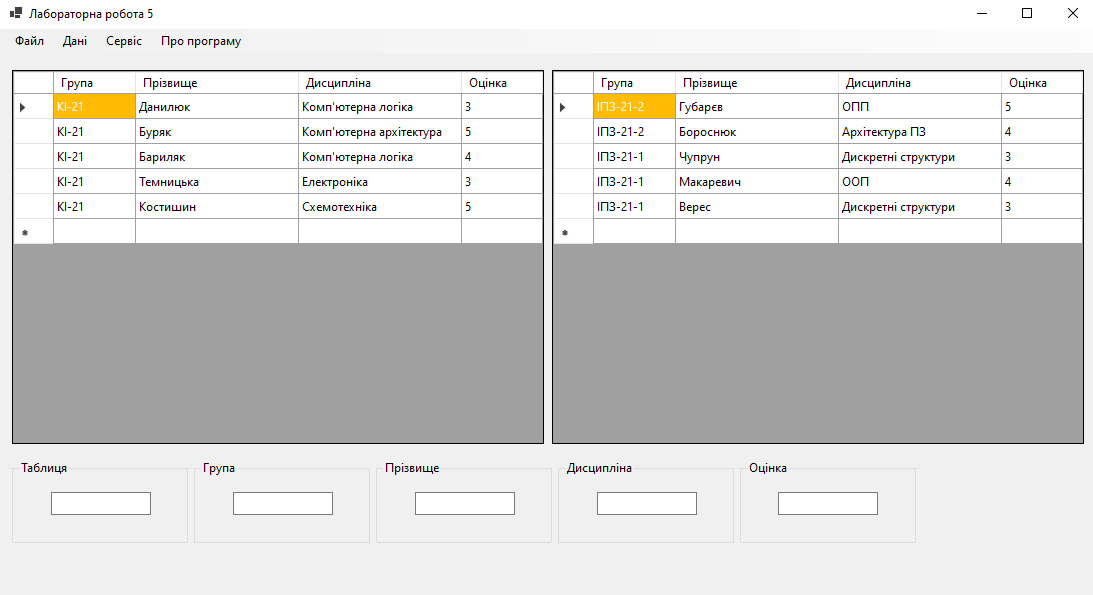
**GroupBox -** Групує набір елементів керування (наприклад, перемикачів) на кадрі з позначкою та без підтримки.

