**Виконав: Мельник Б. В.; КН-922в**

**Лабораторна робота №14**

**Тема:** Використання засобів обробки баз даних (БД) у застосунках Windows Forms.

**Мета роботи:** набути навички у створенні застосунків із використанням можливостей обробки БД на прикладі СУБД .MySQL Server.

**Індивідуальні завдання**

**Варіант №12**

Для виконання подальших завдань буде використовуватися предметна область згідно власного варіанту

|  |  |  |
| --- | --- | --- |
| № варіанту | Предметна область | Сутності та ïx поля (властивості). Головні сутності (довідники) **виділені**, зв'язувальна -підкреслена |
| 1 | 2 | 3 |
| 12. | Міський транспорт | **Транспорт** (вид, вартість проїзду, кількість  одиниць, рентабельність), Маршрут (початковий пункт, кінцевий пункт, кількість зупинок, довжина, кількість часу в  путі),  Маршрутний ресстр (транспорт, маршрут, напрямок pyxy, час виїзду з початкового пункту,  час прибуття в кінцевий пункт) |

Встановлено MySQL-cepвep та будь-який МуЅQL-клієнт, наприклад MySQL Workbench, та MY-SQL-connectors для мови С#.

**Завдання 1**

За допомогою клієнту MySQL огляду серверів баз даних створити базу даних а в ній три таблиці предметної сфери згідно власного варіанту. Кожна з таблиць має містити не менш ніж 20 записів реалістичних (абстрактна інформація на зразок: data1, data2, data3 та ін. чи ім’я1, ім’я2, ім’яЗ та ін. категорично заборонена) реалістичних даних. Зв'язки між таблицями можна підтримувати a6o на фізичному, a6o на логічному рівні за смаком виконавця. За допомогою зовнішнього клієнту огляду баз даних, a6o вбудованого в MS Visual Studio провідника серверів продемонструвати вміст створених таблиць

**Текст програми:**

***Program.cs:***

using System;

using MySql.Data.MySqlClient;

namespace Task\_1

{

class Program

{

static void Main(string[] args)

{

Console.InputEncoding = System.Text.Encoding.UTF8;

Console.OutputEncoding = System.Text.Encoding.UTF8;

// Підключення до MySQL сервера

string connStr = "server=localhost;user=root;password=mnc5UbTRn\_nZ+M:;";

// Створення підключення

MySqlConnection conn = new MySqlConnection(connStr);

try

{

conn.Open();

// Створення бази даних

string createDbQuery = "CREATE DATABASE IF NOT EXISTS CityTransportDB;";

MySqlCommand cmd = new MySqlCommand(createDbQuery, conn);

cmd.ExecuteNonQuery();

Console.WriteLine("База даних успішно створена.");

// Використання нової бази даних

cmd.CommandText = "USE CityTransportDB;";

cmd.ExecuteNonQuery();

// Створення таблиць

string createTransportTable = @"CREATE TABLE IF NOT EXISTS Transport (

TransportID INT AUTO\_INCREMENT PRIMARY KEY,

Type VARCHAR(50),

Fare DECIMAL(5,2),

Units INT,

Profitability DECIMAL(5,2)

);";

string createRouteTable = @"CREATE TABLE IF NOT EXISTS Route (

RouteID INT AUTO\_INCREMENT PRIMARY KEY,

StartPoint VARCHAR(100),

EndPoint VARCHAR(100),

Stops INT,

Length DECIMAL(5,2),

TravelTime INT

);";

string createRouteRegisterTable = @"CREATE TABLE IF NOT EXISTS RouteRegister (

RouteRegisterID INT AUTO\_INCREMENT PRIMARY KEY,

TransportID INT,

RouteID INT,

Direction VARCHAR(50),

DepartureTime TIME,

ArrivalTime TIME,

FOREIGN KEY (TransportID) REFERENCES Transport(TransportID),

FOREIGN KEY (RouteID) REFERENCES Route(RouteID)

);";

cmd.CommandText = createTransportTable;

cmd.ExecuteNonQuery();

cmd.CommandText = createRouteTable;

cmd.ExecuteNonQuery();

cmd.CommandText = createRouteRegisterTable;

cmd.ExecuteNonQuery();

Console.WriteLine("Таблиці успішно створені.");

// Вставка даних у таблицю Transport

string insertTransportData = @"INSERT INTO Transport (Type, Fare, Units, Profitability) VALUES

('Автобус', 1.50, 50, 10.5),

('Трамвай', 1.20, 30, 8.0),

('Тролейбус', 1.00, 40, 7.5),

('Метро', 2.00, 100, 15.0),

('Таксі', 10.00, 20, 25.0),

('Маршрутка', 2.50, 25, 12.0),

('Паром', 3.00, 5, 20.0),

('Потяг', 5.00, 10, 18.0),

('Прокат велосипедів', 0.50, 200, 5.0),

('Електросамокат', 0.75, 150, 6.5),

('Канатна дорога', 2.50, 3, 22.0),

('Фунікулер', 2.00, 2, 15.0),

('Монорельс', 2.20, 8, 13.5),

('Експрес в аеропорт', 8.00, 2, 30.0),

('Шатл', 1.80, 15, 9.0),

('Туристичний автобус', 12.00, 4, 40.0),

('Водне таксі', 7.00, 3, 28.0),

('Гелікоптер', 150.00, 1, 50.0),

('Сегвей', 2.00, 10, 5.5),

('Кінний віз', 5.00, 2, 12.5);";

cmd.CommandText = insertTransportData;

cmd.ExecuteNonQuery();

Console.WriteLine("Дані в таблицю Transport успішно вставлені.");

// Вставка даних у таблицю Route

string insertRouteData = @"INSERT INTO Route (StartPoint, EndPoint, Stops, Length, TravelTime) VALUES

('Центральний вокзал', 'Міський парк', 5, 10.0, 30),

('Аеропорт', 'Центр міста', 8, 15.5, 45),

('Порт', 'Музейний квартал', 6, 12.0, 35),

('Університет', 'Старе місто', 7, 14.0, 40),

('Стадіон', 'Міська рада', 4, 8.5, 25),

('Зоопарк', 'Океанаріум', 9, 18.0, 50),

('Передмістя', 'Промислова зона', 10, 20.0, 55),

('Пляж', 'Гора', 12, 25.0, 60),

('Східний район', 'Західний район', 15, 30.0, 75),

('Північні ворота', 'Південні ворота', 11, 22.0, 55),

('Кінотеатр', 'Бібліотека', 3, 6.0, 20),

('Лікарня', 'Пожежна станція', 5, 10.0, 30),

('Поліція', 'Суд', 4, 9.0, 25),

('Торговий центр', 'Конференц-центр', 6, 13.0, 35),

('Парк', 'Озеро', 7, 14.5, 40),

('Галерея', 'Ботанічний сад', 5, 11.0, 30),

('Залізничний вокзал', 'Автовокзал', 2, 4.5, 15),

('Оперний театр', 'Театр', 3, 7.0, 20),

('Склад', 'Логістичний центр', 8, 16.0, 45),

('Пам''ятник', 'Оглядовий майданчик', 5, 10.5, 30);";

cmd.CommandText = insertRouteData;

cmd.ExecuteNonQuery();

Console.WriteLine("Дані в таблицю Route успішно вставлені.");

// Вставка даних у таблицю RouteRegister

string insertRouteRegisterData = @"INSERT INTO RouteRegister (TransportID, RouteID, Direction, DepartureTime, ArrivalTime) VALUES

(1, 1, 'Туди', '08:00:00', '08:30:00'),

(2, 2, 'Назад', '09:00:00', '09:45:00'),

(3, 3, 'Туди', '07:15:00', '07:50:00'),

(4, 4, 'Назад', '10:00:00', '10:40:00'),

(5, 5, 'Туди', '12:00:00', '12:25:00'),

(6, 6, 'Назад', '13:30:00', '14:20:00'),

(7, 7, 'Туди', '14:00:00', '14:55:00'),

(8, 8, 'Назад', '16:00:00', '17:00:00'),

(9, 9, 'Туди', '17:30:00', '18:45:00'),

(10, 10, 'Назад', '19:00:00', '19:55:00'),

(11, 11, 'Туди', '20:00:00', '20:20:00'),

(12, 12, 'Назад', '21:00:00', '21:30:00'),

(13, 13, 'Туди', '06:00:00', '06:25:00'),

(14, 14, 'Назад', '07:30:00', '08:05:00'),

(15, 15, 'Туди', '09:30:00', '10:10:00'),

(16, 16, 'Назад', '11:00:00', '11:30:00'),

(17, 17, 'Туди', '12:30:00', '12:45:00'),

(18, 18, 'Назад', '13:00:00', '13:20:00'),

(19, 19, 'Туди', '15:00:00', '15:45:00'),

(20, 20, 'Назад', '16:30:00', '17:00:00');";

cmd.CommandText = insertRouteRegisterData;

cmd.ExecuteNonQuery();

Console.WriteLine("Дані в таблицю RouteRegister успішно вставлені.");

// Виведення даних з таблиць

// Виведення таблиці Transport

cmd.CommandText = "SELECT \* FROM Transport;";

MySqlDataReader reader = cmd.ExecuteReader();

Console.WriteLine("\nТаблиця Transport:");

while (reader.Read())

{

Console.WriteLine(

$"ID: {reader["TransportID"]}, " +

$"Тип: {reader["Type"]}, " +

$"Вартість: {reader["Fare"]}, " +

$"Одиниці: {reader["Units"]}, " +

$"Рентабельність: {reader["Profitability"]}"

);

}

reader.Close();

// Виведення таблиці Route

cmd.CommandText = "SELECT \* FROM Route;";

reader = cmd.ExecuteReader();

Console.WriteLine("\nТаблиця Route:");

while (reader.Read())

{

Console.WriteLine(

$"ID: {reader["RouteID"]}, " +

$"Початок: {reader["StartPoint"]}, " +

$"Кінець: {reader["EndPoint"]}, " +

$"Зупинки: {reader["Stops"]}, " +

$"Довжина: {reader["Length"]}, " +

$"Час у дорозі: {reader["TravelTime"]}"

);

}

reader.Close();

// Виведення таблиці RouteRegister

cmd.CommandText = @"SELECT rr.RouteRegisterID, t.Type, r.StartPoint, r.EndPoint, rr.Direction, rr.DepartureTime, rr.ArrivalTime

FROM RouteRegister rr

JOIN Transport t ON rr.TransportID = t.TransportID

JOIN Route r ON rr.RouteID = r.RouteID;";

reader = cmd.ExecuteReader();

Console.WriteLine("\nТаблиця RouteRegister:");

while (reader.Read())

{

Console.WriteLine(

$"ID: {reader["RouteRegisterID"]}, " +

$"Транспорт: {reader["Type"]}, " +

$"Маршрут: {reader["StartPoint"]} - {reader["EndPoint"]}, " +

$"Напрямок: {reader["Direction"]}, " +

$"Відправлення: {reader["DepartureTime"]}, " +

$"Прибуття: {reader["ArrivalTime"]}"

);

}

reader.Close();

conn.Close();

Console.WriteLine("\nВсі дані успішно виведені.");

Console.WriteLine("\nПрограма завершена. Натисніть будь-яку клавішу для виходу.");

Console.ReadKey();

}

catch (Exception ex)

{

Console.WriteLine("Виникла помилка: " + ex.Message);

Console.WriteLine("Деталі помилки: " + ex.StackTrace);

Console.WriteLine("\nНатисніть будь-яку клавішу для виходу.");

Console.ReadKey();

}

}

}

}

**Результат роботи програми:**

Изображение выглядит как текст, снимок экрана, черный, дизайн

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, черно-белый, дизайн

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, черно-белый, дизайн

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Изображение выглядит как снимок экрана, текст, программное обеспечение, число

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

Изображение выглядит как снимок экрана, текст, программное обеспечение, Значок на компьютере

Автоматически созданное описание

**Алгоритм побудови коду:**

1. **Підключення до MySQL**:
   * Встановлення з'єднання через MySqlConnection.
   * Визначення рядка підключення.
2. **Створення бази даних**:
   * Виконання SQL-запиту CREATE DATABASE IF NOT EXISTS.
3. **Створення таблиць**:
   * Таблиця **Transport**:
     + Ідентифікатор транспорту, тип, вартість, кількість одиниць, рентабельність.
   * Таблиця **Route**:
     + Ідентифікатор маршруту, початкова та кінцева точки, кількість зупинок, довжина, час подорожі.
   * Таблиця **RouteRegister**:
     + Ідентифікатор запису, зв'язок із транспортом і маршрутом, напрямок, час відправлення та прибуття.
4. **Наповнення таблиць**:
   * Використання SQL-запитів INSERT INTO для додавання не менш ніж 20 записів у кожну таблицю.
5. **Виведення даних**:
   * SQL-запити SELECT для читання та виведення вмісту кожної таблиці:
     + Виведення усіх записів із таблиць.
     + Об’єднання даних за допомогою JOIN для зв’язку таблиць.
6. **Закриття з'єднання**:
   * Завершення роботи з базою даних через метод Close.

**Функціональність:**

1. **Створення бази даних**:
   * Автоматичне створення бази даних, якщо вона не існує.
2. **Створення та заповнення таблиць**:
   * Таблиці містять реалістичні дані, які забезпечують зв'язок між транспортом, маршрутами та реєстраціями.
3. **Виведення даних**:
   * Консольне відображення:
     + Даних таблиці **Transport**.
     + Даних таблиці **Route**.
     + Зв'язків між таблицями через таблицю **RouteRegister**.
4. **Обробка помилок**:
   * Перехоплення та виведення повідомлень про помилки під час виконання запитів.

**Висновок:**

Програма забезпечує створення та заповнення бази даних із трьома таблицями, їх зв'язування через логічні ключі та виведення вмісту. Реалізація коду дозволяє гнучко налаштовувати дані, виконувати обробку запитів та аналізувати інформацію. Завдання виконано відповідно до вимог, а код є простим для модифікації та повторного використання.

**Завдання 2**

Створити консольний об'єктно-орієнтований застосунок, який друкує в консолі табличну інформацію відповідної предметної сфери та демонструє зв'язки між таблицями. У якості демонстрації зв'язків треба показати відбір інформації одного довідника за інформацією з довідника іншого.

**Текст програми:**

***Program.cs:***

using System;

using System.Collections.Generic;

using System.Globalization;

using System.Linq;

using System.Text;

using MySql.Data.MySqlClient;

namespace CityTransportApp

{

// Класи Transport, Route та RouteRegister

public class Transport

{

public int TransportID { get; set; }

public string Type { get; set; }

public decimal Fare { get; set; }

public int Units { get; set; }

public decimal Profitability { get; set; }

}

public class Route

{

public int RouteID { get; set; }

public string StartPoint { get; set; }

public string EndPoint { get; set; }

public int Stops { get; set; }

public decimal Length { get; set; }

public int TravelTime { get; set; }

}

public class RouteRegister

{

public int RouteRegisterID { get; set; }

public int TransportID { get; set; }

public int RouteID { get; set; }

public string Direction { get; set; }

public TimeSpan DepartureTime { get; set; }

public TimeSpan ArrivalTime { get; set; }

// Властивості для зв'язку з іншими класами

public Transport Transport { get; set; }

public Route Route { get; set; }

}

class Program

{

// Рядок підключення до бази даних

static string connStr = "server=localhost;user=root;password=mnc5UbTRn\_nZ+M:;database=CityTransportDB;";

static void Main(string[] args)

{

Console.InputEncoding = System.Text.Encoding.UTF8;

Console.OutputEncoding = System.Text.Encoding.UTF8;

CultureInfo.DefaultThreadCurrentCulture = new CultureInfo("uk-UA");

CultureInfo.DefaultThreadCurrentUICulture = new CultureInfo("uk-UA");

try

{

// Створюємо списки для зберігання даних

List<Transport> transports = new List<Transport>();

List<Route> routes = new List<Route>();

List<RouteRegister> routeRegisters = new List<RouteRegister>();

// Підключення до бази даних

using (MySqlConnection conn = new MySqlConnection(connStr))

{

conn.Open();

// Завантаження даних з таблиці Transport

string selectTransport = "SELECT \* FROM Transport;";

MySqlCommand cmd = new MySqlCommand(selectTransport, conn);

MySqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

Transport transport = new Transport()

{

TransportID = Convert.ToInt32(reader["TransportID"]),

Type = reader["Type"].ToString(),

Fare = Convert.ToDecimal(reader["Fare"]),

Units = Convert.ToInt32(reader["Units"]),

Profitability = Convert.ToDecimal(reader["Profitability"])

};

transports.Add(transport);

}

reader.Close();

// Завантаження даних з таблиці Route

string selectRoute = "SELECT \* FROM Route;";

cmd.CommandText = selectRoute;

reader = cmd.ExecuteReader();

while (reader.Read())

{

Route route = new Route()

{

RouteID = Convert.ToInt32(reader["RouteID"]),

StartPoint = reader["StartPoint"].ToString(),

EndPoint = reader["EndPoint"].ToString(),

Stops = Convert.ToInt32(reader["Stops"]),

Length = Convert.ToDecimal(reader["Length"]),

TravelTime = Convert.ToInt32(reader["TravelTime"])

};

routes.Add(route);

}

reader.Close();

// Завантаження даних з таблиці RouteRegister

string selectRouteRegister = "SELECT \* FROM RouteRegister;";

cmd.CommandText = selectRouteRegister;

reader = cmd.ExecuteReader();

while (reader.Read())

{

RouteRegister rr = new RouteRegister()

{

RouteRegisterID = Convert.ToInt32(reader["RouteRegisterID"]),

TransportID = Convert.ToInt32(reader["TransportID"]),

RouteID = Convert.ToInt32(reader["RouteID"]),

Direction = reader["Direction"].ToString(),

DepartureTime = (TimeSpan)reader["DepartureTime"],

ArrivalTime = (TimeSpan)reader["ArrivalTime"]

};

routeRegisters.Add(rr);

}

reader.Close();

// Встановлення зв'язків між об'єктами

foreach (var rr in routeRegisters)

{

rr.Transport = transports.Find(t => t.TransportID == rr.TransportID);

rr.Route = routes.Find(r => r.RouteID == rr.RouteID);

}

conn.Close();

}

// Виведення таблиці Transport

Console.WriteLine("Таблиця Transport:");

PrintTransportTable(transports);

// Виведення таблиці Route

Console.WriteLine("\nТаблиця Route:");

PrintRouteTable(routes);

// Виведення таблиці RouteRegister

Console.WriteLine("\nТаблиця RouteRegister:");