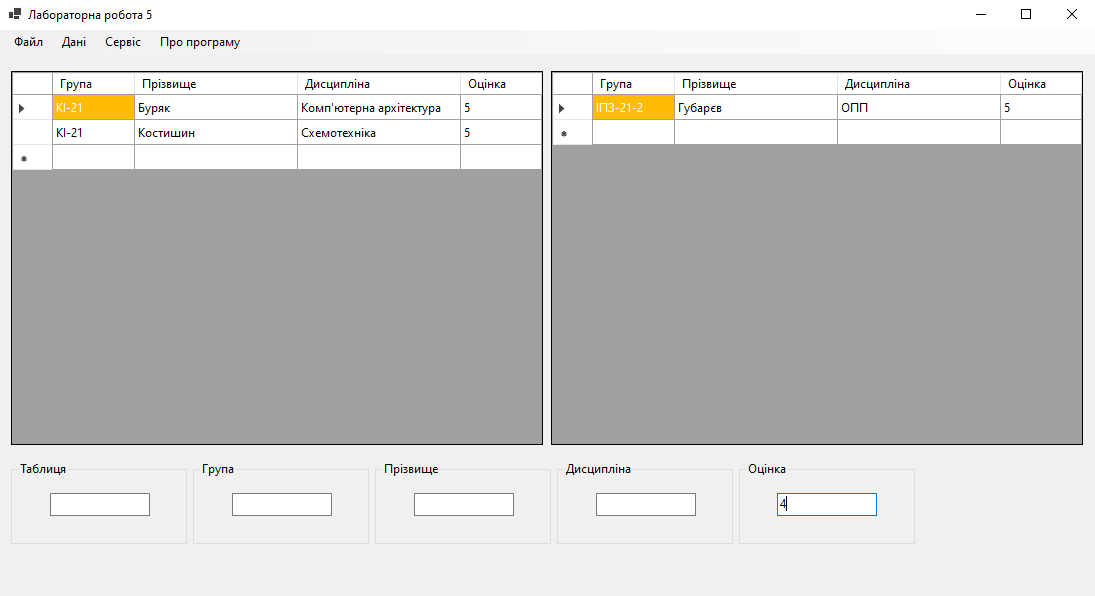
1. **Вихідний текст програми**

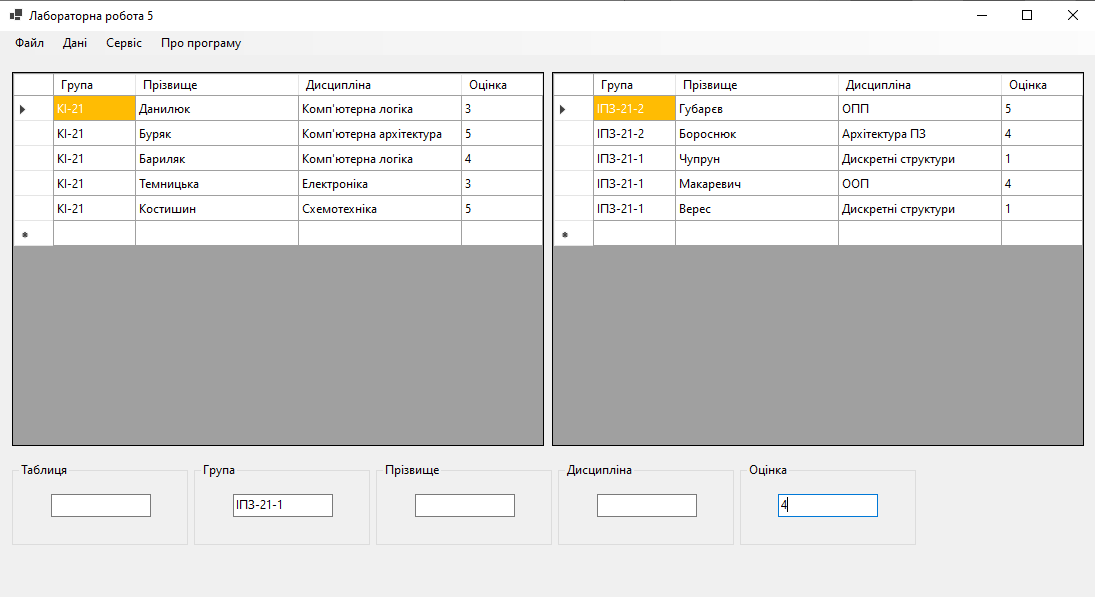
|  |
| --- |
| **Form1.cs**  using System.Collections.Generic;  using System.Data;  using System.Data.OleDb;  using System.Windows.Forms;  namespace Lab5  {  public partial class Form1 : Form  {  public string filePath;  OleDbConnection dbConnection;  List<string[]> dataRows = new List<string[]>();  List<string[]> dataRows2 = new List<string[]>();  public Form1()  {  InitializeComponent();  }  public void встановитиШляхДоБазиДанихToolStripMenuItem\_Click(object sender, EventArgs e)  {  filePath = "Сесія.accdb";  }  private void підключитиБазуДанихToolStripMenuItem\_Click(object sender, EventArgs e)  {  // Створюємо з'єднання  string connectionString = $"Provider=Microsoft.ACE.OLEDB.12.0;Data Source={filePath}"; // Строка з'єднання  dbConnection = new OleDbConnection(connectionString); // Створюємо з'єднання  }  private void завантажитиДані1ToolStripMenuItem\_Click(object sender, EventArgs e)  {  dataGridView2.Rows.Clear();  dbConnection.Open();  string query = "SELECT \* FROM Сесія";  OleDbCommand dbCommand = new OleDbCommand(query, dbConnection);  OleDbDataReader dbReader = dbCommand.ExecuteReader();  if (dbReader.HasRows == false)  {  MessageBox.Show("Даних не знайдено!", "Помилка!");  }  else  {  while (dbReader.Read())  {  dataGridView2.Rows.Add(dbReader["Група"], dbReader["Прізвище"], dbReader["Дисципліна"], dbReader["Оцінка"]);  }  }  dbReader.Close();  dbConnection.Close();  // Заповнення резервного списку даних  dataRows2 = DataBackupList(dataGridView2, dataRows2);  }  private void завантажитиДані2ToolStripMenuItem\_Click(object sender, EventArgs e)  {  dataGridView1.Rows.Clear();  dataGridView1.Columns.Clear();  // Створення колонок у DataGridView1  dataGridView1.Columns.Add("Column1", "Група");  dataGridView1.Columns.Add("Column2", "Прізвище");  dataGridView1.Columns.Add("Column3", "Дисципліна");  dataGridView1.Columns.Add("Column4", "Оцінка");  foreach (DataGridViewColumn column in dataGridView1.Columns)  {  column.AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;  }  dataGridView1.Columns[0].FillWeight = 20;  dataGridView1.Columns[1].FillWeight = 40;  dataGridView1.Columns[2].FillWeight = 40;  dataGridView1.Columns[3].FillWeight = 20;  // Додавання рядків у DataGridView1  dataGridView1.Rows.Add("КІ-21", "Данилюк", "Комп'ютерна логіка", "3");  dataGridView1.Rows.Add("КІ-21", "Буряк", "Комп'ютерна архітектура", "5");  dataGridView1.Rows.Add("КІ-21", "Бариляк", "Комп'ютерна логіка", "4");  dataGridView1.Rows.Add("КІ-21", "Темницька", "Електроніка", "3");  dataGridView1.Rows.Add("КІ-21", "Костишин", "Схемотехніка", "5");  // Заповнення резервного списку даних  dataRows = DataBackupList(dataGridView1, dataRows);  }  private List<string[]> DataBackupList(DataGridView dataGridView, List<string[]> dRows)  {  foreach (DataGridViewRow row in dataGridView.Rows)  {  string[] rowData = new string[dataGridView.Columns.Count];  for (int i = 0; i < dataGridView.Columns.Count; i++)  {  // Перевіряємо, чи комірка не є null  if (row.Cells[i].Value != null)  {  rowData[i] = row.Cells[i].Value.ToString();  }  else  {  rowData[i] = string.Empty; // або інше значення за замовчуванням  }  }  dRows.Add(rowData);  }  return dRows;  }  private void textBox1\_TextChanged(object sender, EventArgs e)  {  }  private void застосуватиЗапитНаВибіркуToolStripMenuItem\_Click(object sender, EventArgs e)  {  // Отримуємо запит на вибірку з TextBox  string filter = textBox1.Text;  // Очищення старих даних у DataGridView2  dataGridView2.Rows.Clear();  dataGridView1.Rows.Clear();  foreach (string[] rowData in dataRows2)  {  if (rowData[0] != "")  {  // Доступ до значень комірок рядка  string group = rowData[0];  string surname = rowData[1];  string discipline = rowData[2];  string grade = rowData[3];  if (Int32.Parse(grade) > Int32.Parse(filter))  {  dataGridView2.Rows.Add(group, surname, discipline, grade);  }  }  else  break;  }  dataRows2 = DataBackupList(dataGridView2, dataRows2);  foreach (string[] rowData in dataRows)  {  if (rowData[0] != "")  {  // Доступ до значень комірок рядка  string group = rowData[0];  string surname = rowData[1];  string discipline = rowData[2];  string grade = rowData[3];  if (Int32.Parse(grade) > Int32.Parse(filter))  {  dataGridView1.Rows.Add(group, surname, discipline, grade);  }  }  else  break;  }  dataRows = DataBackupList(dataGridView1, dataRows);  }  private void застосуватиЗапитНаМодифікаціюToolStripMenuItem\_Click(object sender, EventArgs e)  {  string filter = textBox1.Text;  string group\_filter = textBox2.Text;  dataGridView2.Rows.Clear();  dataGridView1.Rows.Clear();  foreach (string[] rowData in dataRows2)  {  if (rowData[0] != "")  {  // Доступ до значень комірок рядка  string group = rowData[0];  string surname = rowData[1];  string discipline = rowData[2];  string grade = rowData[3];  if (Int32.Parse(grade) < Int32.Parse(filter) && group == group\_filter)  {  dataGridView2.Rows.Add(group, surname, discipline, '1');  }  else  {  dataGridView2.Rows.Add(group, surname, discipline, grade);  }  }  else  break;  }  dataRows2 = DataBackupList(dataGridView2, dataRows2);  foreach (string[] rowData in dataRows)  {  if (rowData[0] != "")  {  // Доступ до значень комірок рядка  string group = rowData[0];  string surname = rowData[1];  string discipline = rowData[2];  string grade = rowData[3];  if (Int32.Parse(grade) < Int32.Parse(filter) && group == group\_filter)  {  dataGridView1.Rows.Add(group, surname, discipline, '1');  }  else  {  dataGridView1.Rows.Add(group, surname, discipline, grade);  }  }  else  break;  }  dataRows = DataBackupList(dataGridView1, dataRows);  }  private void застосуватиЗапитНаВидаленняToolStripMenuItem\_Click(object sender, EventArgs e)  {  dataGridView2.Rows.Clear();  dataGridView1.Rows.Clear();  string filter = textBox1.Text;  foreach (string[] rowData in dataRows2)  {  if (rowData[0] != "")  {  // Доступ до значень комірок рядка  string group = rowData[0];  string surname = rowData[1];  string discipline = rowData[2];  string grade = rowData[3];  if (Int32.Parse(grade) != Int32.Parse(filter))  {  dataGridView2.Rows.Add(group, surname, discipline, grade);  }  else  {  continue;  }  }  else  break;  }  foreach (string[] rowData in dataRows)  {  if (rowData[0] != "")  {  // Доступ до значень комірок рядка  string group = rowData[0];  string surname = rowData[1];  string discipline = rowData[2];  string grade = rowData[3];  if (Int32.Parse(grade) != Int32.Parse(filter))  {  dataGridView1.Rows.Add(group, surname, discipline, grade);  }  else  {  continue;  }  }  else  break;  }  }  private