PrintRouteRegisterTable(routeRegisters);

// Головний цикл програми

while (true)

{

// Запит ID маршруту

Console.WriteLine("\nВведіть ID маршруту для відображення транспортів (або 'exit' для виходу):");

string inputRouteId = Console.ReadLine();

if (inputRouteId.Trim().ToLower() == "exit")

break;

if (!int.TryParse(inputRouteId, out int routeId))

{

Console.WriteLine("Неправильний формат ID маршруту. Спробуйте ще раз.");

continue;

}

var transportsOnRoute = routeRegisters.FindAll(rr => rr.RouteID == routeId);

if (transportsOnRoute.Count == 0)

{

Console.WriteLine($"Немає транспортів на маршруті з ID {routeId}.");

}

else

{

Console.WriteLine($"\nТранспорти на маршруті ID {routeId}:");

PrintRouteRegisterTable(transportsOnRoute);

}

// Запит типу транспорту

Console.WriteLine("\nВведіть тип транспорту для відображення маршрутів (або 'exit' для виходу):");

string transportType = Console.ReadLine();

if (transportType.Trim().ToLower() == "exit")

break;

var routesWithTransport = routeRegisters.FindAll(rr => rr.Transport.Type.Equals(transportType, StringComparison.OrdinalIgnoreCase));

if (routesWithTransport.Count == 0)

{

Console.WriteLine($"Немає маршрутів з транспортом '{transportType}'.");

}

else

{

Console.WriteLine($"\nМаршрути з транспортом '{transportType}':");

PrintRouteRegisterTable(routesWithTransport);

}

}

Console.WriteLine("\nПрограма завершена. Натисніть будь-яку клавішу для виходу.");

Console.ReadKey();

}

catch (Exception ex)

{

Console.WriteLine("Виникла помилка: " + ex.Message);

Console.WriteLine("Деталі помилки: " + ex.StackTrace);

Console.WriteLine("\nНатисніть будь-яку клавішу для виходу.");

Console.ReadKey();

}

}

// Вивід таблиці Transport (динам. шир.)

static void PrintTransportTable(List<Transport> transports)

{

// Макс. довжини для кожної колонки

int idWidth = Math.Max("ID".Length, transports.Max(t => t.TransportID.ToString().Length)) + 2;

int typeWidth = Math.Max("Тип".Length, transports.Max(t => t.Type.Length)) + 2;

int fareWidth = Math.Max("Вартість".Length, transports.Max(t => t.Fare.ToString("C").Length)) + 2;

int unitsWidth = Math.Max("Одиниці".Length, transports.Max(t => t.Units.ToString().Length)) + 2;

int profitWidth = Math.Max("Рентабельність".Length, transports.Max(t => (t.Profitability / 100).ToString("P").Length)) + 2;

string separator = new string('-', idWidth + typeWidth + fareWidth + unitsWidth + profitWidth + 13);

// Формат для рядків

string rowFormat = $"| {{0,-{idWidth}}} | " +

$"{{1,-{typeWidth}}} | " +

$"{{2,-{fareWidth}}} | " +

$"{{3,-{unitsWidth}}} | " +

$"{{4,-{profitWidth}}} |";

Console.WriteLine(separator);

Console.WriteLine(rowFormat, "ID", "Тип", "Вартість", "Одиниці", "Рентабельність");

Console.WriteLine(separator);

foreach (var t in transports)

{

Console.WriteLine(

rowFormat,

t.TransportID,

t.Type,

t.Fare.ToString("C"),

t.Units,

(t.Profitability / 100).ToString("P")

);

}

Console.WriteLine(separator);

}

// Вивід таблиці Route (динам. шир.)

static void PrintRouteTable(List<Route> routes)

{

// Макс. довжини для кожної колонки

int idWidth = Math.Max("ID".Length, routes.Max(r => r.RouteID.ToString().Length)) + 2;

int startWidth = Math.Max("Початок".Length, routes.Max(r => r.StartPoint.Length)) + 2;

int endWidth = Math.Max("Кінець".Length, routes.Max(r => r.EndPoint.Length)) + 2;

int stopsWidth = Math.Max("Зупинки".Length, routes.Max(r => r.Stops.ToString().Length)) + 2;

int lengthWidth = Math.Max("Довжина".Length, routes.Max(r => r.Length.ToString().Length + 4)) + 2; // +4 для " км"

int timeWidth = Math.Max("Час у дорозі".Length, routes.Max(r => r.TravelTime.ToString().Length + 4)) + 2; // +4 для " хв"

string separator = new string('-', idWidth + startWidth + endWidth + stopsWidth + lengthWidth + timeWidth + 19);

// Формат для рядків

string rowFormat = $"| {{0,-{idWidth}}} | " +

$"{{1,-{startWidth}}} | " +

$"{{2,-{endWidth}}} | " +

$"{{3,-{stopsWidth}}} | " +

$"{{4,-{lengthWidth}}} | " +

$"{{5,-{timeWidth}}} |";

Console.WriteLine(separator);

Console.WriteLine(rowFormat, "ID", "Початок", "Кінець", "Зупинки", "Довжина", "Час у дорозі");

Console.WriteLine(separator);

foreach (var r in routes)

{

Console.WriteLine(

rowFormat,

r.RouteID,

r.StartPoint,

r.EndPoint,

r.Stops,

r.Length + " км",

r.TravelTime + " хв"

);

}

Console.WriteLine(separator);

}

// Вивід таблиці RouteRegister (динам. шир.)

static void PrintRouteRegisterTable(List<RouteRegister> routeRegisters)

{

// Макс. довжини для кожної колонки

int idWidth = Math.Max("ID".Length, routeRegisters.Max(rr => rr.RouteRegisterID.ToString().Length)) + 2;

int transportWidth = Math.Max("Транспорт".Length, routeRegisters.Max(rr => rr.Transport.Type.Length)) + 2;

int startWidth = Math.Max("Початок".Length, routeRegisters.Max(rr => rr.Route.StartPoint.Length)) + 2;

int endWidth = Math.Max("Кінець".Length, routeRegisters.Max(rr => rr.Route.EndPoint.Length)) + 2;

int directionWidth = Math.Max("Напрямок".Length, routeRegisters.Max(rr => rr.Direction.Length)) + 2;

int departWidth = Math.Max("Відправлення".Length, routeRegisters.Max(rr => rr.DepartureTime.ToString(@"hh\:mm").Length)) + 2;

int arrivalWidth = Math.Max("Прибуття".Length, routeRegisters.Max(rr => rr.ArrivalTime.ToString(@"hh\:mm").Length)) + 2;

string separator = new string('-', idWidth + transportWidth + startWidth + endWidth + directionWidth + departWidth + arrivalWidth + 19);

// Формат для рядків

string rowFormat = $"| {{0,-{idWidth}}} | " +

$"{{1,-{transportWidth}}} | " +

$"{{2,-{startWidth}}} | " +

$"{{3,-{endWidth}}} | " +

$"{{4,-{directionWidth}}} | " +

$"{{5,-{departWidth}}} | " +

$"{{6,-{arrivalWidth}}} |";

Console.WriteLine(separator);

Console.WriteLine(rowFormat, "ID", "Транспорт", "Початок", "Кінець", "Напрямок", "Відправлення", "Прибуття");

Console.WriteLine(separator);

foreach (var rr in routeRegisters)

{

Console.WriteLine(

rowFormat,

rr.RouteRegisterID,

rr.Transport.Type,

rr.Route.StartPoint,

rr.Route.EndPoint,

rr.Direction,

rr.DepartureTime.ToString(@"hh\:mm"),

rr.ArrivalTime.ToString(@"hh\:mm")

);

}

Console.WriteLine(separator);

}

}

}

**Результат роботи програми:**

Изображение выглядит как текст, снимок экрана, черный

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Мультимедийное программное обеспечение, Графическое программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Мультимедийное программное обеспечение, Графическое программное обеспечение

Автоматически созданное описание

Изображение выглядит как снимок экрана, текст, схема, дизайн

Автоматически созданное описание

**Алгоритм побудови коду:**

1. **Створення моделей даних**:
   * Реалізовані класи Transport, Route, RouteRegister з відповідними властивостями.
   * У класі RouteRegister встановлені зв'язки через властивості Transport і Route.
2. **Підключення до бази даних**:
   * Рядок підключення передано через MySQL-клієнт.
   * Дані завантажуються через SQL-запити SELECT у списки об'єктів C#.
3. **Завантаження даних**:
   * Читання таблиць Transport, Route, RouteRegister.
   * Встановлення зв'язків між об'єктами після завантаження.
4. **Вивід даних**:
   * Створені методи PrintTransportTable, PrintRouteTable, PrintRouteRegisterTable для форматованого виводу.
   * Таблиці відображаються в динамічному форматі, залежно від довжини даних.
5. **Фільтрація даних**:
   * Реалізовано два сценарії:
     1. Вибір транспортів за ID маршруту.
     2. Вибір маршрутів за типом транспорту.
   * Дані відбираються за допомогою LINQ.
6. **Реалізація інтерфейсу**:
   * Інтерактивний консольний інтерфейс для введення ID маршруту чи типу транспорту.
   * Можливість вийти з програми через команду exit.

**Функціональність:**

1. **Завантаження даних**:
   * Зчитування інформації з таблиць бази даних.
   * Автоматичне встановлення зв'язків між об'єктами RouteRegister, Transport і Route.
2. **Вивід інформації**:
   * Форматований табличний вивід для кожної таблиці.
   * Відображення назв колонок і даних з урахуванням їх довжини.
3. **Фільтрація**:
   * Вибір транспортів, які обслуговують заданий маршрут.
   * Вибір маршрутів, які обслуговуються заданим типом транспорту.
4. **Інтерактивність**:
   * Користувач вводить запити в реальному часі.
   * Програма обробляє введені дані та відображає результати.
5. **Обробка помилок**:
   * Реалізовано захист від некоректного введення.
   * Виводяться повідомлення у разі помилок.

**Висновок:**

Програма успішно реалізує об'єктно-орієнтований підхід для роботи з даними бази даних:

* Забезпечено зв’язки між таблицями.
* Реалізовано зручний інтерфейс для вибору та фільтрації даних.
* Код є масштабованим і легко адаптується для інших предметних областей.

**Завдання 3**

Створити застосунок з формою чи формами який демонструє табличний контент власної предметної області в режимі формуляру (різні властивості з підписами одного об'єкту демонструються одна над іншою, далі теж саме для іншого об'єкту i т.д.) та в режимі таблиці (однорідні властивості різних об'єктів демонструються одна над іншою у вигляді стовпчиків за допомогою компоненту gridview), додавши до відповідних класів відповідні методи, які б формували властивості об'єктів у вигляді рядку таблиці чи формуляру. Перемикання між режимами та таблицями зробити за умов згідно до свого варіанту.

|  |  |  |
| --- | --- | --- |
| № варіанту | Перемикання між режимами | Перемикання між таблицями |
| 1 | 2 | 3 |
| 12. | Списки, що розкриваються (combobox) | Радіо кнопки (radio button) |

**Текст програми:**

**Папка Models:**

***Route.cs:***

// Models/Route.cs

namespace Task\_3.Models

{

public class Route

{

public int RouteID { get; set; }

public string StartPoint { get; set; }

public string EndPoint { get; set; }

public int Stops { get; set; }

public decimal Length { get; set; }

public int TravelTime { get; set; }

// Метод для формуляру

public string ToForm()

{

return $"ID: {RouteID}\n" +

$"Початковий пункт: {StartPoint}\n" +

$"Кінцевий пункт: {EndPoint}\n" +

$"Зупинки: {Stops}\n" +

$"Довжина: {Length}\n" +

$"Час у дорозі: {TravelTime}";

}

}

}

***RouteRegister.cs:***

// Models/RouteRegister.cs

using System;

using Task\_3.Models;

namespace Task\_3.Models

{

public class RouteRegister

{

public int RouteRegisterID { get; set; }

public int TransportID { get; set; }

public int RouteID { get; set; }

public string Direction { get; set; }

public TimeSpan DepartureTime { get; set; }

public TimeSpan ArrivalTime { get; set; }

// Для зв'язку з Transport та Route

public Transport Transport { get; set; }

public Route Route { get; set; }

// Метод для формуляру

public string ToForm()

{

return $"ID: {RouteRegisterID}\n" +

$"Транспорт: {Transport.Type}\n" +

$"Маршрут: {Route.StartPoint} - {Route.EndPoint}\n" +

$"Напрямок: {Direction}\n" +

$"Відправлення: {DepartureTime}\n" +

$"Прибуття: {ArrivalTime}";

}

}

}

***Transport.cs:***

// Models/Transport.cs

namespace Task\_3.Models

{

public class Transport

{

public int TransportID { get; set; }

public string Type { get; set; }

public decimal Fare { get; set; }

public int Units { get; set; }

public decimal Profitability { get; set; }

// Метод для формуляру

public string ToForm()

{

return $"ID: {TransportID}\n" +

$"Тип: {Type}\n" +

$"Вартість: {Fare}\n" +

$"Одиниці: {Units}\n" +

$"Рентабельність: {Profitability}";

}

}

}

***Database.cs:***

using System.Collections.Generic;

using MySql.Data.MySqlClient;

using Task\_3.Models;

namespace Task\_3

{

public class Database

{

// Строка підключення

private string connectionString = "server=localhost;user=root;password=mnc5UbTRn\_nZ+M:;database=CityTransportDB;";

// Отримати транспорти

public List<Transport> GetTransports()

{

// Список транспорту

List<Transport> transports = new List<Transport>();

// Відкрити з'єднання

using (MySqlConnection conn = new MySqlConnection(connectionString))

{

conn.Open(); // Відкрити

string query = "SELECT \* FROM Transport;";

// Створити команду

MySqlCommand cmd = new MySqlCommand(query, conn);

// Виконати запит

MySqlDataReader reader = cmd.ExecuteReader();

// Читати дані

while (reader.Read())

{

// Створити транспорт

Transport transport = new Transport

{

TransportID = reader.GetInt32("TransportID"),

Type = reader.GetString("Type"),

Fare = reader.GetDecimal("Fare"),

Units = reader.GetInt32("Units"),

Profitability = reader.GetDecimal("Profitability")

};

transports.Add(transport); // Додати транспорт

}

}

return transports;

}

// Отримати маршрути

public List<Route> GetRoutes()

{

// Список маршрутів

List<Route> routes = new List<Route>();

// Відкрити з'єднання

using (MySqlConnection conn = new MySqlConnection(connectionString))

{

conn.Open(); // Відкрити

string query = "SELECT \* FROM Route;";

// Створити команду

MySqlCommand cmd = new MySqlCommand(query, conn);

// Виконати запит

MySqlDataReader reader = cmd.ExecuteReader();

// Читати дані

while (reader.Read())

{

// Створити маршрут

Route route = new Route

{

RouteID = reader.GetInt32("RouteID"),

StartPoint = reader.GetString("StartPoint"),

EndPoint = reader.GetString("EndPoint"),

Stops = reader.GetInt32("Stops"),

Length = reader.GetDecimal("Length"),

TravelTime = reader.GetInt32("TravelTime")

};

routes.Add(route); // Додати маршрут

}

}

return routes;

}

// Отримати реєстри маршрутів

public List<RouteRegister> GetRouteRegisters()

{

// Список реєстрів

List<RouteRegister> routeRegisters = new List<RouteRegister>();

// Відкрити з'єднання

using (MySqlConnection conn = new MySqlConnection(connectionString))

{

conn.Open(); // Відкрити

string query = @"SELECT rr.\*, t.\*, r.\* FROM RouteRegister rr

JOIN Transport t ON rr.TransportID = t.TransportID

JOIN Route r ON rr.RouteID = r.RouteID;";

// Створити команду

MySqlCommand cmd = new MySqlCommand(query, conn);

// Виконати запит

MySqlDataReader reader = cmd.ExecuteReader();

// Читати дані

while (reader.Read())

{

// Створити транспорт

Transport transport = new Transport

{

TransportID = reader.GetInt32("TransportID"),

Type = reader.GetString("Type"),

Fare = reader.GetDecimal("Fare"),

Units = reader.GetInt32("Units"),

Profitability = reader.GetDecimal("Profitability")

};

// Створити маршрут

Route route = new Route

{

RouteID = reader.GetInt32("RouteID"),

StartPoint = reader.GetString("StartPoint"),

EndPoint = reader.GetString("EndPoint"),

Stops = reader.GetInt32("Stops"),

Length = reader.GetDecimal("Length"),

TravelTime = reader.GetInt32("TravelTime")

};

// Створити реєстр маршруту

RouteRegister routeRegister = new RouteRegister

{

RouteRegisterID = reader.GetInt32("RouteRegisterID"),

TransportID = reader.GetInt32("TransportID"),

RouteID = reader.GetInt32("RouteID"),

Direction = reader.GetString("Direction"),

DepartureTime = reader.GetTimeSpan("DepartureTime"),

ArrivalTime = reader.GetTimeSpan("ArrivalTime"),

Transport = transport,

Route = route

};

routeRegisters.Add(routeRegister); // Додати реєстр

}

}

return routeRegisters;

}

}

}

***MainForm.cs:***

using System;

using System.Collections.Generic;

using System.Windows.Forms;

using Task\_3.Models;

namespace Task\_3

{

public partial class MainForm : Form

{

private Database db; // База даних

private List<Transport> transports; // Транспорт

private List<Route> routes; // Маршрути

private List<RouteRegister> routeRegisters; // Реєстри

public MainForm()

{

InitializeComponent();

db = new Database();

// Додавання елементів

if (comboBoxMode.Items.Count == 0)

{

comboBoxMode.Items.Add("Формуляр");

comboBoxMode.Items.Add("Таблиця");

}

comboBoxMode.SelectedIndex = 0; // Вибір за замовчуванням

// Підписка на події

comboBoxMode.SelectedIndexChanged += comboBoxMode\_SelectedIndexChanged; // Зміна режиму

radioButtonTransport.CheckedChanged += radioButton\_CheckedChanged; // Транспорт

radioButtonRoute.CheckedChanged += radioButton\_CheckedChanged; // Маршрут

radioButtonRouteRegister.CheckedChanged += radioButton\_CheckedChanged; // Реєстр

this.Load += MainForm\_Load;

}

private void MainForm\_Load(object sender, EventArgs e)

{

LoadData();

UpdateView();

}

private void LoadData()

{

try

{

transports = db.GetTransports(); // Отримати транспорти

routes = db.GetRoutes(); // Отримати маршрути

routeRegisters = db.GetRouteRegisters(); // Отримати реєстри

}

catch (Exception ex)