void відєднатиБазуДанихToolStripMenuItem\_Click(object sender, EventArgs e)  {  dataGridView1.Rows.Clear();  dataGridView1.Columns.Clear();  dataGridView2.Rows.Clear();  dataGridView2.Columns.Clear();  dbConnection.Close();  }  private void завершитиРоботуToolStripMenuItem\_Click(object sender, EventArgs e)  {  Close();  }  private void groupBox1\_Enter(object sender, EventArgs e)  {  }  private void застосуватиЗапитНаДодаванняToolStripMenuItem\_Click(object sender, EventArgs e)  {  string group = textBox2.Text;  string surname = textBox3.Text;  string discipline = textBox4.Text;  string grade = textBox1.Text;  if (textBox5.Text == "1")  {  dataGridView1.Rows.Add(group, surname, discipline, grade);  dataRows = DataBackupList(dataGridView1, dataRows);  }  else if (textBox5.Text == "2")  {  dataGridView2.Rows.Add(group, surname, discipline, grade);  dataRows2 = DataBackupList(dataGridView2, dataRows2);  }  }  private void проПрограмуToolStripMenuItem\_Click(object sender, EventArgs e)  {  AboutBox1 aboutBox = new AboutBox1();  aboutBox.Show();  }  }  } |
| **AboutBox1.cs**  using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Drawing;  using System.Linq;  using System.Reflection;  using System.Threading.Tasks;  using System.Windows.Forms;  namespace Lab5  {  partial class AboutBox1 : Form  {  public AboutBox1()  {  InitializeComponent();  this.Text = "Про програму";  this.labelProductName.Text = "Лабораторна робота 5";  this.labelVersion.Text = String.Format("Version {0}", AssemblyVersion);  this.labelCopyright.Text = AssemblyCopyright;  this.labelCompanyName.Text = "Rostik Hubariev";  }  #region Assembly Attribute Accessors  public string AssemblyTitle  {  get  {  object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyTitleAttribute), false);  if (attributes.Length > 0)  {  AssemblyTitleAttribute titleAttribute = (AssemblyTitleAttribute)attributes[0];  if (titleAttribute.Title != "")  {  return titleAttribute.Title;  }  }  return System.IO.Path.GetFileNameWithoutExtension(Assembly.GetExecutingAssembly().CodeBase);  }  }  public string AssemblyVersion  {  get  {  return Assembly.GetExecutingAssembly().GetName().Version.ToString();  }  }  public string AssemblyDescription  {  get  {  object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyDescriptionAttribute), false);  if (attributes.Length == 0)  {  return "";  }  return ((AssemblyDescriptionAttribute)attributes[0]).Description;  }  }  public string AssemblyProduct  {  get  {  object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyProductAttribute), false);  if (attributes.Length == 0)  {  return "";  }  return ((AssemblyProductAttribute)attributes[0]).Product;  }  }  public string AssemblyCopyright  {  get  {  object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyCopyrightAttribute), false);  if (attributes.Length == 0)  {  return "";  }  return ((AssemblyCopyrightAttribute)attributes[0]).Copyright;  }  }  public string AssemblyCompany  {  get  {  object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyCompanyAttribute), false);  if (attributes.Length == 0)  {  return "";  }  return ((AssemblyCompanyAttribute)attributes[0]).Company;  }  }  #endregion  private void labelVersion\_Click(object sender, EventArgs e)  {  }  private void okButton\_Click(object sender, EventArgs e)  {  Close();  }  }  } |