{

MessageBox.Show($"Помилка при завантаженні даних: {ex.Message}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error); // Повідомлення про помилку

}

}

private void comboBoxMode\_SelectedIndexChanged(object sender, EventArgs e)

{

UpdateView(); // Оновити вигляд

}

private void radioButton\_CheckedChanged(object sender, EventArgs e)

{

UpdateView(); // Оновити вигляд

}

private void UpdateView()

{

if (comboBoxMode.SelectedItem == null)

return; // Вихід якщо нічого не вибрано

string mode = comboBoxMode.SelectedItem.ToString(); // Отримати режим

if (mode == "Таблиця")

{

textBoxForm.Visible = false; // Сховати формуляр

dataGridView.Visible = true; // Показати таблицю

if (radioButtonTransport.Checked)

{

dataGridView.DataSource = transports; // Показати транспорти

}

else if (radioButtonRoute.Checked)

{

dataGridView.DataSource = routes; // Показати маршрути

}

else if (radioButtonRouteRegister.Checked)

{

dataGridView.DataSource = routeRegisters; // Показати реєстри

}

}

else if (mode == "Формуляр")

{

dataGridView.Visible = false;

textBoxForm.Visible = true;

textBoxForm.Clear(); // Очистити формуляр

if (radioButtonTransport.Checked)

{

foreach (var transport in transports)

{

textBoxForm.AppendText(transport.ToForm() + "\n\n"); // Додати транспорт

}

}

else if (radioButtonRoute.Checked)

{

foreach (var route in routes)

{

textBoxForm.AppendText(route.ToForm() + "\n\n"); // Додати маршрут

}

}

else if (radioButtonRouteRegister.Checked)

{

foreach (var rr in routeRegisters)

{

textBoxForm.AppendText(rr.ToForm() + "\n\n"); // Додати реєстр

}

}

}

}

}

}

**Результат роботи програми:**

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, Параллельный

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, Параллельный

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, Параллельный

Автоматически созданное описание

**Алгоритм побудови коду:**

1. **Моделі даних**:
   * Класи Transport, Route, RouteRegister описують таблиці бази даних.
   * Кожен клас містить метод ToForm(), який повертає текстовий опис для режиму формуляру.
2. **Робота з базою даних**:
   * Реалізовано клас Database для отримання даних із таблиць бази даних через MySQL.
   * Методи GetTransports, GetRoutes, GetRouteRegisters завантажують відповідні таблиці у списки.
3. **Графічний інтерфейс**:
   * Використано елементи ComboBox, RadioButton, DataGridView, TextBox для демонстрації даних.
   * Вибір між режимами та таблицями здійснюється через SelectedIndexChanged (для ComboBox) і CheckedChanged (для RadioButton).
4. **Функціональність програми**:
   * При виборі режиму Таблиця (ComboBox) і таблиці (RadioButton) у DataGridView відображаються відповідні дані.
   * При виборі режиму Формуляр у TextBox відображаються текстові описи об’єктів.
5. **Обробка помилок**:
   * Захист від помилок при з'єднанні з базою даних.
   * Відображення повідомлень у разі помилок

**Функціональність:**

1. **Режими відображення**:
   * **Формуляр**: властивості одного об’єкта виводяться одна під одною у форматі тексту.
   * **Таблиця**: всі об’єкти відображаються в таблиці (DataGridView).
2. **Вибір таблиці**:
   * Вибір між таблицями Transport, Route, RouteRegister через RadioButton.
3. **Перемикання режимів**:
   * Здійснюється через ComboBox:
     + При виборі "Формуляр" відображається текстовий формат.
     + При виборі "Таблиця" дані відображаються у вигляді таблиці.
4. **Дані з бази**:
   * Завантажуються через запити до MySQL.
   * Підтримуються зв’язки між таблицями (наприклад, між RouteRegister і Transport).

**Висновок:**

* **Реалізовано функціональний Windows Forms застосунок**:
  + Забезпечено відображення даних у двох режимах.
  + Передбачено зручне перемикання між таблицями та режимами.
  + Реалізовано зв’язок між таблицями бази даних.
* **Код легко розширюється**:
  + Нові таблиці чи моделі можна додати без значних змін в архітектурі.
  + Використання OOP полегшує управління даними.

**Завдання 4**

Створити застосунок з формою чи формами який дозволяв би сортувати вміст таблиць БД за зростанням чи зменшенням відповідно інформації стовпчика, на заголовку якого відбувається клацання мишею (перше клацання— зростання, друге — зменшення i т.д. ) . Перемикання між таблицями зробити за умов згідно до свого варіанту минулого завдання.

**Текст програми:**

**Папка Models:**

***Route.cs:***

// Models/Route.cs

namespace Task\_4.Models

{

public class Route

{

public int RouteID { get; set; }

public string StartPoint { get; set; }

public string EndPoint { get; set; }

public int Stops { get; set; }

public decimal Length { get; set; }

public int TravelTime { get; set; }

}

}

***RouteRegister.cs:***

// Models/RouteRegister.cs

using System;

namespace Task\_4.Models

{

public class RouteRegister

{

public int RouteRegisterID { get; set; }

public int TransportID { get; set; }

public int RouteID { get; set; }

public string Direction { get; set; }

public TimeSpan DepartureTime { get; set; }

public TimeSpan ArrivalTime { get; set; }

}

}

***Transport.cs:***

// Models/Transport.cs

namespace Task\_4.Models

{

public class Transport

{

public int TransportID { get; set; }

public string Type { get; set; }

public decimal Fare { get; set; }

public int Units { get; set; }

public decimal Profitability { get; set; }

}

}

***Database.cs:***

using System.Collections.Generic;

using MySql.Data.MySqlClient;

using Task\_4.Models;

namespace Task\_4

{

public class Database

{

// Підключення

private string connectionString = "server=localhost;user=root;password=mnc5UbTRn\_nZ+M:;database=CityTransportDB;";

// Отримати транспорти

public List<Transport> GetTransports()

{

List<Transport> transports = new List<Transport>();

// Відкрити з'єднання

using (MySqlConnection conn = new MySqlConnection(connectionString))

{

conn.Open(); // Відкрити

string query = "SELECT \* FROM Transport;";

// Створити команду

MySqlCommand cmd = new MySqlCommand(query, conn);

// Виконати читання

MySqlDataReader reader = cmd.ExecuteReader();

// Читати дані

while (reader.Read())

{

// Створити транспорт

Transport transport = new Transport

{

TransportID = reader.GetInt32("TransportID"),

Type = reader.GetString("Type"),

Fare = reader.GetDecimal("Fare"),

Units = reader.GetInt32("Units"),

Profitability = reader.GetDecimal("Profitability")

};

transports.Add(transport); // Додати транспорт

}

}

return transports;

}

// Отримати маршрути

public List<Route> GetRoutes()

{

// Список маршрутів

List<Route> routes = new List<Route>();

// Відкрити з'єднання

using (MySqlConnection conn = new MySqlConnection(connectionString))

{

conn.Open(); // Відкрити

string query = "SELECT \* FROM Route;";

// Створити команду

MySqlCommand cmd = new MySqlCommand(query, conn);

// Виконати читання

MySqlDataReader reader = cmd.ExecuteReader();

// Читати дані

while (reader.Read())

{

// Створити маршрут

Route route = new Route

{

RouteID = reader.GetInt32("RouteID"),

StartPoint = reader.GetString("StartPoint"),

EndPoint = reader.GetString("EndPoint"),

Stops = reader.GetInt32("Stops"),

Length = reader.GetDecimal("Length"),

TravelTime = reader.GetInt32("TravelTime")

};

routes.Add(route); // Додати маршрут

}

}

return routes;

}

// Отримати реєстри маршрутів

public List<RouteRegister> GetRouteRegisters()

{

// Список реєстрів

List<RouteRegister> routeRegisters = new List<RouteRegister>();

// Відкрити з'єднання

using (MySqlConnection conn = new MySqlConnection(connectionString))

{

conn.Open(); // Відкрити

string query = "SELECT \* FROM RouteRegister;";

// Створити команду

MySqlCommand cmd = new MySqlCommand(query, conn);

// Виконати читання

MySqlDataReader reader = cmd.ExecuteReader();

// Читати дані

while (reader.Read())

{

// Створити реєстр

RouteRegister routeRegister = new RouteRegister

{

RouteRegisterID = reader.GetInt32("RouteRegisterID"),

TransportID = reader.GetInt32("TransportID"),

RouteID = reader.GetInt32("RouteID"),

Direction = reader.GetString("Direction"),

DepartureTime = reader.GetTimeSpan("DepartureTime"),

ArrivalTime = reader.GetTimeSpan("ArrivalTime")

};

routeRegisters.Add(routeRegister); // Додати реєстр

}

}

return routeRegisters;

}

}

}

***MainForm.cs:***

using System;

using System.Collections.Generic;

using System.Windows.Forms;

using Task\_4.Models;

namespace Task\_4

{

public partial class MainForm : Form

{

private Database db; // База даних

private List<Transport> transports; // Транспорт

private List<Route> routes; // Маршрути

private List<RouteRegister> routeRegisters; // Реєстри

// Сортування стану

private Dictionary<string, bool> sortDirections;

public MainForm()

{

InitializeComponent();

db = new Database();

// Ініціалізація сортування

sortDirections = new Dictionary<string, bool>();

// Підписка на події

radioButtonTransport.CheckedChanged += RadioButton\_CheckedChanged; // Транспорт

radioButtonRoute.CheckedChanged += RadioButton\_CheckedChanged; // Маршрут

radioButtonRouteRegister.CheckedChanged += RadioButton\_CheckedChanged; // Реєстр

dataGridView.ColumnHeaderMouseClick += DataGridView\_ColumnHeaderMouseClick; // Сортування

this.Load += MainForm\_Load; // Завантаження форми

}

private void MainForm\_Load(object sender, EventArgs e)

{

LoadData();

UpdateView();

}

private void LoadData()

{

try

{

transports = db.GetTransports(); // Отримати транспорти

routes = db.GetRoutes(); // Отримати маршрути

routeRegisters = db.GetRouteRegisters(); // Отримати реєстри

}

catch (Exception ex)

{

MessageBox.Show($"Помилка при завантаженні даних: {ex.Message}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error); // Повідомлення про помилку

}

}

private void RadioButton\_CheckedChanged(object sender, EventArgs e)

{

UpdateView(); // Оновити вигляд

}

private void UpdateView()

{

if (radioButtonTransport.Checked)

{

dataGridView.DataSource = transports; // Відобразити транспорти

}

else if (radioButtonRoute.Checked)

{

dataGridView.DataSource = routes; // Відобразити маршрути

}

else if (radioButtonRouteRegister.Checked)

{

dataGridView.DataSource = routeRegisters; // Відобразити реєстри

}

// Очистити сортування

sortDirections.Clear();

}

private void DataGridView\_ColumnHeaderMouseClick(object sender, DataGridViewCellMouseEventArgs e)

{

string columnName = dataGridView.Columns[e.ColumnIndex].DataPropertyName; // Отримати колонку

if (!sortDirections.ContainsKey(columnName))

{

sortDirections[columnName] = true; // Зростання

}

else

{

sortDirections[columnName] = !sortDirections[columnName]; // Зміна напрямку

}

SortData(columnName, sortDirections[columnName]); // Сортувати дані

}

private void SortData(string columnName, bool ascending)

{

if (radioButtonTransport.Checked)

{

if (ascending)

{

transports.Sort((x, y) => CompareProperty(x, y, columnName)); // Сортування

}

else

{

transports.Sort((x, y) => CompareProperty(y, x, columnName)); // Сортування

}

dataGridView.DataSource = null; // Оновити

dataGridView.DataSource = transports; // Відобразити

}

else if (radioButtonRoute.Checked)

{

if (ascending)

{

routes.Sort((x, y) => CompareProperty(x, y, columnName)); // Сортування

}

else

{

routes.Sort((x, y) => CompareProperty(y, x, columnName)); // Сортування

}

dataGridView.DataSource = null; // Оновити

dataGridView.DataSource = routes; // Відобразити

}

else if (radioButtonRouteRegister.Checked)

{

if (ascending)

{

routeRegisters.Sort((x, y) => CompareProperty(x, y, columnName)); // Сортування

}

else

{

routeRegisters.Sort((x, y) => CompareProperty(y, x, columnName)); // Сортування

}

dataGridView.DataSource = null; // Оновити

dataGridView.DataSource = routeRegisters; // Відобразити

}

}

private int CompareProperty<T>(T x, T y, string propertyName)

{

var prop = typeof(T).GetProperty(propertyName); // Отримати властивість

var valueX = prop.GetValue(x); // Значення X

var valueY = prop.GetValue(y); // Значення Y

return Comparer<object>.Default.Compare(valueX, valueY); // Порівняти

}

}

}

**Результат роботи програми:**

Изображение выглядит как текст, снимок экрана, число, Параллельный

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, Параллельный

Автоматически созданное описание

Изображение выглядит как текст, число, снимок экрана, Параллельный

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, Параллельный

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, Параллельный

Автоматически созданное описание

**Алгоритм побудови коду**

1. **Моделі даних**:
   * Описані класи Transport, Route, RouteRegister, які представляють відповідні таблиці бази даних.
2. **Доступ до бази даних**:
   * Реалізований клас Database для взаємодії з базою даних MySQL.
   * Методи GetTransports, GetRoutes, GetRouteRegisters повертають списки об'єктів, завантажених із відповідних таблиць.
3. **Сортування даних**:
   * Для кожного стовпця відстежується напрямок сортування за допомогою словника sortDirections.
   * При натисканні на заголовок стовпця (подія ColumnHeaderMouseClick) викликається метод SortData, який сортує дані відповідного типу (Transport, Route або RouteRegister).
4. **Перемикання між таблицями**:
   * Реалізовано через RadioButton. Вибраний тип таблиці визначає, який список даних (transports, routes, routeRegisters) відображається у DataGridView.
5. **Відображення даних**:
   * Дані відображаються у компоненті DataGridView.
   * При перемиканні між таблицями або після сортування DataGridView оновлюється.

**Функціональність**

1. **Сортування даних**:
   * При натисканні на заголовок стовпця здійснюється сортування відповідного списку.
   * Напрямок сортування чергується між зростанням та зменшенням.
2. **Перемикання між таблицями**:
   * Вибір між Transport, Route та RouteRegister здійснюється за допомогою RadioButton.
3. **Зручність у використанні**:
   * Дані автоматично оновлюються в інтерфейсі при перемиканні між таблицями або зміні напрямку сортування.

**Висновок**

* **Реалізований застосунок**:
  + Підтримує сортування даних за зростанням/зменшенням.
  + Забезпечує зручне перемикання між таблицями.
  + Сумісний із будь-якою базою даних, якщо структура відповідає описаним моделям.
* **Переваги**:
  + Легко розширюється (можна додати нові таблиці та функції).
  + Універсальний підхід до сортування будь-яких даних.
* **Результат**:
  + Інтуїтивно зрозумілий інтерфейс, що підходить для демонстрації великих наборів даних.

**Завдання 5**

Створити застосунок з формою чи формами який дозволяв би фільтрувати вміст таблиць БД за інформацією з пошукового рядка, який міститься під рядком заголовків стовпчиків чи містить список значень пошукових полів (якщо інформація відповідного стовпчика не повторюється, наприклад ПІБ, ціна, кількість та ін.) та пошукові списки (якщо інформація відповідного стовпчика повторюється наприклад тип транспорту, посада, кваліфікація та ін.), які мають автоматично оновлюватись під час фільтрації. Перемикання між таблицями зробити за умов згідно до свого варіанту минулого завдання.

**Текст програми:**

**Папка Models:**

***Route.cs:***

namespace Task\_5.Models

{

public class Route

{

public int RouteID { get; set; }

public string StartPoint { get; set; }

public string EndPoint { get; set; }

public int Stops { get; set; }

public decimal Length { get; set; }

public int TravelTime { get; set; }

}

}

***RouteRegister.cs:***

using System;

namespace Task\_5.Models

{

public class RouteRegister

{

public int RouteRegisterID { get; set; }

public int TransportID { get; set; }

public int RouteID { get; set; }

public string Direction { get; set; }

public TimeSpan DepartureTime { get; set; }

public TimeSpan ArrivalTime { get; set; }

}

}

***Transport.cs:***

namespace Task\_5.Models

{

public class Transport

{

public int TransportID { get; set; }

public string Type { get; set; }

public decimal Fare { get; set; }

public int Units { get; set; }

public decimal Profitability { get; set; }

}

}

***Database.cs:***

using System.Collections.Generic;

using MySql.Data.MySqlClient;

using Task\_5.Models;

namespace Task\_5

{

public class Database

{

// Підключення

private string connectionString = "server=localhost;user=root;password=mnc5UbTRn\_nZ+M:;database=CityTransportDB;";

// Отримати транспорти

public List<Transport> GetTransports()

{

// Список транспорту

List<Transport> transports = new List<Transport>();

// Відкрити з'єднання

using (MySqlConnection conn = new MySqlConnection(connectionString))

{

conn.Open();

string query = "SELECT \* FROM Transport;";

// Створити команду

MySqlCommand cmd = new MySqlCommand(query, conn);

// Виконати читання

MySqlDataReader reader = cmd.ExecuteReader();

// Читати дані

while (reader.Read())

{

// Створити транспорт

Transport transport = new Transport

{

TransportID = reader.GetInt32("TransportID"), // ID транспорту

Type = reader.GetString("Type"), // Тип

Fare = reader.GetDecimal("Fare"), // Вартість

Units = reader.GetInt32("Units"), // Одиниці

Profitability = reader.GetDecimal("Profitability") // Прибутковість

};

transports.Add(transport); // Додати транспорт

}

}

return transports;

}

// Отримати маршрути

public List<Route> GetRoutes()

{

// Список маршрутів

List<Route> routes = new List<Route>();

// Відкрити з'єднання

using (MySqlConnection conn = new MySqlConnection(connectionString))

{

conn.Open();

string query = "SELECT \* FROM Route;";

// Створити команду

MySqlCommand cmd = new MySqlCommand(query, conn);

// Виконати читання

MySqlDataReader reader = cmd.ExecuteReader();

// Читати дані

while (reader.Read())

{

// Створити маршрут

Route route = new Route

{

RouteID = reader.GetInt32("RouteID"),

StartPoint = reader.GetString("StartPoint"),

EndPoint = reader.GetString("EndPoint"),

Stops = reader.GetInt32("Stops"),

Length = reader.GetDecimal("Length"),

TravelTime = reader.GetInt32("TravelTime")

};

routes.Add(route); // Додати маршрут

}

}

return routes;

}

// Отримати реєстри маршрутів

public List<RouteRegister> GetRouteRegisters()

{

// Список реєстрів

List<RouteRegister> routeRegisters = new List<RouteRegister>();

// Відкрити з'єднання

using (MySqlConnection conn = new MySqlConnection(connectionString))

{

conn.Open(); // Відкрити

string query = "SELECT \* FROM RouteRegister;";

// Створити команду

MySqlCommand cmd = new MySqlCommand(query, conn);

// Виконати читання

MySqlDataReader reader = cmd.ExecuteReader();

// Читати дані

while (reader.Read())

{

// Створити реєстр

RouteRegister routeRegister = new RouteRegister

{

RouteRegisterID = reader.GetInt32("RouteRegisterID"),

TransportID = reader.GetInt32("TransportID"),

RouteID = reader.GetInt32("RouteID"),

Direction = reader.GetString("Direction"),

DepartureTime = reader.GetTimeSpan("DepartureTime"),

ArrivalTime = reader.GetTimeSpan("ArrivalTime")

};

routeRegisters.Add(routeRegister); // Додати реєстр

}

}

return routeRegisters; // Повернути список

}

}

}

***MainForm.cs:***

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Windows.Forms;

using Task\_5.Models;

namespace Task\_5

{

public partial class MainForm : Form

{

private Database db;

private List<Transport> transports;

private List<Route> routes;

private List<RouteRegister> routeRegisters;

private List<Transport> filteredTransports; // Фільтр Транспорт

private List<Route> filteredRoutes; // Фільтр Маршрути

private List<RouteRegister> filteredRouteRegisters; // Фільтр Реєстри

private Dictionary<string, Control> filterControls; // Контролі фільтрів

public MainForm()

{

InitializeComponent();

db = new Database();

// Підписка на події

radioButtonTransport.CheckedChanged += RadioButton\_CheckedChanged;

radioButtonRoute.CheckedChanged += RadioButton\_CheckedChanged;

radioButtonRouteRegister.CheckedChanged += RadioButton\_CheckedChanged;

this.Load += MainForm\_Load; // Завантаження форми

}

private void MainForm\_Load(object sender, EventArgs e)

{

LoadData();

InitializeFilters();

UpdateView();

}

private void LoadData()

{

try

{

transports = db.GetTransports(); // Отримати транспорти

routes = db.GetRoutes(); // Отримати маршрути

routeRegisters = db.GetRouteRegisters(); // Отримати реєстри

}

catch (Exception ex)

{

MessageBox.Show($"Помилка при завантаженні даних: {ex.Message}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void InitializeFilters()

{

filterControls = new Dictionary<string, Control>(); // Створити словник

}

private void RadioButton\_CheckedChanged(object sender, EventArgs e)

{

UpdateView();

}

private void UpdateView()

{

panelFilters.Controls.Clear(); // Очистити панель

filterControls.Clear(); // Очистити контролі

if (radioButtonTransport.Checked)

{

filteredTransports = new List<Transport>(transports); // Фільтр Транспорт

dataGridView.DataSource = filteredTransports; // Відобразити транспорти

CreateFilters(typeof(Transport));

}

else if (radioButtonRoute.Checked)

{

filteredRoutes = new List<Route>(routes); // Фільтр Маршрути

dataGridView.DataSource = filteredRoutes; // Відобразити маршрути

CreateFilters(typeof(Route));

}

else if (radioButtonRouteRegister.Checked)

{

filteredRouteRegisters = new List<RouteRegister>(routeRegisters); // Фільтр Реєстри

dataGridView.DataSource = filteredRouteRegisters; // Відобразити реєстри

CreateFilters(typeof(RouteRegister));

}

dataGridView.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill; // Розмір колонок

}

private void CreateFilters(Type type)

{

int x = 0; // Початкова позиція X

int columnCount = dataGridView.Columns.Count; // Кількість колонок

for (int i = 0; i < columnCount; i++)

{

DataGridViewColumn column = dataGridView.Columns[i]; // Колонка

string propertyName = column.DataPropertyName; // Властивість

Control filterControl; // Контроль фільтра

// Визначення типу поля

var propInfo = type.GetProperty(propertyName);

if (propInfo.PropertyType == typeof(string))

{

// ComboBox для тексту

ComboBox comboBox = new ComboBox();

comboBox.Name = "filter\_" + propertyName;

comboBox.Width = column.Width;

comboBox.Left = column.DisplayIndex \* column.Width; // Позиція X

comboBox.Top = 0; // Позиція Y

comboBox.Anchor = AnchorStyles.Left | AnchorStyles.Top;

comboBox.DropDownStyle = ComboBoxStyle.DropDown; // Стиль

comboBox.TextChanged += FilterControl\_ValueChanged; // Подія зміни

panelFilters.Controls.Add(comboBox);

filterControl = comboBox; // Призначення контролю

// Заповнення значень

var values = GetDistinctValues(type, propertyName); // Унікальні значення

comboBox.Items.AddRange(values.ToArray()); // Додати елементи

}

else

{

// TextBox для чисел

TextBox textBox = new TextBox();

textBox.Name = "filter\_" + propertyName;

textBox.Width = column.Width;

textBox.Left = column.DisplayIndex \* column.Width; // Позиція X

textBox.Top = 0; // Позиція Y

textBox.Anchor = AnchorStyles.Left | AnchorStyles.Top;

textBox.TextChanged += FilterControl\_ValueChanged; // Подія зміни

panelFilters.Controls.Add(textBox);

filterControl = textBox; // Призначити контролю

}

filterControls[propertyName] = filterControl; // Додати до словника

}

}

private List<string> GetDistinctValues(Type type, string propertyName)

{

if (type == typeof(Transport))

{

return transports.Select(t => t.GetType().GetProperty(propertyName).GetValue(t)?.ToString())

.Distinct()

.OrderBy(v => v)

.ToList(); // Унікальні транспорти

}

else if (type == typeof(Route))

{

return routes.Select(r => r.GetType().GetProperty(propertyName).GetValue(r)?.ToString())

.Distinct()

.OrderBy(v => v)

.ToList(); // Унікальні маршрути

}

else if (type == typeof(RouteRegister))

{

return routeRegisters.Select(rr => rr.GetType().GetProperty(propertyName).GetValue(rr)?.ToString())

.Distinct()

.OrderBy(v => v)

.ToList(); // Унікальні реєстри

}

else

{

return new List<string>(); // Порожній список

}

}

private void FilterControl\_ValueChanged(object sender, EventArgs e)

{

ApplyFilters(); // Застосувати фільтри

}

private void ApplyFilters()

{

if (radioButtonTransport.Checked)

{

IEnumerable<Transport> query = transports; // Запит транспорти

foreach (var kvp in filterControls)

{

string propertyName = kvp.Key; // Властивість

Control control = kvp.Value; // Контроль

string filterValue = ""; // Значення фільтра

if (control is TextBox textBox)

{

filterValue = textBox.Text; // Отримати текст

}

else if (control is ComboBox comboBox)

{

filterValue = comboBox.Text; // Отримати текст

}

if (!string.IsNullOrEmpty(filterValue))

{

query = query.Where(t =>

{

var propertyValue = t.GetType()

.GetProperty(propertyName)

?.GetValue(t)

?.ToString();

return propertyValue?.Contains(filterValue, StringComparison.OrdinalIgnoreCase) == true;

});

}

}

filteredTransports = query.ToList(); // Результат фільтрації

dataGridView.DataSource = null; // Очистити джерело

dataGridView.DataSource = filteredTransports; // Відобразити

}

else if (radioButtonRoute.Checked)

{

IEnumerable<Route> query = routes; // Запит маршрути

foreach (var kvp in filterControls)

{

string propertyName = kvp.Key; // Властивість

Control control = kvp.Value; // Контроль

string filterValue = ""; // Значення фільтра

if (control is TextBox textBox)

{

filterValue = textBox.Text; // Отримати текст

}

else if (control is ComboBox comboBox)

{

filterValue = comboBox.Text; // Отримати текст

}

if (!string.IsNullOrEmpty(filterValue))

{

query = query.Where(r =>

{

var propertyValue = r.GetType()

.GetProperty(propertyName)

?.GetValue(r)

?.ToString();

return propertyValue?.Contains(filterValue, StringComparison.OrdinalIgnoreCase) == true;

});

}

}

filteredRoutes = query.ToList(); // Результат фільтрації

dataGridView.DataSource = null; // Очистити джерело

dataGridView.DataSource = filteredRoutes; // Відобразити

}

else if (radioButtonRouteRegister.Checked)

{

IEnumerable<RouteRegister> query = routeRegisters; // Запит реєстри

foreach (var kvp in filterControls)

{

string propertyName = kvp.Key; // Властивість

Control control = kvp.Value; // Контроль

string filterValue = ""; // Значення фільтра

if (control is TextBox textBox)

{

filterValue = textBox.Text; // Отримати текст

}

else if (control is ComboBox comboBox)

{

filterValue = comboBox.Text; // Отримати текст

}

if (!string.IsNullOrEmpty(filterValue))

{

query = query.Where(rr =>

{

var propertyValue = rr.GetType()

.GetProperty(propertyName)

?.GetValue(rr)

?.ToString();

return propertyValue?.Contains(filterValue, StringComparison.OrdinalIgnoreCase) == true;

});

}

}

filteredRouteRegisters = query.ToList(); // Результат фільтрації

dataGridView.DataSource = null; // Очистити джерело

dataGridView.DataSource = filteredRouteRegisters; // Відобразити

}

}

}

}

**Результат роботи програми:**

Изображение выглядит как текст, снимок экрана, число, Параллельный

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, дисплей, число

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

**Алгоритм побудови коду**

1. **Моделі даних**:
   * Описані класи Transport, Route, RouteRegister, які представляють відповідні таблиці бази даних.
2. **Доступ до бази даних**:
   * Реалізовано клас Database для отримання даних із бази MySQL.
   * Методи GetTransports, GetRoutes, GetRouteRegisters завантажують списки об'єктів із відповідних таблиць.
3. **Фільтрування даних**:
   * Для кожного стовпця генеруються фільтри:
     + TextBox для унікальних значень.
     + ComboBox для повторюваних значень.
   * Пошукові списки (ComboBox) автоматично оновлюються при зміні фільтрації.
4. **Перемикання між таблицями**:
   * За допомогою RadioButton визначається, яка таблиця відображається у DataGridView.
5. **Відображення даних**:
   * Дані відображаються в DataGridView, а панель фільтрів (Panel) динамічно оновлюється відповідно до вибраної таблиці.

**Функціональність**

1. **Фільтрація даних**:
   * Користувач може вводити текст у пошукові рядки або вибирати значення зі списків для фільтрації даних.
   * Фільтри враховують регістр символів та виконують пошук за частковим збігом.
2. **Автоматичне оновлення списків**:
   * При зміні фільтрації список значень у ComboBox оновлюється на основі відфільтрованих даних.
3. **Перемикання між таблицями**:
   * Використання RadioButton дозволяє легко змінювати таблиці, що відображаються.
4. **Динамічна генерація фільтрів**:
   * Панель із фільтрами автоматично створюється залежно від структури таблиці.
5. **Зручність інтерфейсу**:
   * Всі фільтри розташовані над таблицею, що дозволяє зручно виконувати пошук.

**Висновок**

* **Реалізований застосунок**:
  + Дозволяє ефективно фільтрувати дані в таблицях бази даних.
  + Використовує динамічне оновлення пошукових полів для підвищення зручності.
* **Переваги**:
  + Універсальність: підтримує будь-яку базу даних, структура якої відповідає описаним моделям.
  + Гнучкість: автоматичне створення фільтрів для кожного стовпця.
* **Результат**:
  + Інтуїтивний інтерфейс для пошуку даних у великому наборі таблиць.
  + Застосунок підходить для аналізу даних і зручного управління інформацією з бази даних.

**Завдання 6**

Створити застосунок з формою чи формами, який би містив таблиці з БД та інші діалогові елементи (текстовими полями, кнопками, списками та ін.), за допомогою яких можна було б додавати, видаляти та редагувати вміст таблиць та зберігати ïx в БД. Перемикання між таблицями зробити за умов згідно до свого варіанту минулого завдання.

**Текст програми:**

**Папка Models:**

***Route.cs:***

// Models/Route.cs

namespace Task\_6.Models

{

public class Route

{

public int RouteID { get; set; }

public string StartPoint { get; set; }

public string EndPoint { get; set; }

public int Stops { get; set; }

public decimal Length { get; set; }

public int TravelTime { get; set; }

}

}

***RouteRegister.cs:***

// Models/RouteRegister.cs

using System;

namespace Task\_6.Models

{

public class RouteRegister

{

public int RouteRegisterID { get; set; }

public int TransportID { get; set; }

public int RouteID { get; set; }

public string Direction { get; set; }

public TimeSpan DepartureTime { get; set; }

public TimeSpan ArrivalTime { get; set; }

}

}

***Transport.cs:***

// Models/Transport.cs

namespace Task\_6.Models

{

public class Transport

{

public int TransportID { get; set; }

public string Type { get; set; }

public decimal Fare { get; set; }

public int Units { get; set; }

public decimal Profitability { get; set; }

}

}

***Database.cs:***

using System;

using System.Collections.Generic;

using MySql.Data.MySqlClient;

using Task\_6.Models;

namespace Task\_6

{

public class Database

{

// Підключення

private string connectionString = "server=localhost;user=root;password=mnc5UbTRn\_nZ+M:;database=CityTransportDB;";

// Методи Transport

public List<Transport> GetTransports()

{

// Список транспорту

List<Transport> transports = new List<Transport>();

using (MySqlConnection conn = new MySqlConnection(connectionString))

{

conn.Open();

// Запит SQL

string query = "SELECT \* FROM Transport;";

MySqlCommand cmd = new MySqlCommand(query, conn); // Команда

MySqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

// Створити транспорт

Transport transport = new Transport

{

TransportID = reader.GetInt32("TransportID"),

Type = reader.GetString("Type"),

Fare = reader.GetDecimal("Fare"),

Units = reader.GetInt32("Units"),

Profitability = reader.GetDecimal("Profitability")

};

transports.Add(transport);

}

}

return transports;

}

public void AddTransport(Transport transport)

{

using (MySqlConnection conn = new MySqlConnection(connectionString))

{

conn.Open();

// Запит вставки

string query =

"INSERT INTO Transport (Type, Fare, Units, Profitability) " +

"VALUES (@Type, @Fare, @Units, @Profitability);";

MySqlCommand cmd = new MySqlCommand(query, conn);

cmd.Parameters.AddWithValue("@Type", transport.Type);

cmd.Parameters.AddWithValue("@Fare", transport.Fare);

cmd.Parameters.AddWithValue("@Units", transport.Units);

cmd.Parameters.AddWithValue("@Profitability", transport.Profitability);

cmd.ExecuteNonQuery(); // Виконати

}

}

public void UpdateTransport(Transport transport)

{

using (MySqlConnection conn = new MySqlConnection(connectionString))

{

conn.Open();

// Запит оновлення

string query =

"UPDATE Transport " +

"SET Type = @Type, Fare = @Fare, Units = @Units, Profitability = @Profitability " +

"WHERE TransportID = @TransportID;";

MySqlCommand cmd = new MySqlCommand(query, conn);

cmd.Parameters.AddWithValue("@TransportID", transport.TransportID);

cmd.Parameters.AddWithValue("@Type", transport.Type);

cmd.Parameters.AddWithValue("@Fare", transport.Fare);

cmd.Parameters.AddWithValue("@Units", transport.Units);

cmd.Parameters.AddWithValue("@Profitability", transport.Profitability);

cmd.ExecuteNonQuery(); // Виконати

}

}

public void DeleteTransport(int transportID)

{

using (MySqlConnection conn = new MySqlConnection(connectionString))

{

conn.Open();

// Запит видалення

string query = "DELETE FROM Transport WHERE TransportID = @TransportID;";

MySqlCommand cmd = new MySqlCommand(query, conn); // Команда

cmd.Parameters.AddWithValue("@TransportID", transportID); // Параметр ID

cmd.ExecuteNonQuery(); // Виконати

}

}

// Методи Route

public List<Route> GetRoutes()

{

// Список маршрутів

List<Route> routes = new List<Route>();

using (MySqlConnection conn = new MySqlConnection(connectionString))

{

conn.Open();

// Запит SQL

string query = "SELECT \* FROM Route;";

MySqlCommand cmd = new MySqlCommand(query, conn);

MySqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

// Створити маршрут

Route route = new Route

{

RouteID = reader.GetInt32("RouteID"),

StartPoint = reader.GetString("StartPoint"),

EndPoint = reader.GetString("EndPoint"),

Stops = reader.GetInt32("Stops"),

Length = reader.GetDecimal("Length"),

TravelTime = reader.GetInt32("TravelTime")

};

routes.Add(route);

}

}

return routes;

}

public void AddRoute(Route route)

{

using (MySqlConnection conn = new MySqlConnection(connectionString))

{

conn.Open();

// Запит вставки

string query =

"INSERT INTO Route (StartPoint, EndPoint, Stops, Length, TravelTime) " +

"VALUES (@StartPoint, @EndPoint, @Stops, @Length, @TravelTime);";

MySqlCommand cmd = new MySqlCommand(query, conn);

cmd.Parameters.AddWithValue("@StartPoint", route.StartPoint);

cmd.Parameters.AddWithValue("@EndPoint", route.EndPoint);

cmd.Parameters.AddWithValue("@Stops", route.Stops);

cmd.Parameters.AddWithValue("@Length", route.Length);

cmd.Parameters.AddWithValue("@TravelTime", route.TravelTime);

cmd.ExecuteNonQuery(); // Виконати

}

}

public void UpdateRoute(Route route)

{

using (MySqlConnection conn = new MySqlConnection(connectionString))

{

conn.Open(); // Відкрити

// Запит оновлення

string query =

"UPDATE Route " +

"SET StartPoint = @StartPoint, EndPoint = @EndPoint, Stops = @Stops, " +

"Length = @Length, TravelTime = @TravelTime " +

"WHERE RouteID = @RouteID;";