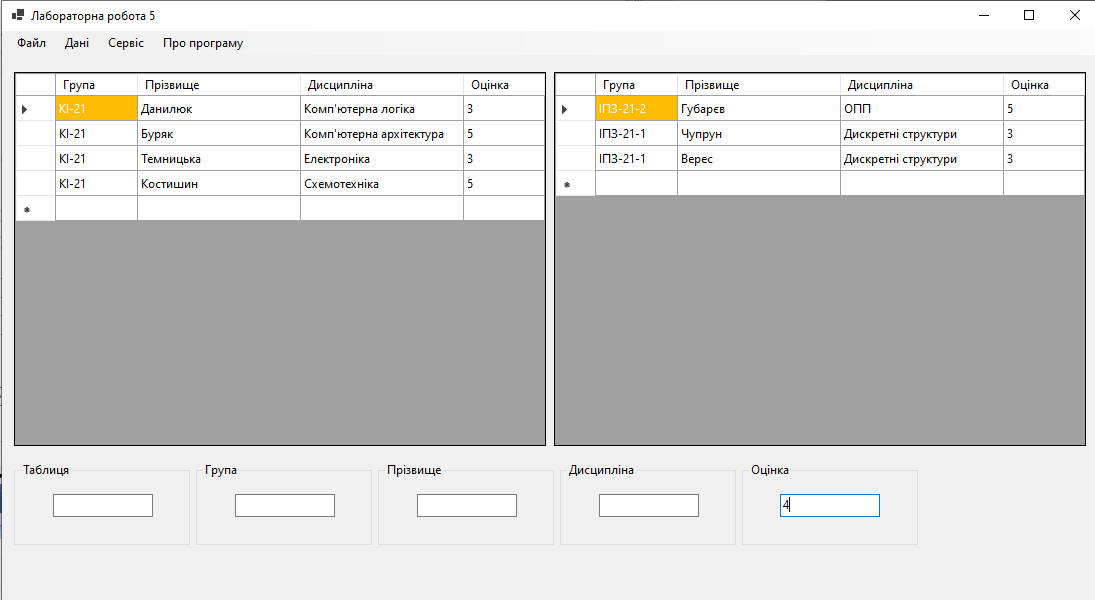
1. **Знімок інтерфейсу програми з результатами роботи**