MySqlCommand cmd = new MySqlCommand(query, conn);

cmd.Parameters.AddWithValue("@RouteID", route.RouteID);

cmd.Parameters.AddWithValue("@StartPoint", route.StartPoint);

cmd.Parameters.AddWithValue("@EndPoint", route.EndPoint);

cmd.Parameters.AddWithValue("@Stops", route.Stops);

cmd.Parameters.AddWithValue("@Length", route.Length);

cmd.Parameters.AddWithValue("@TravelTime", route.TravelTime);

cmd.ExecuteNonQuery(); // Виконати

}

}

public void DeleteRoute(int routeID)

{

using (MySqlConnection conn = new MySqlConnection(connectionString))

{

conn.Open(); // Відкрити

// Запит видалення

string query = "DELETE FROM Route WHERE RouteID = @RouteID;";

MySqlCommand cmd = new MySqlCommand(query, conn); // Команда

cmd.Parameters.AddWithValue("@RouteID", routeID); // Параметр ID

cmd.ExecuteNonQuery(); // Виконати

}

}

// Методи RouteRegister

public List<RouteRegister> GetRouteRegisters()

{

// Список реєстрів

List<RouteRegister> routeRegisters = new List<RouteRegister>();

using (MySqlConnection conn = new MySqlConnection(connectionString))

{

conn.Open();

// Запит SQL

string query = "SELECT \* FROM RouteRegister;";

MySqlCommand cmd = new MySqlCommand(query, conn); // Команда

MySqlDataReader reader = cmd.ExecuteReader(); // Читання

while (reader.Read())

{

// Створити реєстр

RouteRegister routeRegister = new RouteRegister

{

RouteRegisterID = reader.GetInt32("RouteRegisterID"),

TransportID = reader.GetInt32("TransportID"),

RouteID = reader.GetInt32("RouteID"),

Direction = reader.GetString("Direction"),

DepartureTime = reader.GetTimeSpan("DepartureTime"),

ArrivalTime = reader.GetTimeSpan("ArrivalTime")

};

routeRegisters.Add(routeRegister); // Додати

}

}

return routeRegisters;

}

public void AddRouteRegister(RouteRegister routeRegister)

{

using (MySqlConnection conn = new MySqlConnection(connectionString))

{

conn.Open();

// Запит вставки

string query =

"INSERT INTO RouteRegister (TransportID, RouteID, Direction, DepartureTime, ArrivalTime) " +

"VALUES (@TransportID, @RouteID, @Direction, @DepartureTime, @ArrivalTime);";

MySqlCommand cmd = new MySqlCommand(query, conn);

cmd.Parameters.AddWithValue("@TransportID", routeRegister.TransportID);

cmd.Parameters.AddWithValue("@RouteID", routeRegister.RouteID);

cmd.Parameters.AddWithValue("@Direction", routeRegister.Direction);

cmd.Parameters.AddWithValue("@DepartureTime", routeRegister.DepartureTime);

cmd.Parameters.AddWithValue("@ArrivalTime", routeRegister.ArrivalTime);

cmd.ExecuteNonQuery(); // Виконати

}

}

public void UpdateRouteRegister(RouteRegister routeRegister)

{

using (MySqlConnection conn = new MySqlConnection(connectionString))

{

conn.Open(); // Відкрити

// Запит оновлення

string query =

"UPDATE RouteRegister " +

"SET TransportID = @TransportID, RouteID = @RouteID, Direction = @Direction, " +

"DepartureTime = @DepartureTime, ArrivalTime = @ArrivalTime " +

"WHERE RouteRegisterID = @RouteRegisterID;";

MySqlCommand cmd = new MySqlCommand(query, conn);

cmd.Parameters.AddWithValue("@RouteRegisterID", routeRegister.RouteRegisterID);

cmd.Parameters.AddWithValue("@TransportID", routeRegister.TransportID);

cmd.Parameters.AddWithValue("@RouteID", routeRegister.RouteID);

cmd.Parameters.AddWithValue("@Direction", routeRegister.Direction);

cmd.Parameters.AddWithValue("@DepartureTime", routeRegister.DepartureTime);

cmd.Parameters.AddWithValue("@ArrivalTime", routeRegister.ArrivalTime);

cmd.ExecuteNonQuery(); // Виконати

}

}

public void DeleteRouteRegister(int routeRegisterID)

{

using (MySqlConnection conn = new MySqlConnection(connectionString))

{

conn.Open(); // Відкрити

// Запит видалення

string query = "DELETE FROM RouteRegister WHERE RouteRegisterID = @RouteRegisterID;";

MySqlCommand cmd = new MySqlCommand(query, conn); // Команда

cmd.Parameters.AddWithValue("@RouteRegisterID", routeRegisterID); // Параметр ID

cmd.ExecuteNonQuery(); // Виконати

}

}

}

}

***MainForm.cs:***

using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows.Forms;

using Task\_6.Models;

namespace Task\_6

{

public partial class MainForm : Form

{

private Database db;

private List<Transport> transports;

private List<Route> routes;

private List<RouteRegister> routeRegisters;

public MainForm()

{

InitializeComponent();

db = new Database();

// Підписка на події

comboBoxMode.SelectedIndexChanged += ComboBoxMode\_SelectedIndexChanged; // Зміна режиму

radioButtonTransport.CheckedChanged += RadioButton\_CheckedChanged;

radioButtonRoute.CheckedChanged += RadioButton\_CheckedChanged;

radioButtonRouteRegister.CheckedChanged += RadioButton\_CheckedChanged;

buttonAction.Click += ButtonAction\_Click; // Кнопка дія

dataGridView.SelectionChanged += DataGridView\_SelectionChanged; // Вибір рядка

comboBoxMode.SelectedIndex = 0; // Вибір за замовчуванням

this.Load += MainForm\_Load;

}

private void MainForm\_Load(object sender, EventArgs e)

{

LoadData(); // Завантажити дані

UpdateView(); // Оновити вигляд

}

private void LoadData()

{

try

{

transports = db.GetTransports(); // Отримати транспорти

routes = db.GetRoutes(); // Отримати маршрути

routeRegisters = db.GetRouteRegisters(); // Отримати реєстри

}

catch (Exception ex)

{

MessageBox.Show($"Помилка при завантаженні даних: {ex.Message}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void UpdateView()

{

ClearInputFields(); // Очистити поля

if (radioButtonTransport.Checked)

{

dataGridView.DataSource = transports; // Відобразити транспорти

CreateInputFields(typeof(Transport)); // Поля вводу

}

else if (radioButtonRoute.Checked)

{

dataGridView.DataSource = routes; // Відобразити маршрути

CreateInputFields(typeof(Route));

}

else if (radioButtonRouteRegister.Checked)

{

dataGridView.DataSource = routeRegisters; // Відобразити реєстри

CreateInputFields(typeof(RouteRegister));

}

UpdateButtonText(); // Оновити кнопку

dataGridView.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill; // Авто розмір колонок

}

private void CreateInputFields(Type type)

{

panelInputs.Controls.Clear();

int x = 0; // Позиція X

int y = 0; // Позиція Y

int spacing = 10; // Відступ

var properties = type.GetProperties();

foreach (var prop in properties)

{

Label lbl = new Label

{

Text = prop.Name,

Location = new System.Drawing.Point(x, y), // Розташування

Width = 100

};

panelInputs.Controls.Add(lbl); // Додати мітку

Control inputControl;

if (prop.PropertyType == typeof(int) || prop.PropertyType == typeof(decimal) || prop.PropertyType == typeof(string))

{

TextBox textBox = new TextBox

{

Name = "input\_" + prop.Name,

Location = new System.Drawing.Point(x + 110, y), // Розташування

Width = 150

};

// Якщо це поле ID і режим - "Додавання", дозволяємо вводити, інакше робимо поле тільки для читання

if (prop.Name.EndsWith("ID") && comboBoxMode.SelectedItem.ToString() != "Додавання")

{

textBox.ReadOnly = true;

}

panelInputs.Controls.Add(textBox);

inputControl = textBox;

}

else if (prop.PropertyType == typeof(TimeSpan))

{

DateTimePicker timePicker = new DateTimePicker

{

Name = "input\_" + prop.Name,

Location = new System.Drawing.Point(x + 110, y), // Розташування

Width = 150,

Format = DateTimePickerFormat.Time, // Формат часу

ShowUpDown = true // Стиль

};

panelInputs.Controls.Add(timePicker); // Додати вибір часу

inputControl = timePicker;

}

else

{

continue; // Пропустити невідомий тип

}

y += inputControl.Height + spacing; // Збільшити Y

}

}

private void UpdateButtonText()

{

string mode = comboBoxMode.SelectedItem.ToString(); // Отримати режим

buttonAction.Text = mode;

}

private void ComboBoxMode\_SelectedIndexChanged(object sender, EventArgs e)

{

UpdateButtonText();

ClearInputFields();

UpdateView();

}

private void RadioButton\_CheckedChanged(object sender, EventArgs e)

{

UpdateView();

}

private void ButtonAction\_Click(object sender, EventArgs e)

{

string mode = comboBoxMode.SelectedItem.ToString(); // Отримати режим

try

{

if (mode == "Додавання")

{

AddRecord(); // Додати запис

}

else if (mode == "Редагування")

{

EditRecord(); // Редагувати запис

}

else if (mode == "Видалення")

{

DeleteRecord(); // Видалити запис

}

LoadData();

UpdateView();

}

catch (Exception ex)

{

MessageBox.Show($"Помилка: {ex.Message}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void AddRecord()

{

if (radioButtonTransport.Checked)

{

Transport transport = new Transport(); // Новий транспорт

FillObjectFromInputs(transport); // Заповнити дані

db.AddTransport(transport);

}

else if (radioButtonRoute.Checked)

{

Route route = new Route(); // Новий маршрут

FillObjectFromInputs(route);

db.AddRoute(route);

}

else if (radioButtonRouteRegister.Checked)

{

RouteRegister routeRegister = new RouteRegister(); // Новий реєстр

FillObjectFromInputs(routeRegister);

db.AddRouteRegister(routeRegister);

}

}

private void EditRecord()

{

if (dataGridView.SelectedRows.Count == 0)

{

MessageBox.Show("Оберіть запис для редагування.");

return;

}

if (radioButtonTransport.Checked)

{

Transport transport = (Transport)dataGridView.SelectedRows[0].DataBoundItem; // Вибраний транспорт

FillObjectFromInputs(transport); // Заповнити дані

db.UpdateTransport(transport);

}

else if (radioButtonRoute.Checked)

{

Route route = (Route)dataGridView.SelectedRows[0].DataBoundItem; // Вибраний маршрут

FillObjectFromInputs(route);

db.UpdateRoute(route);

}

else if (radioButtonRouteRegister.Checked)

{

RouteRegister routeRegister = (RouteRegister)dataGridView.SelectedRows[0].DataBoundItem; // Вибраний реєстр

FillObjectFromInputs(routeRegister);

db.UpdateRouteRegister(routeRegister);

}

}

private void DeleteRecord()

{

if (dataGridView.SelectedRows.Count == 0)

{

MessageBox.Show("Оберіть запис для видалення."); // Повідомлення

return;

}

if (radioButtonTransport.Checked)

{

Transport transport = (Transport)dataGridView.SelectedRows[0].DataBoundItem; // Вибраний транспорт

db.DeleteTransport(transport.TransportID); // Видалити з бази

}

else if (radioButtonRoute.Checked)

{

Route route = (Route)dataGridView.SelectedRows[0].DataBoundItem; // Вибраний маршрут

db.DeleteRoute(route.RouteID);

}

else if (radioButtonRouteRegister.Checked)

{

RouteRegister routeRegister = (RouteRegister)dataGridView.SelectedRows[0].DataBoundItem; // Вибраний реєстр

db.DeleteRouteRegister(routeRegister.RouteRegisterID);

}

}

private void FillObjectFromInputs<T>(T obj)

{

var type = typeof(T); // Тип об'єкта

var properties = type.GetProperties(); // Властивості

foreach (var prop in properties)

{

var control = panelInputs.Controls["input\_" + prop.Name]; // Контроль вводу

if (control == null)

continue; // Пропустити

if (control is TextBox textBox)

{

string value = textBox.Text; // Отримати текст

if (prop.PropertyType == typeof(int))

{

if (int.TryParse(value, out int intValue))

{

prop.SetValue(obj, intValue); // Встановити int

}

else

{

throw new Exception($"Невірний формат для {prop.Name}. Очікується ціле число.");

}

}

else if (prop.PropertyType == typeof(decimal))

{

if (decimal.TryParse(value, out decimal decimalValue))

{

prop.SetValue(obj, decimalValue); // Встановити decimal

}

else

{

throw new Exception($"Невірний формат для {prop.Name}. Очікується число.");

}

}

else if (prop.PropertyType == typeof(string))

{

prop.SetValue(obj, value); // Встановити string

}

}

else if (control is DateTimePicker timePicker)

{

if (prop.PropertyType == typeof(TimeSpan))

{

prop.SetValue(obj, timePicker.Value.TimeOfDay); // Встановити TimeSpan

}

}

}

}

private void ClearInputFields()

{

foreach (Control control in panelInputs.Controls)

{

if (control is TextBox textBox)

{

textBox.Clear(); // Очистити текстове поле

}

else if (control is DateTimePicker timePicker)

{

timePicker.Value = DateTime.Now; // Скинути час

}

}

}

private void DataGridView\_SelectionChanged(object sender, EventArgs e)

{

string mode = comboBoxMode.SelectedItem.ToString(); // Отримати режим

if ((mode == "Редагування" || mode == "Видалення") && dataGridView.SelectedRows.Count > 0)

{

LoadSelectedRecordIntoInputs(); // Завантажити запис

}

}

private void LoadSelectedRecordIntoInputs()

{

object selectedItem = dataGridView.SelectedRows[0].DataBoundItem; // Вибраний запис

var type = selectedItem.GetType(); // Тип запису

var properties = type.GetProperties(); // Властивості

foreach (var prop in properties)

{

var control = panelInputs.Controls["input\_" + prop.Name]; // Контроль вводу

if (control == null)

continue; // Пропустити

var value = prop.GetValue(selectedItem); // Значення властивості

if (control is TextBox textBox)

{

textBox.Text = value?.ToString(); // Встановити текст

}

else if (control is DateTimePicker timePicker)

{

if (value is TimeSpan timeSpan)

{

timePicker.Value = DateTime.Today.Add(timeSpan); // Встановити час

}

}

}

}

private void buttonAction\_Click\_1(object sender, EventArgs e)

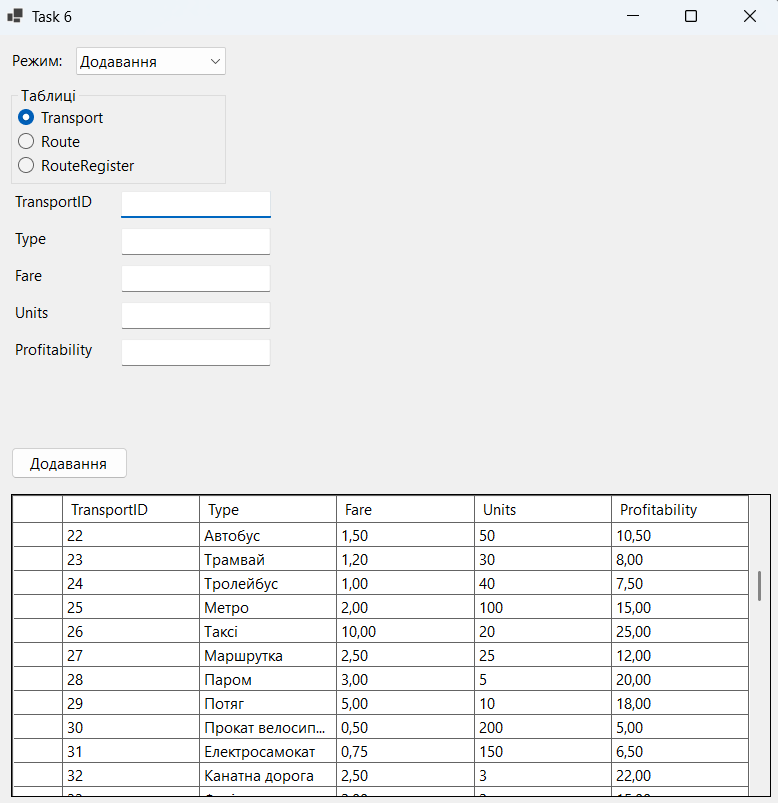
{

}

}

}

**Результат роботи програми:**



Изображение выглядит как текст, снимок экрана, число, Параллельный

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, Параллельный

Автоматически созданное описание

Изображение выглядит как текст, программное обеспечение, снимок экрана, Значок на компьютере

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, Параллельный

Автоматически созданное описание

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, программное обеспечение, снимок экрана, Значок на компьютере

Автоматически созданное описание

**Алгоритм побудови коду**

1. **Створення моделей**:
   * Реалізовані класи Transport, Route, RouteRegister, які відображають структуру відповідних таблиць бази даних.
2. **Робота з базою даних**:
   * Клас Database для доступу до MySQL.
   * Методи для отримання (Get), додавання (Add), редагування (Update) та видалення (Delete) записів.
3. **Форма користувача (MainForm)**:
   * Компоненти:
     + DataGridView для перегляду таблиці.
     + Панель із текстовими полями та іншими елементами для редагування даних.
     + Кнопка для виконання дій (Додавання, Редагування, Видалення).
   * Події:
     + ComboBox для вибору режиму роботи (Додавання, Редагування, Видалення).
     + RadioButton для вибору таблиці (Transport, Route, RouteRegister).
     + SelectionChanged для заповнення полів вводу при виборі запису з таблиці.
4. **Режими роботи**:
   * **Додавання**:
     + Заповнені поля додають новий запис у базу даних.
   * **Редагування**:
     + Дані вибраного запису редагуються на основі введених значень.
   * **Видалення**:
     + Видаляється вибраний запис.
5. **Динамічне оновлення інтерфейсу**:
   * Поля для вводу генеруються відповідно до структури вибраної таблиці.
   * Залежно від вибраного режиму змінюється доступність полів.

**Функціональність**

1. **Робота з таблицями**:
   * Підтримуються три таблиці бази даних: Transport, Route, RouteRegister.
   * Відображення у DataGridView з можливістю перегляду, редагування та видалення.
2. **Додавання записів**:
   * Користувач вводить дані у форму, натискає кнопку Додати, і запис додається до бази даних.
3. **Редагування записів**:
   * Вибирається запис у DataGridView, дані відображаються у полях вводу, редагуються та зберігаються після натискання кнопки Редагувати.
4. **Видалення записів**:
   * Вибирається запис у DataGridView, який видаляється після натискання кнопки Видалити.
5. **Автоматичне оновлення даних**:
   * Після кожної операції (додавання, редагування, видалення) дані оновлюються у таблиці.
6. **Інтерфейс**:
   * Інтуїтивно зрозумілий дизайн.
   * Поля вводу створюються відповідно до структури вибраної таблиці.

**Висновок**

1. **Результат**:
   * Створено універсальний застосунок для управління базою даних.
   * Всі операції над таблицями (CRUD) виконуються через інтерфейс.
2. **Переваги**:
   * Гнучкий інтерфейс: підтримка будь-якої структури таблиць бази даних.
   * Інтуїтивна навігація: чітке розділення режимів роботи.
3. **Використання**:
   * Застосунок підходить для управління транспортними маршрутами або іншими аналогічними структурами даних.
4. **Можливі покращення**:
   * Додати перевірку коректності введених даних.
   * Реалізувати пошук та сортування даних у таблицях.

Загалом, застосунок успішно виконує поставлені завдання та забезпечує базові операції роботи з даними.

**Завдання 7**

Створити застосунок з формою чи формами, який би містив таблиці з БД та інші діалогові елементи (текстовими полями, кнопками, списками та ін.), за допомогою яких можна було 6 одержати три статистичні (наприклад середня вартість виробу, вироби з максимальною чи мінімальною вартістю, середня довжина маршруту, кількість працівників певної кваліфікації, відсоток працівників певної кваліфікації від загальної кількості працівників та ін.) характеристики за вмістом кожної з таблиць БД. Перемикання між таблицями зробити за умов згідно до свого варіанту минулого завдання

**Текст програми:**

**Папка Models:**

***Route.cs:***

namespace Task\_7.Models

{

public class Route

{

public int RouteID { get; set; }

public string StartPoint { get; set; }

public string EndPoint { get; set; }

public int Stops { get; set; }

public decimal Length { get; set; }

public int TravelTime { get; set; }

}

}

***RouteRegister.cs:***

using System;

namespace Task\_7.Models

{

public class RouteRegister

{

public int RouteRegisterID { get; set; }

public int TransportID { get; set; }

public int RouteID { get; set; }

public string Direction { get; set; }

public TimeSpan DepartureTime { get; set; }

public TimeSpan ArrivalTime { get; set; }

}

}

***Transport.cs:***

namespace Task\_7.Models

{

public class Transport

{

public int TransportID { get; set; }

public string Type { get; set; }

public decimal Fare { get; set; }

public int Units { get; set; }

public decimal Profitability { get; set; }

}

}

***Database.cs:***

using System;

using System.Collections.Generic;

using MySql.Data.MySqlClient;

using Task\_7.Models;

namespace Task\_7

{

public class Database

{

// Підключення

private string connectionString = "server=localhost;user=root;password=mnc5UbTRn\_nZ+M:;database=CityTransportDB;";

// Transport Methods

public List<Transport> GetTransports()

{

List<Transport> transports = new List<Transport>();

using (MySqlConnection conn = new MySqlConnection(connectionString))

{

conn.Open();

string query = "SELECT \* FROM Transport;";

MySqlCommand cmd = new MySqlCommand(query, conn); // Створити команду

MySqlDataReader reader = cmd.ExecuteReader(); // Виконати читання

while (reader.Read())

{

Transport transport = new Transport

{

TransportID = reader.GetInt32("TransportID"),

Type = reader.GetString("Type"),

Fare = reader.GetDecimal("Fare"),

Units = reader.GetInt32("Units"),

Profitability = reader.GetDecimal("Profitability")

};

transports.Add(transport); // Додати транспорт

}

}

return transports;

}

public decimal GetAverageFare()

{

decimal averageFare = 0;

using (MySqlConnection conn = new MySqlConnection(connectionString))

{

conn.Open();

string query = "SELECT AVG(Fare) FROM Transport;";

MySqlCommand cmd = new MySqlCommand(query, conn);

var result = cmd.ExecuteScalar();

if (result != DBNull.Value)

averageFare = Convert.ToDecimal(result); // Обробка результату

}

return averageFare; // Повернути середнє значення

}

public Transport GetTransportWithMaxFare()

{

Transport transport = null;

using (MySqlConnection conn = new MySqlConnection(connectionString))

{

conn.Open();

string query = "SELECT \* FROM Transport WHERE Fare = (SELECT MAX(Fare) FROM Transport);";

MySqlCommand cmd = new MySqlCommand(query, conn);

MySqlDataReader reader = cmd.ExecuteReader();

if (reader.Read())

{

transport = new Transport

{

TransportID = reader.GetInt32("TransportID"),

Type = reader.GetString("Type"),

Fare = reader.GetDecimal("Fare"),

Units = reader.GetInt32("Units"),

Profitability = reader.GetDecimal("Profitability")

};

}

}

return transport; // Повернути транспорт

}

public Transport GetTransportWithMinFare()

{

Transport transport = null;

using (MySqlConnection conn = new MySqlConnection(connectionString))

{

conn.Open();

string query = "SELECT \* FROM Transport WHERE Fare = (SELECT MIN(Fare) FROM Transport);";

MySqlCommand cmd = new MySqlCommand(query, conn);

MySqlDataReader reader = cmd.ExecuteReader();

if (reader.Read())

{

transport = new Transport

{

TransportID = reader.GetInt32("TransportID"),

Type = reader.GetString("Type"),

Fare = reader.GetDecimal("Fare"),

Units = reader.GetInt32("Units"),

Profitability = reader.GetDecimal("Profitability")

};

}

}

return transport; // Повернути транспорт

}

// Route Methods

public List<Route> GetRoutes()

{

List<Route> routes = new List<Route>();

using (MySqlConnection conn = new MySqlConnection(connectionString))

{

conn.Open();

string query = "SELECT \* FROM Route;";

MySqlCommand cmd = new MySqlCommand(query, conn); // Створити команду

MySqlDataReader reader = cmd.ExecuteReader(); // Виконати читання

while (reader.Read())

{

Route route = new Route

{

RouteID = reader.GetInt32("RouteID"),

StartPoint = reader.GetString("StartPoint"),

EndPoint = reader.GetString("EndPoint"),

Stops = reader.GetInt32("Stops"),

Length = reader.GetDecimal("Length"),

TravelTime = reader.GetInt32("TravelTime")

};

routes.Add(route); // Додати маршрут

}

}

return routes;

}

public decimal GetAverageRouteLength()

{

decimal averageLength = 0;

using (MySqlConnection conn = new MySqlConnection(connectionString))

{

conn.Open();

string query = "SELECT AVG(Length) FROM Route;";

MySqlCommand cmd = new MySqlCommand(query, conn);

var result = cmd.ExecuteScalar();

if (result != DBNull.Value)

averageLength = Convert.ToDecimal(result); // Обробка результату

}

return averageLength; // Повернути середню довжину

}

public Route GetRouteWithMaxLength()

{

Route route = null;

using (MySqlConnection conn = new MySqlConnection(connectionString))

{

conn.Open();

string query = "SELECT \* FROM Route WHERE Length = (SELECT MAX(Length) FROM Route);";

MySqlCommand cmd = new MySqlCommand(query, conn);

MySqlDataReader reader = cmd.ExecuteReader();

if (reader.Read())

{

route = new Route

{

RouteID = reader.GetInt32("RouteID"),

StartPoint = reader.GetString("StartPoint"),

EndPoint = reader.GetString("EndPoint"),

Stops = reader.GetInt32("Stops"),

Length = reader.GetDecimal("Length"),

TravelTime = reader.GetInt32("TravelTime")

};

}

}

return route; // Повернути маршрут

}

public Route GetRouteWithMinLength()

{

Route route = null;

using (MySqlConnection conn = new MySqlConnection(connectionString))

{

conn.Open();

string query = "SELECT \* FROM Route WHERE Length = (SELECT MIN(Length) FROM Route);";

MySqlCommand cmd = new MySqlCommand(query, conn);

MySqlDataReader reader = cmd.ExecuteReader();

if (reader.Read())

{

route = new Route

{

RouteID = reader.GetInt32("RouteID"),

StartPoint = reader.GetString("StartPoint"),

EndPoint = reader.GetString("EndPoint"),

Stops = reader.GetInt32("Stops"),

Length = reader.GetDecimal("Length"),

TravelTime = reader.GetInt32("TravelTime")

};

}

}

return route; // Повернути маршрут

}

// RouteRegister Methods

public List<RouteRegister> GetRouteRegisters()

{

List<RouteRegister> routeRegisters = new List<RouteRegister>();

using (MySqlConnection conn = new MySqlConnection(connectionString))

{

conn.Open(); // Відкрити з'єднання

string query = "SELECT \* FROM RouteRegister;";

MySqlCommand cmd = new MySqlCommand(query, conn); // Створити команду

MySqlDataReader reader = cmd.ExecuteReader(); // Виконати читання

while (reader.Read())

{

RouteRegister routeRegister = new RouteRegister

{

RouteRegisterID = reader.GetInt32("RouteRegisterID"),

TransportID = reader.GetInt32("TransportID"),

RouteID = reader.GetInt32("RouteID"),

Direction = reader.GetString("Direction"),

DepartureTime = reader.GetTimeSpan("DepartureTime"),

ArrivalTime = reader.GetTimeSpan("ArrivalTime")

};

routeRegisters.Add(routeRegister); // Додати реєстр

}

}

return routeRegisters;

}

public int GetTotalTrips()

{

int totalTrips = 0;

using (MySqlConnection conn = new MySqlConnection(connectionString))

{

conn.Open();

string query = "SELECT COUNT(\*) FROM RouteRegister;";

MySqlCommand cmd = new MySqlCommand(query, conn);

totalTrips = Convert.ToInt32(cmd.ExecuteScalar());

}

return totalTrips; // Повернути загальну кількість поїздок

}

public TimeSpan GetEarliestDepartureTime()

{

TimeSpan earliestTime = TimeSpan.Zero;

using (MySqlConnection conn = new MySqlConnection(connectionString))

{

conn.Open();

string query = "SELECT MIN(DepartureTime) FROM RouteRegister;";

MySqlCommand cmd = new MySqlCommand(query, conn);

var result = cmd.ExecuteScalar();

if (result != DBNull.Value)

earliestTime = (TimeSpan)result;

}

return earliestTime; // Повернути найраніший час відправлення

}

public TimeSpan GetLatestArrivalTime()

{

TimeSpan latestTime = TimeSpan.Zero;

using (MySqlConnection conn = new MySqlConnection(connectionString))

{

conn.Open();

string query = "SELECT MAX(ArrivalTime) FROM RouteRegister;";

MySqlCommand cmd = new MySqlCommand(query, conn);

var result = cmd.ExecuteScalar();

if (result != DBNull.Value)

latestTime = (TimeSpan)result; // Обробка результату

}

return latestTime; // Повернути найпізніший час прибуття

}

}

}

***MainForm.cs:***

using System;

using System.Collections.Generic;

using System.Windows.Forms;

using Task\_7.Models;

using System.Globalization;

namespace Task\_7

{

public partial class MainForm : Form

{

private Database db; // Підключення до бази

private List<Transport> transports; // Транспортні засоби

private List<Route> routes; // Маршрути

private List<RouteRegister> routeRegisters; // Реєстри маршрутів

public MainForm()

{

InitializeComponent();

db = new Database();

// Підписка на події

radioButtonTransport.CheckedChanged += RadioButton\_CheckedChanged;

radioButtonRoute.CheckedChanged += RadioButton\_CheckedChanged;

radioButtonRouteRegister.CheckedChanged += RadioButton\_CheckedChanged;

comboBoxMode.SelectedIndexChanged += ComboBoxMode\_SelectedIndexChanged;

buttonCalculate.Click += ButtonCalculate\_Click;

// Початкові налаштування

comboBoxMode.SelectedIndex = 0;

LoadData(); // Завантаження даних

InitializeStatisticsComboBox(); // Налаштування статистики

UpdateView(); // Оновлення інтерфейсу

}

// Завантаження даних з бази

private void LoadData()

{

try

{

transports = db.GetTransports();

routes = db.GetRoutes();

routeRegisters = db.GetRouteRegisters();

}

catch (Exception ex)

{

MessageBox.Show($"Помилка при завантаженні даних: {ex.Message}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

// Обробка зміни радіокнопок

private void RadioButton\_CheckedChanged(object sender, EventArgs e)

{

InitializeStatisticsComboBox();

UpdateView();

}

// Обробка зміни режиму в комбобоксі

private void ComboBoxMode\_SelectedIndexChanged(object sender, EventArgs e)

{

InitializeStatisticsComboBox();

UpdateView();

}

// Налаштування комбобоксу статистики

private void InitializeStatisticsComboBox()

{

comboBoxStatistics.Items.Clear();

if (comboBoxMode.SelectedItem?.ToString() == "Статистика")

{

// Показати елементи статистики

labelStatistics.Visible = true;

comboBoxStatistics.Visible = true;

buttonCalculate.Visible = true;

labelResult.Visible = true;

// Додати варіанти залежно від вибору категорії

if (radioButtonTransport.Checked)

{

comboBoxStatistics.Items.AddRange(new string[]

{

"Середня вартість проїзду",

"Транспорт з максимальною вартістю",

"Транспорт з мінімальною вартістю"

});

}

else if (radioButtonRoute.Checked)

{

comboBoxStatistics.Items.AddRange(new string[]

{

"Середня довжина маршруту",

"Маршрут з максимальною довжиною",

"Маршрут з мінімальною довжиною"

});

}

else if (radioButtonRouteRegister.Checked)

{

comboBoxStatistics.Items.AddRange(new string[]

{

"Загальна кількість рейсів",

"Найраніший час відправлення",

"Найпізніший час прибуття"

});

}

if (comboBoxStatistics.Items.Count > 0)

comboBoxStatistics.SelectedIndex = 0;

}

else

{

// Сховати елементи статистики

labelStatistics.Visible = false;

comboBoxStatistics.Visible = false;

buttonCalculate.Visible = false;

labelResult.Visible = false;

}

}

// Оновлення відображення форми

private void UpdateView()

{

if (comboBoxMode.SelectedItem == null) return;

string mode = comboBoxMode.SelectedItem.ToString();

if (mode == "Перегляд даних")

{

dataGridView.Visible = true;

labelResult.Visible = false;

// Відобразити відповідні дані

if (radioButtonTransport.Checked)

dataGridView.DataSource = transports;

else if (radioButtonRoute.Checked)

dataGridView.DataSource = routes;

else if (radioButtonRouteRegister.Checked)

dataGridView.DataSource = routeRegisters;

}

else if (mode == "Статистика")

{

dataGridView.Visible = false;

labelResult.Visible = true;

labelResult.Text = "Результат:";

}

}

// Обробка натискання кнопки розрахунку

private void ButtonCalculate\_Click(object sender, EventArgs e)

{

if (comboBoxStatistics.SelectedItem == null)

{

MessageBox.Show("Будь ласка, оберіть статистичну характеристику.");

return;

}

string selectedStatistic = comboBoxStatistics.SelectedItem.ToString();

// Виклик відповідного методу обробки

if (radioButtonTransport.Checked)

ProcessTransportStatistics(selectedStatistic);

else if (radioButtonRoute.Checked)

ProcessRouteStatistics(selectedStatistic);

else if (radioButtonRouteRegister.Checked)

ProcessRouteRegisterStatistics(selectedStatistic);

}

// Статистика для транспорту

private void ProcessTransportStatistics(string statistic)

{

switch (statistic)

{

case "Середня вартість проїзду":

decimal avgFare = db.GetAverageFare();

labelResult.Text = $"Середня вартість проїзду: {avgFare.ToString("C", CultureInfo.GetCultureInfo("uk-UA"))}";

break;

case "Транспорт з максимальною вартістю":

var maxTransport = db.GetTransportWithMaxFare();

labelResult.Text = maxTransport != null

? $"Макс. транспорт: {maxTransport.Type} - {maxTransport.Fare.ToString("C", CultureInfo.GetCultureInfo("uk-UA"))}"

: "Дані відсутні.";

break;

case "Транспорт з мінімальною вартістю":

var minTransport = db.GetTransportWithMinFare();

labelResult.Text = minTransport != null

? $"Мін. транспорт: {minTransport.Type} - {minTransport.Fare.ToString("C", CultureInfo.GetCultureInfo("uk-UA"))}"

: "Дані відсутні.";

break;

}

}

// Статистика для маршрутів

private void ProcessRouteStatistics(string statistic)

{

switch (statistic)

{

case "Середня довжина маршруту":

decimal avgLength = db.GetAverageRouteLength();

labelResult.Text = $"Середня довжина маршруту: {avgLength} км";

break;

case "Маршрут з максимальною довжиною":

var maxRoute = db.GetRouteWithMaxLength();

labelResult.Text = maxRoute != null

? $"Макс. маршрут: {maxRoute.Length} км ({maxRoute.StartPoint} - {maxRoute.EndPoint})"

: "Дані відсутні.";

break;

case "Маршрут з мінімальною довжиною":

var minRoute = db.GetRouteWithMinLength();

labelResult.Text = minRoute != null

? $"Мін. маршрут: {minRoute.Length} км ({minRoute.StartPoint} - {minRoute.EndPoint})"

: "Дані відсутні.";

break;

}

}

// Статистика для реєстрів маршрутів

private void ProcessRouteRegisterStatistics(string statistic)

{

switch (statistic)

{

case "Загальна кількість рейсів":

int totalTrips = db.GetTotalTrips();

labelResult.Text = $"Загальна кількість рейсів: {totalTrips}";

break;

case "Найраніший час відправлення":

TimeSpan earliest = db.GetEarliestDepartureTime();

labelResult.Text = $"Найраніший час відправлення: {earliest}";

break;

case "Найпізніший час прибуття":

TimeSpan latest = db.GetLatestArrivalTime();

labelResult.Text = $"Найпізніший час прибуття: {latest}";

break;

}

}

private void labelStatistics\_Click(object sender, EventArgs e)

{

}

}

}

**Результат роботи програми:**

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

**Алгоритм побудови коду**

1. **Створення моделей**:
   * Реалізовано класи Transport, Route, RouteRegister, які відповідають структурам таблиць у базі даних.
2. **Робота з базою даних**:
   * Клас Database для доступу до MySQL.
   * Методи для отримання даних (GetTransports, GetRoutes, GetRouteRegisters) та статистики (наприклад, GetAverageFare, GetRouteWithMaxLength).
3. **Реалізація форми користувача**:
   * **Компоненти**:
     + DataGridView для перегляду таблиць.
     + ComboBox для вибору режиму (Перегляд даних, Статистика).
     + Поле результату для виводу статистичних даних.
     + Кнопка для обчислення статистики.
   * **Події**:
     + Обробка зміни радіокнопок (RadioButton) для вибору таблиці.
     + Зміна комбобоксу (ComboBox) для перемикання між режимами.
     + Натискання кнопки розрахунку статистики.
4. **Статистика для таблиць**:
   * Для кожної таблиці реалізовано методи для обчислення статистичних характеристик.
   * Вибір статистики здійснюється через ComboBox.
5. **Динамічне оновлення інтерфейсу**:
   * У режимі Статистика відображаються відповідні опції для розрахунків.
   * У режимі Перегляд даних відображається таблиця.