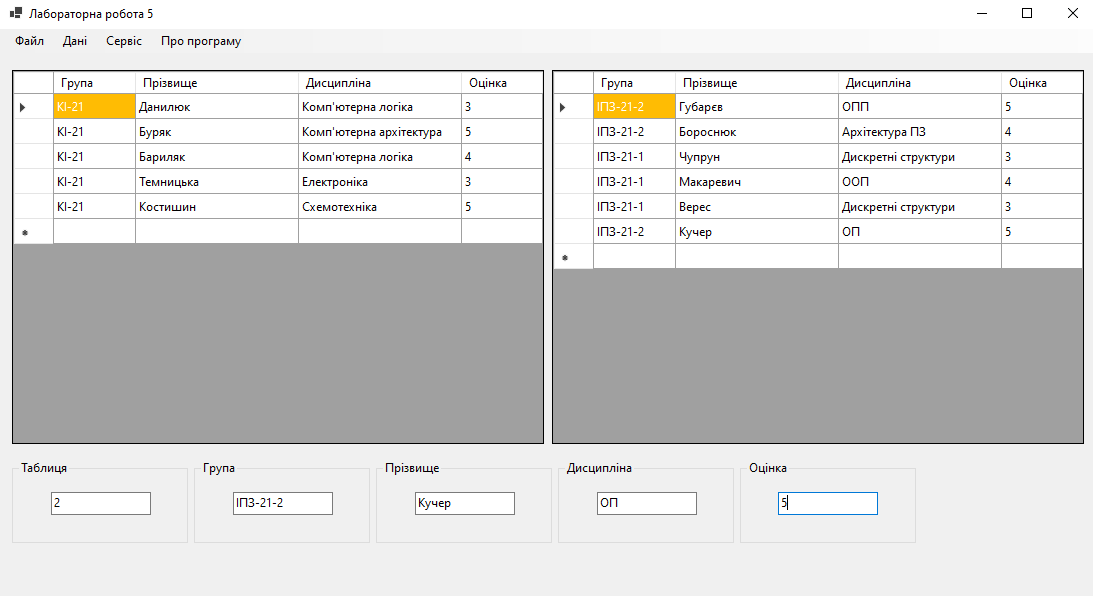
****

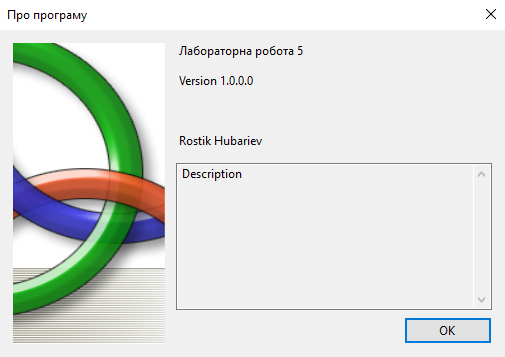
****

****

****

****

****

****

1. **Короткі висновки**

В цій лабораторній роботі я навчився підключати базу даних до WinForms та працювати з нею.

1. **Список літератури**

* [**https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/controls/windows-forms-controls-by-function?view=netframeworkdesktop-4.8**](https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/controls/windows-forms-controls-by-function?view=netframeworkdesktop-4.8)