**Функціональність**

1. **Робота з таблицями**:
   * Відображення даних у DataGridView із таблиць Transport, Route, RouteRegister.
2. **Статистичні розрахунки**:
   * Для таблиці Transport:
     + Середня вартість проїзду.
     + Транспорт із мінімальною вартістю.
     + Транспорт із максимальною вартістю.
   * Для таблиці Route:
     + Середня довжина маршруту.
     + Маршрут із мінімальною довжиною.
     + Маршрут із максимальною довжиною.
   * Для таблиці RouteRegister:
     + Загальна кількість рейсів.
     + Найраніший час відправлення.
     + Найпізніший час прибуття.
3. **Інтерфейс користувача**:
   * Простий і зручний.
   * Дозволяє швидко перемикатися між таблицями та режимами.

**Висновок**

1. **Результат**:
   * Створено застосунок, який дозволяє як переглядати дані, так і обчислювати статистичні характеристики.
2. **Переваги**:
   * Гнучкий інтерфейс для роботи з таблицями.
   * Чітке розділення режимів роботи (Перегляд даних і Статистика).
3. **Можливі покращення**:
   * Реалізувати графічну візуалізацію статистики.
   * Додати підтримку фільтрації даних.

Застосунок відповідає поставленим вимогам, дозволяє виконувати необхідні операції та забезпечує зручний інтерфейс для роботи з даними бази.

**Завдання 8\***

Створити застосунок умовної соціальної мережі за завданнями лабораторної роботи № 8. Але в якості середовища зберігання параметрів акаунтів користувачів використовувати файли MDF (Microsoft database file) вбудованого у Visual Studio сервісу управління базами даних.

**Текст програми:**

***SQL-запит:***

***Створення таблиці Users:***

CREATE TABLE [dbo].[Users] (

[UserID] INT IDENTITY (1, 1) NOT NULL,

[Username] NVARCHAR(50) NOT NULL,

[Password] NVARCHAR(255) NOT NULL,

[Email] NVARCHAR(100) NOT NULL,

[Phone] NVARCHAR(20) NOT NULL,

CONSTRAINT [PK\_Users] PRIMARY KEY CLUSTERED ([UserID] ASC),

CONSTRAINT [UQ\_Users\_Username] UNIQUE ([Username]),

CONSTRAINT [UQ\_Users\_Email] UNIQUE ([Email]),

CONSTRAINT [UQ\_Users\_Phone] UNIQUE ([Phone])

);

***Створення таблиці Messages:***

CREATE TABLE [dbo].[Messages] (

[MessageID] INT IDENTITY (1, 1) NOT NULL,

[SenderUserID] INT NOT NULL,

[RecipientUserID] INT NOT NULL,

[Theme] NVARCHAR(100) NOT NULL,

[Text] NVARCHAR(MAX) NOT NULL,

[Date] DATETIME NOT NULL,

CONSTRAINT [PK\_Messages] PRIMARY KEY CLUSTERED ([MessageID] ASC),

CONSTRAINT [FK\_Messages\_SenderUser] FOREIGN KEY ([SenderUserID]) REFERENCES [dbo].[Users] ([UserID]),

CONSTRAINT [FK\_Messages\_RecipientUser] FOREIGN KEY ([RecipientUserID]) REFERENCES [dbo].[Users] ([UserID])

);

***RegisterForm.cs:***

using System;

using System.Data.SqlClient;

using System.Drawing;

using System.Drawing.Drawing2D;

using System.Text.RegularExpressions;

using System.Windows.Forms;

namespace Lab\_\_12

{

public partial class RegisterForm : Form

{

public RegisterForm()

{

InitializeComponent();

StyleComponents();

ClearErrorLabels();

}

private void StyleComponents()

{

// Кнопка реєстрації

registerButton.BackColor = Color.MediumSeaGreen;

registerButton.ForeColor = Color.White;

registerButton.FlatStyle = FlatStyle.Flat;

registerButton.FlatAppearance.BorderSize = 0;

RoundButtonEdges(registerButton, 15);

// Кнопка переходу до логіну

loginButton.BackColor = Color.Transparent;

loginButton.ForeColor = Color.MediumSeaGreen;

loginButton.FlatStyle = FlatStyle.Flat;

loginButton.FlatAppearance.BorderSize = 0;

RoundButtonEdges(loginButton, 15);

// Видалення рамок текстових полів та додавання підкреслення

nameTextBox.BorderStyle = BorderStyle.None;

emailTextBox.BorderStyle = BorderStyle.None;

phoneTextBox.BorderStyle = BorderStyle.None;

passwordTextBox.BorderStyle = BorderStyle.None;

confirmPasswordTextBox.BorderStyle = BorderStyle.None;

// Задання кольору для панелей під текстовими полями

namePanel.BackColor = Color.LightGray;

emailPanel.BackColor = Color.LightGray;

phonePanel.BackColor = Color.LightGray;

passwordPanel.BackColor = Color.LightGray;

confirmPasswordPanel.BackColor = Color.LightGray;

}

private void RoundButtonEdges(Button button, int radius)

{

GraphicsPath path = new GraphicsPath();

path.AddArc(0, 0, radius, radius, 180, 90);

path.AddArc(button.Width - radius, 0, radius, radius, 270, 90);

path.AddArc(button.Width - radius, button.Height - radius, radius, radius, 0, 90);

path.AddArc(0, button.Height - radius, radius, radius, 90, 90);

path.CloseAllFigures();

button.Region = new Region(path);

}

private void ClearErrorLabels()

{

nameErrorLabel.Text = "";

emailErrorLabel.Text = "";

phoneErrorLabel.Text = "";

passwordErrorLabel.Text = "";

confirmPasswordErrorLabel.Text = "";

}

private void RegisterButton\_Click(object sender, EventArgs e)

{

ClearErrorLabels();

string userName = nameTextBox.Text.Trim();

string email = emailTextBox.Text.Trim();

string phone = phoneTextBox.Text.Trim();

string password = passwordTextBox.Text;

string confirmPassword = confirmPasswordTextBox.Text;

bool isValid = ValidateFields(userName, email, phone, password, confirmPassword);

if (isValid)

{

bool isUnique = true;

if (IsUsernameTaken(userName))

{

nameErrorLabel.Text = "Ім'я користувача вже зайняте.";

isUnique = false;

}

if (IsEmailTaken(email))

{

emailErrorLabel.Text = "Електронна пошта вже використовується.";

isUnique = false;

}

if (IsPhoneTaken(phone))

{

phoneErrorLabel.Text = "Номер телефону вже використовується.";

isUnique = false;

}

if (!isUnique)

{

return;

}

SaveUserData(userName, email, phone, password);

MessageBox.Show("Реєстрація успішна!", "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Information);

ClearFields();

LoginForm loginForm = new LoginForm();

loginForm.Show();

this.Hide();

}

}

private bool ValidateFields(string userName, string email, string phone, string password, string confirmPassword)

{

bool isValid = true;

if (!Regex.IsMatch(userName, @"^[a-zA-Z0-9]{1,15}$"))

{

nameErrorLabel.Text = "Неправильне ім'я. Макс. 15 символів, латиниця та цифри.";

isValid = false;

}

if (!Regex.IsMatch(email, @"^[^@\s]+@[^@\s]+**\.**[^@\s]+$"))

{

emailErrorLabel.Text = "Невірний формат електронної пошти.";

isValid = false;

}

if (!Regex.IsMatch(phone, @"^**\+**?\d{10,18}$"))

{

phoneErrorLabel.Text = "Номер телефону має містити від 10 до 18 цифр.";

isValid = false;

}

if (!Regex.IsMatch(password, @"^(?=.\*[a-zA-Z])(?=.\*[0-9])(?=.\*[!@#$%^&\*]).{12,}$"))

{

passwordErrorLabel.Text = "Слабкий пароль. Мін. 12 символів, цифри та спеціальні символи.";

isValid = false;

}

if (password != confirmPassword)

{

confirmPasswordErrorLabel.Text = "Паролі не співпадають.";

isValid = false;

}

return isValid;

}

private bool IsUsernameTaken(string userName)

{

string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\nemof\OneDrive\Изображения\Lab\_\_12\Lab\_\_12\Lab\_\_12\SocialNetwork.mdf;Integrated Security=True";

string query = "SELECT COUNT(\*) FROM Users WHERE Username = @Username";

using (SqlConnection conn = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@Username", userName);

conn.Open();

int count = (int)cmd.ExecuteScalar();

return count > 0;

}

}

private bool IsEmailTaken(string email)

{

string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\nemof\OneDrive\Изображения\Lab\_\_12\Lab\_\_12\Lab\_\_12\SocialNetwork.mdf;Integrated Security=True";

string query = "SELECT COUNT(\*) FROM Users WHERE Email = @Email";

using (SqlConnection conn = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@Email", email);

conn.Open();

int count = (int)cmd.ExecuteScalar();

return count > 0;

}

}

private bool IsPhoneTaken(string phone)

{

string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\nemof\OneDrive\Изображения\Lab\_\_12\Lab\_\_12\Lab\_\_12\SocialNetwork.mdf;Integrated Security=True";

string query = "SELECT COUNT(\*) FROM Users WHERE Phone = @Phone";

using (SqlConnection conn = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@Phone", phone);

conn.Open();

int count = (int)cmd.ExecuteScalar();

return count > 0;

}

}

private void ClearFields()

{

nameTextBox.Clear();

emailTextBox.Clear();

phoneTextBox.Clear();

passwordTextBox.Clear();

confirmPasswordTextBox.Clear();

ClearErrorLabels();

}

private void SaveUserData(string userName, string email, string phone, string password)

{

string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\nemof\OneDrive\Изображения\Lab\_\_12\Lab\_\_12\Lab\_\_12\SocialNetwork.mdf;Integrated Security=True";

string query = "INSERT INTO Users (Username, Password, Email, Phone) VALUES (@Username, @Password, @Email, @Phone)";

using (SqlConnection conn = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@Username", userName);

cmd.Parameters.AddWithValue("@Password", password);

cmd.Parameters.AddWithValue("@Email", email);

cmd.Parameters.AddWithValue("@Phone", phone);

conn.Open();

cmd.ExecuteNonQuery();

}

}

private void LoginButton\_Click(object sender, EventArgs e)

{

LoginForm loginForm = new LoginForm();

loginForm.Show();

this.Close();

}

}

}

***LoginForm.cs:***

using System;

using System.Data.SqlClient;

using System.Drawing;

using System.Drawing.Drawing2D;

using System.Windows.Forms;

namespace Lab\_\_12

{

public partial class LoginForm : Form

{

public LoginForm()

{

InitializeComponent();

errorLabel.Visible = false;

StyleComponents();

}

private void StyleComponents()

{

loginButton.BackColor = Color.MediumSeaGreen;

loginButton.ForeColor = Color.White;

loginButton.FlatStyle = FlatStyle.Flat;

loginButton.FlatAppearance.BorderSize = 0;

RoundButtonEdges(loginButton, 15);

registerButton.BackColor = Color.Transparent;

registerButton.ForeColor = Color.MediumSeaGreen;

registerButton.FlatStyle = FlatStyle.Flat;

registerButton.FlatAppearance.BorderSize = 0;

RoundButtonEdges(registerButton, 15);

usernameTextBox.BorderStyle = BorderStyle.None;

passwordTextBox.BorderStyle = BorderStyle.None;

usernamePanel.BackColor = Color.LightGray;

passwordPanel.BackColor = Color.LightGray;

}

private void RoundButtonEdges(Button button, int radius)

{

GraphicsPath path = new GraphicsPath();

path.AddArc(0, 0, radius, radius, 180, 90);

path.AddArc(button.Width - radius, 0, radius, radius, 270, 90);

path.AddArc(button.Width - radius, button.Height - radius, radius, radius, 0, 90);

path.AddArc(0, button.Height - radius, radius, radius, 90, 90);

path.CloseAllFigures();

button.Region = new Region(path);

}

private void LoginButton\_Click(object sender, EventArgs e)

{

string username = usernameTextBox.Text.Trim();

string password = passwordTextBox.Text;

if (ValidateCredentials(username, password))

{

ProfileForm profileForm = new ProfileForm(username);

profileForm.Show();

this.Hide();

}

else

{

ShowErrorMessage("Невірний логін\nабо пароль");

}

}

private bool ValidateCredentials(string username, string password)

{

string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\nemof\OneDrive\Изображения\Lab\_\_12\Lab\_\_12\Lab\_\_12\SocialNetwork.mdf;Integrated Security=True";

string query = "SELECT COUNT(\*) FROM Users WHERE Username = @Username AND Password = @Password";

using (SqlConnection conn = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@Username", username);

cmd.Parameters.AddWithValue("@Password", password);

conn.Open();

int count = (int)cmd.ExecuteScalar();

return count > 0;

}

}

private void ShowErrorMessage(string message)

{

errorLabel.Text = message;

errorLabel.Visible = true;

}

private void RegisterButton\_Click(object sender, EventArgs e)

{

RegisterForm registerForm = new RegisterForm();

registerForm.Show();

this.Hide();

}

}

}

***ProfileForm.cs:***

using System;

using System.Data.SqlClient;

using System.Drawing;

using System.Drawing.Drawing2D;

using System.Windows.Forms;

namespace Lab\_\_12

{

public partial class ProfileForm : Form

{

private string username;

public ProfileForm(string username)

{

InitializeComponent();

this.username = username;

LoadUserData();

StyleComponents();

}

private void LoadUserData()

{

string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\nemof\OneDrive\Изображения\Lab\_\_12\Lab\_\_12\Lab\_\_12\SocialNetwork.mdf;Integrated Security=True";

string query = "SELECT Email, Phone FROM Users WHERE Username = @Username";

using (SqlConnection conn = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@Username", username);

conn.Open();

SqlDataReader reader = cmd.ExecuteReader();

if (reader.Read())

{

nameLabel.Text = username;

emailLabel.Text = reader["Email"].ToString();

phoneLabel.Text = reader["Phone"].ToString();

}

}

}

private void StyleComponents()

{

RoundButtonEdges(editNameButton, 15);

RoundButtonEdges(editEmailButton, 15);

RoundButtonEdges(editPhoneButton, 15);

RoundButtonEdges(editPasswordButton, 15);

RoundButtonEdges(sendMessageButton, 15);

RoundButtonEdges(receiveMessageButton, 15);

RoundButtonEdges(logoutButton, 15);

sendMessageButton.BackColor = Color.MediumSeaGreen;

sendMessageButton.ForeColor = Color.White;

sendMessageButton.FlatStyle = FlatStyle.Flat;

sendMessageButton.FlatAppearance.BorderSize = 0;

Button[] buttons = { editNameButton, editEmailButton, editPhoneButton, editPasswordButton, receiveMessageButton, logoutButton };

foreach (Button btn in buttons)

{

btn.FlatStyle = FlatStyle.Flat;

btn.FlatAppearance.BorderSize = 0;

}

}

private void RoundButtonEdges(Button button, int radius)

{

GraphicsPath path = new GraphicsPath();

path.AddArc(0, 0, radius, radius, 180, 90);

path.AddArc(button.Width - radius, 0, radius, radius, 270, 90);

path.AddArc(button.Width - radius, button.Height - radius, radius, radius, 0, 90);

path.AddArc(0, button.Height - radius, radius, radius, 90, 90);

path.CloseAllFigures();

button.Region = new Region(path);

}

private void LogoutButton\_Click(object sender, EventArgs e)

{

new LoginForm().Show();

this.Close();

}

private void EditNameButton\_Click(object sender, EventArgs e)

{

EditNameForm editNameForm = new EditNameForm(username);

if (editNameForm.ShowDialog() == DialogResult.OK)

{

username = editNameForm.NewUsername;

LoadUserData();

}

}

private void EditEmailButton\_Click(object sender, EventArgs e)

{

new EditEmailForm(username).ShowDialog();

LoadUserData();

}

private void EditPhoneButton\_Click(object sender, EventArgs e)

{

new EditPhoneForm(username).ShowDialog();

LoadUserData();

}

private void EditPasswordButton\_Click(object sender, EventArgs e)

{

new EditPasswordForm(username).ShowDialog();

}

private void SendMessageButton\_Click(object sender, EventArgs e)

{

new MessageSendForm(username).ShowDialog();

}

private void ReceiveMessageButton\_Click(object sender, EventArgs e)

{

new MessageReceiveForm(username).ShowDialog();

}

}

}

***EditPhoneForm.cs:***

using System;

using System.Data.SqlClient;

using System.Text.RegularExpressions;

using System.Windows.Forms;

namespace Lab\_\_12

{

public partial class EditPhoneForm : Form

{

private string username;

public EditPhoneForm(string username)

{

InitializeComponent();

this.username = username;

LoadCurrentPhone();

}

private void LoadCurrentPhone()

{

string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\nemof\OneDrive\Изображения\Lab\_\_12\Lab\_\_12\Lab\_\_12\SocialNetwork.mdf;Integrated Security=True";

string query = "SELECT Phone FROM Users WHERE Username = @Username";

using (SqlConnection conn = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@Username", username);

conn.Open();

phoneTextBox.Text = cmd.ExecuteScalar()?.ToString();

}

}

private bool IsPhoneTaken(string phone)

{

string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\nemof\OneDrive\Изображения\Lab\_\_12\Lab\_\_12\Lab\_\_12\SocialNetwork.mdf;Integrated Security=True";

string query = "SELECT COUNT(\*) FROM Users WHERE Phone = @Phone AND Username != @Username";

using (SqlConnection conn = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@Phone", phone);

cmd.Parameters.AddWithValue("@Username", username);

conn.Open();

int count = (int)cmd.ExecuteScalar();

return count > 0;

}

}

private void SaveButton\_Click(object sender, EventArgs e)

{

string newPhone = phoneTextBox.Text.Trim();

if (!Regex.IsMatch(newPhone, @"^**\+**?\d{10,15}$"))

{

MessageBox.Show("Некоректний формат номера телефону.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (IsPhoneTaken(newPhone))

{

MessageBox.Show("Цей номер телефону вже використовується іншим користувачем.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\nemof\OneDrive\Изображения\Lab\_\_12\Lab\_\_12\Lab\_\_12\SocialNetwork.mdf;Integrated Security=True";

string query = "UPDATE Users SET Phone = @Phone WHERE Username = @Username";

using (SqlConnection conn = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@Phone", newPhone);

cmd.Parameters.AddWithValue("@Username", username);

conn.Open();

cmd.ExecuteNonQuery();

MessageBox.Show("Номер телефону успішно оновлено!", "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.Close();

}

}

}

}

***EditPasswordForm.cs:***

using System;

using System.Data.SqlClient;

using System.Text.RegularExpressions;

using System.Windows.Forms;

namespace Lab\_\_12

{

public partial class EditPasswordForm : Form

{

private string username;

public EditPasswordForm(string username)

{

InitializeComponent();

this.username = username;

}

private void SaveButton\_Click(object sender, EventArgs e)

{

string currentPassword = currentPasswordTextBox.Text;

string newPassword = newPasswordTextBox.Text;

string confirmPassword = confirmPasswordTextBox.Text;

string errorMessage = ValidateFields(currentPassword, newPassword, confirmPassword);

if (string.IsNullOrEmpty(errorMessage))

{

if (UpdatePasswordInDatabase(username, currentPassword, newPassword))

{

MessageBox.Show("Пароль успішно оновлено!", "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.Close();

}

else

{

MessageBox.Show("Невірний поточний пароль.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

else

{

MessageBox.Show(errorMessage, "Помилка валідації", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private string ValidateFields(string currentPassword, string newPassword, string confirmPassword)

{

string error = "";

if (string.IsNullOrEmpty(currentPassword))

error += "Введіть поточний пароль.\n";

if (!Regex.IsMatch(newPassword, @"^(?=.\*[a-zA-Z])(?=.\*[0-9])(?=.\*[!@#$%^&\*]).{12,}$"))

error += "Новий пароль має бути мінімум 12 символів і включати літери, цифри та спеціальні символи.\n";

if (newPassword != confirmPassword)

error += "Нові паролі не співпадають.\n";

return error;

}

private bool UpdatePasswordInDatabase(string username, string currentPassword, string newPassword)

{

string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\nemof\OneDrive\Изображения\Lab\_\_12\Lab\_\_12\Lab\_\_12\SocialNetwork.mdf;Integrated Security=True";

string query = "UPDATE Users SET Password = @NewPassword WHERE Username = @Username AND Password = @CurrentPassword";

using (SqlConnection conn = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@NewPassword", newPassword);

cmd.Parameters.AddWithValue("@Username", username);

cmd.Parameters.AddWithValue("@CurrentPassword", currentPassword);

conn.Open();

int rowsAffected = cmd.ExecuteNonQuery();

return rowsAffected > 0;

}

}

}

}

***EditNameForm.cs:***

using System;

using System.Data.SqlClient;

using System.Text.RegularExpressions;

using System.Windows.Forms;

namespace Lab\_\_12

{

public partial class EditNameForm : Form

{

private string username;

public string NewUsername { get; private set; }

public EditNameForm(string username)

{

InitializeComponent();

this.username = username;

nameTextBox.Text = username;

}

private void SaveButton\_Click(object sender, EventArgs e)

{

string newUsername = nameTextBox.Text.Trim();

if (!Regex.IsMatch(newUsername, @"^[a-zA-Z0-9]{1,15}$"))

{

MessageBox.Show("Невірне ім'я. Використовуйте тільки літери та цифри (макс. 15 символів).", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (username == newUsername)

{

MessageBox.Show("Нове ім'я співпадає з поточним.", "Інформація", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.DialogResult = DialogResult.Cancel;

this.Close();

return;

}

if (IsUsernameTaken(newUsername))

{

MessageBox.Show("Ім'я вже зайняте. Оберіть інше.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

UpdateUsernameInDatabase(username, newUsername);

}

private void UpdateUsernameInDatabase(string oldUsername, string newUsername)

{

string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\nemof\OneDrive\Изображения\Lab\_\_12\Lab\_\_12\Lab\_\_12\SocialNetwork.mdf;Integrated Security=True";

string query = "UPDATE Users SET Username = @NewUsername WHERE Username = @OldUsername";

using (SqlConnection conn = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@NewUsername", newUsername);

cmd.Parameters.AddWithValue("@OldUsername", oldUsername);

conn.Open();

int rowsAffected = cmd.ExecuteNonQuery();

if (rowsAffected > 0)

{

MessageBox.Show("Ім'я успішно оновлено!", "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.NewUsername = newUsername;

this.DialogResult = DialogResult.OK;

this.Close();

}

else

{

MessageBox.Show("Не вдалося оновити ім'я.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

private bool IsUsernameTaken(string newUsername)

{

string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\nemof\OneDrive\Изображения\Lab\_\_12\Lab\_\_12\Lab\_\_12\SocialNetwork.mdf;Integrated Security=True";

string query = "SELECT COUNT(\*) FROM Users WHERE Username = @NewUsername";

using (SqlConnection conn = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@NewUsername", newUsername);

conn.Open();

int count = (int)cmd.ExecuteScalar();

return count > 0;

}

}

}

}

***EditEmailForm.cs:***

using System;

using System.Data.SqlClient;

using System.Text.RegularExpressions;

using System.Windows.Forms;

namespace Lab\_\_12

{

public partial class EditEmailForm : Form

{

private string username;

public EditEmailForm(string username)

{

InitializeComponent();

this.username = username;

LoadCurrentEmail();

}

private void LoadCurrentEmail()

{

string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\nemof\OneDrive\Изображения\Lab\_\_12\Lab\_\_12\Lab\_\_12\SocialNetwork.mdf;Integrated Security=True";

string query = "SELECT Email FROM Users WHERE Username = @Username";

using (SqlConnection conn = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@Username", username);

conn.Open();

emailTextBox.Text = cmd.ExecuteScalar()?.ToString();

}

}

private bool IsEmailTaken(string email)

{

string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\nemof\OneDrive\Изображения\Lab\_\_12\Lab\_\_12\Lab\_\_12\SocialNetwork.mdf;Integrated Security=True";

string query = "SELECT COUNT(\*) FROM Users WHERE Email = @Email AND Username != @Username";

using (SqlConnection conn = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@Email", email);

cmd.Parameters.AddWithValue("@Username", username);

conn.Open();

int count = (int)cmd.ExecuteScalar();

return count > 0;

}

}

private void SaveButton\_Click(object sender, EventArgs e)

{

string newEmail = emailTextBox.Text.Trim();

if (!Regex.IsMatch(newEmail, @"^[^@\s]+@[^@\s]+**\.**[^@\s]+$"))

{

MessageBox.Show("Невірний формат email.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (IsEmailTaken(newEmail))

{

MessageBox.Show("Цей email вже використовується іншим користувачем.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\nemof\OneDrive\Изображения\Lab\_\_12\Lab\_\_12\Lab\_\_12\SocialNetwork.mdf;Integrated Security=True";

string query = "UPDATE Users SET Email = @Email WHERE Username = @Username";

using (SqlConnection conn = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@Email", newEmail);

cmd.Parameters.AddWithValue("@Username", username);

conn.Open();

cmd.ExecuteNonQuery();

MessageBox.Show("Email успішно оновлено!", "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.Close();

}

}

}

}

***MessageSendForm.cs:***

using System;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace Lab\_\_12

{

public partial class MessageSendForm : Form

{

private string senderUsername;

public MessageSendForm(string senderUsername)

{

InitializeComponent();

this.senderUsername = senderUsername;

}

private void SendButton\_Click(object sender, EventArgs e)

{

string recipientUsername = toTextBox.Text.Trim();

string theme = themeTextBox.Text.Trim();

string text = messageTextBox.Text.Trim();

if (string.IsNullOrEmpty(recipientUsername) || string.IsNullOrEmpty(theme) || string.IsNullOrEmpty(text))

{

MessageBox.Show("Заповніть всі поля.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (!UserExists(recipientUsername))

{

MessageBox.Show("Отримувач не знайдений.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

int senderID = GetUserID(senderUsername);

int recipientID = GetUserID(recipientUsername);

if (senderID == -1 || recipientID == -1)

{

MessageBox.Show("Помилка при отриманні даних користувача.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\nemof\OneDrive\Изображения\Lab\_\_12\Lab\_\_12\Lab\_\_12\SocialNetwork.mdf;Integrated Security=True";

string query = "INSERT INTO Messages (SenderUserID, RecipientUserID, Theme, Text, Date) VALUES (@SenderUserID, @RecipientUserID, @Theme, @Text, @Date)";

using (SqlConnection conn = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@SenderUserID", senderID);

cmd.Parameters.AddWithValue("@RecipientUserID", recipientID);

cmd.Parameters.AddWithValue("@Theme", theme);

cmd.Parameters.AddWithValue("@Text", text);

cmd.Parameters.AddWithValue("@Date", DateTime.Now);

conn.Open();

cmd.ExecuteNonQuery();

}

MessageBox.Show("Повідомлення відправлено!", "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.Close();

}

private bool UserExists(string username)

{

string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\nemof\OneDrive\Изображения\Lab\_\_12\Lab\_\_12\Lab\_\_12\SocialNetwork.mdf;Integrated Security=True";

string query = "SELECT COUNT(\*) FROM Users WHERE Username = @Username";

using (SqlConnection conn = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@Username", username);

conn.Open();

int count = (int)cmd.ExecuteScalar();

return count > 0;

}

}

private int GetUserID(string username)

{

string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\nemof\OneDrive\Изображения\Lab\_\_12\Lab\_\_12\Lab\_\_12\SocialNetwork.mdf;Integrated Security=True";

string query = "SELECT UserID FROM Users WHERE Username = @Username";

using (SqlConnection conn = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@Username", username);

conn.Open();

object result = cmd.ExecuteScalar();

if (result != null)

{

return (int)result;

}

else

{

return -1;

}

}

}

}

}

***MessageReceiveForm.cs:***

using System;

using System.Data;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace Lab\_\_12

{

public partial class MessageReceiveForm : Form

{

private string username;

public MessageReceiveForm(string username)

{

InitializeComponent();

this.username = username;

LoadMessages();

}

private void LoadMessages()

{

int userID = GetUserID(username);

if (userID == -1)

{

MessageBox.Show("Помилка при отриманні даних користувача.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\nemof\OneDrive\Изображения\Lab\_\_12\Lab\_\_12\Lab\_\_12\SocialNetwork.mdf;Integrated Security=True";

string query = "SELECT MessageID, Theme FROM Messages WHERE RecipientUserID = @RecipientUserID";

using (SqlConnection conn = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@RecipientUserID", userID);

conn.Open();

SqlDataReader reader = cmd.ExecuteReader();

DataTable messagesTable = new DataTable();

messagesTable.Load(reader);

if (messagesTable.Rows.Count > 0)

{

messagesListBox.DisplayMember = "Theme";

messagesListBox.ValueMember = "MessageID";

messagesListBox.DataSource = messagesTable;

}

else

{

messagesListBox.Items.Add("No messages.");

}

}

}

private void MessagesListBox\_DoubleClick(object sender, EventArgs e)

{

if (messagesListBox.SelectedItem != null && messagesListBox.SelectedItem.ToString() != "No messages.")

{

int messageID = Convert.ToInt32(messagesListBox.SelectedValue);

ShowMessageDetails(messageID);

}

}

private void ShowMessageDetails(int messageID)

{

string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\nemof\OneDrive\Изображения\Lab\_\_12\Lab\_\_12\Lab\_\_12\SocialNetwork.mdf;Integrated Security=True";

string query = @"SELECT m.Theme, m.Text, m.Date, u.Username AS SenderUsername

FROM Messages m

JOIN Users u ON m.SenderUserID = u.UserID

WHERE m.MessageID = @MessageID";

using (SqlConnection conn = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@MessageID", messageID);

conn.Open();

SqlDataReader reader = cmd.ExecuteReader();

if (reader.Read())

{

string senderUsername = reader["SenderUsername"].ToString();

string theme = reader["Theme"].ToString();

string text = reader["Text"].ToString();

DateTime date = Convert.ToDateTime(reader["Date"]);

MessageDetailForm detailForm = new MessageDetailForm(senderUsername, theme, text, date);

detailForm.ShowDialog();

}

}

}

private int GetUserID(string username)

{

string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\nemof\OneDrive\Изображения\Lab\_\_12\Lab\_\_12\Lab\_\_12\SocialNetwork.mdf;Integrated Security=True";

string query = "SELECT UserID FROM Users WHERE Username = @Username";

using (SqlConnection conn = new SqlConnection(connectionString))

{

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@Username", username);

conn.Open();

object result = cmd.ExecuteScalar();

if (result != null)

{

return (int)result;

}

else

{

return -1;

}

}

}

}

}

***MessageDetailForm.cs:***

using System;

using System.Windows.Forms;

namespace Lab\_\_12

{

public partial class MessageDetailForm : Form

{

public MessageDetailForm(string from, string theme, string text, DateTime date)

{

InitializeComponent();

fromLabel.Text = $"From: {from}";

themeLabel.Text = $"Theme: {theme}";

textTextBox.Text = text;

dateLabel.Text = $"Date: {date.ToString("dd.MM.yyyy HH:mm")}";

}

}

}

***SplashForm.cs:***

using System;

using System.Windows.Forms;

namespace Lab\_\_12

{

public partial class SplashForm : Form

{

public SplashForm()

{

InitializeComponent();

// Таймер на 3 секунди

System.Windows.Forms.Timer timer = new System.Windows.Forms.Timer();

timer.Interval = 3000;

timer.Tick += Timer\_Tick;

timer.Start();

}

private void Timer\_Tick(object sender, EventArgs e)

{

// Зупинка таймера

(sender as System.Windows.Forms.Timer).Stop();

// Відкриття LoginForm

LoginForm loginForm = new LoginForm();

loginForm.Show();

// Приховуємо цю форму

this.Hide();

}

}

}

**Результат роботи програми:**

Изображение выглядит как Мультимедийное программное обеспечение, программное обеспечение, Графическое программное обеспечение, Редактирование

Автоматически созданное описание

Изображение выглядит как снимок экрана, программное обеспечение, Мультимедийное программное обеспечение, Графическое программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, Графическое программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

**Алгоритм побудови коду**

1. **Створення бази даних**:
   * Дві таблиці Users та Messages створено через SQL-запити.
   * Таблиця Users містить дані про користувачів, а Messages – про повідомлення між ними.
2. **Форма реєстрації (RegisterForm)**:
   * Перевірка валідності даних (регулярні вирази).
   * Унікальність даних (користувачів, email, номерів телефонів).
   * Додавання нового користувача до бази через SQL-запит.
3. **Форма входу (LoginForm)**:
   * Перевірка введених даних (користувач/пароль).
   * Вхід до профілю при успішній автентифікації.
4. **Форма профілю (ProfileForm)**:
   * Відображення основної інформації користувача.
   * Можливість редагування даних (ім’я, email, телефон, пароль).
5. **Функціонал повідомлень**:
   * Відправка повідомлень через форму MessageSendForm.
   * Отримання та перегляд повідомлень через MessageReceiveForm.
6. **Додаткові форми**:
   * Редагування окремих даних користувача (EditNameForm, EditEmailForm, EditPhoneForm, EditPasswordForm).
   * Деталізований перегляд повідомлень (MessageDetailForm).
7. **Splash-екран (SplashForm)**:
   * Відображення стартового екрану із затримкою перед відкриттям форми входу.
8. **Підключення до бази**:
   * Всі операції виконуються через підключення SqlConnection.
   * З’єднання з MDF-файлом реалізовано через шлях до файлу у проекті Visual Studio.

**Функціональність**

1. **Реєстрація користувачів**:
   * Перевірка унікальності логіну, email, телефону.
   * Зберігання зашифрованого пароля.
2. **Вхід до системи**:
   * Перевірка введених даних.
   * Перехід до профілю після успішного входу.
3. **Профіль користувача**:
   * Відображення інформації: логін, email, телефон.
   * Редагування даних: логіну, email, телефону, пароля.
4. **Повідомлення**:
   * Відправлення повідомлень іншому користувачеві.
   * Перегляд отриманих повідомлень із деталями (тема, текст, дата).
5. **Інтерфейс**:
   * Інтуїтивно зрозумілий дизайн.
   * Використання стилізації елементів (згладжені кнопки, підкреслення).
6. **База даних**:
   * Таблиця Users для зберігання даних користувачів.
   * Таблиця Messages для зберігання повідомлень із посиланнями на відправника/отримувача.

**Висновок**

1. **Результат**:
   * Створено повноцінний застосунок соціальної мережі з базовим функціоналом.
2. **Переваги**:
   * Використання MDF для інтегрованого з Visual Studio зберігання.
   * Гнучкість у роботі з повідомленнями та акаунтами.
   * Розширений інтерфейс для редагування даних.
3. **Можливі покращення**:
   * Додавання функцій для групових повідомлень.
   * Впровадження шифрування паролів та повідомлень.
   * Оптимізація запитів для великих обсягів даних.

Застосунок відповідає вимогам і забезпечує зручний функціонал для роботи з базою даних.

**Загальний Висновок**

Протягом лабораторної роботи було розроблено серію консольних та Windows Forms застосунків для взаємодії з базою даних MySQL.

Основні функції, реалізовані в застосунках:

* Створення бази даних та таблиць згідно з заданою предметною областю (транспорт, маршрути).
* Наповнення таблиць реалістичними даними.
* Виведення даних у консоль та форми в різних режимах (формуляр, таблиця).
* Організація зв'язків між таблицями.
* Фільтрація та сортування даних за різними критеріями.
* Додавання, видалення та редагування записів у базі даних.
* Розрахунок статистичних характеристик (середнє значення, максимум, мінімум).
* Використання бібліотеки MySql.Data.MySqlClient для роботи з MySQL.
* Застосування об'єктно-орієнтованого програмування (класи, об'єкти, методи).
* Робота з SQL-запитами (SELECT, INSERT, UPDATE, DELETE).
* Динамічне формування елементів інтерфейсу.
* Обробка подій (натискання кнопки, зміна вибору).
* Використання регулярних виразів для валідації даних.

В цілому, лабораторна робота дозволила закріпити знання та навички у розробці застосунків для роботи з базами даних, включаючи проектування структури бази даних, написання SQL-запитів, обробку даних та створення зручного інтерфейсу користувача.