**33 Технология LINQ to SQL**

Задание 1. Выполните операции просмотра, изменения, добавления и удаления данных с помощью технологии LinqToSQL. Принцип работы описан в документе «Пример LINQ to SQL.docx»

Листинг программы:

namespace Task1

{

class Program

{

static async Task Main(string[] args)

{

string dbPath = @"..\..\..\Database\DBTur\_firm.db";

TouristContext dbContext = new TouristContext(dbPath);

Tourist newTourist = CreateTourist();

int addResult = await dbContext.AddTouristAsync(newTourist);

if (addResult > 0)

{

Console.WriteLine($"Турист добавлен.");

}

else

{

Console.WriteLine($"Турист не добавлен.");

}

List<Tourist> tourists = await dbContext.GetTouristsAsync();

Console.WriteLine("Список туристов:");

PrintTourists(tourists);

if (tourists.Count > 0)

{

Console.WriteLine("Выберите туриста для удаления:");

int touristIdToDelete = int.Parse(Console.ReadLine());

int deleteResult = await dbContext.DeleteTouristAsync(touristIdToDelete);

if (deleteResult > 0)

{

Console.WriteLine($"Турист удален.");

}

else

{

Console.WriteLine($"Турист не найден.");

}

}

tourists = await dbContext.GetTouristsAsync();

Console.WriteLine("Список туристов:");

PrintTourists(tourists);

}

static Tourist CreateTourist()

{

Console.WriteLine("Введите фамилию туриста:");

string lastName = Console.ReadLine();

Console.WriteLine("Введите имя туриста:");

string firstName = Console.ReadLine();

Console.WriteLine("Введите отчество туриста:");

string patronymic = Console.ReadLine();

Tourist newTourist = new Tourist

{

LastName = lastName,

FirstName = firstName,

Patronymic = patronymic

};

return newTourist;

}

static void PrintTourists(List<Tourist> tourists)

{

foreach (var tourist in tourists)

{

Console.WriteLine($"ID: {tourist.TouristId}, ФИО: {tourist.LastName} {tourist.FirstName} {tourist.Patronymic}");

}

}

}

}

namespace Task1.Models

{

[Table("Туристы")]

public class Tourist

{

[PrimaryKey, AutoIncrement, Column("Код\_туриста")]

public int TouristId { get; set; }

[Column("Фамилия")]

public string LastName { get; set; }

[Column("Имя")]

public string FirstName { get; set; }

[Column("Отчество")]

public string Patronymic { get; set; }

}

}

namespace Task1.Contexts

{

public class TouristContext

{

private readonly SQLiteAsyncConnection \_database;

public TouristContext(string dbPath)

{

\_database = new SQLiteAsyncConnection(dbPath);

\_database.ExecuteAsync("PRAGMA foreign\_keys = ON;").Wait();

}

public async Task<List<Tourist>> GetTouristsAsync()

{

return await \_database.Table<Tourist>().ToListAsync();

}

public async Task<Tourist> GetTouristAsync(int id)

{

return await \_database.Table<Tourist>().Where(t => t.TouristId == id).FirstOrDefaultAsync();

}

public async Task<int> AddTouristAsync(Tourist tourist)

{

return await \_database.InsertAsync(tourist);

}

public async Task<int> DeleteTouristAsync(int touristId)

{

await \_database.Table<TouristInfo>().DeleteAsync(info => info.TouristId == touristId);

var relatedPackages = await \_database.Table<Package>().Where(p => p.TouristId == touristId).ToListAsync();

foreach (var package in relatedPackages)

{

await \_database.Table<Payment>().DeleteAsync(p => p.PackageId == package.PackageId);

await \_database.DeleteAsync(package);

}

var tourist = await GetTouristAsync(touristId);

if (tourist != null)

{

return await \_database.DeleteAsync(tourist);

}

return 0;

}

}

}

Таблица 33.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Ерошкевич, Кирилл, Андреевич, 3 | Турист добавлен, турист удален |

Анализ результатов:

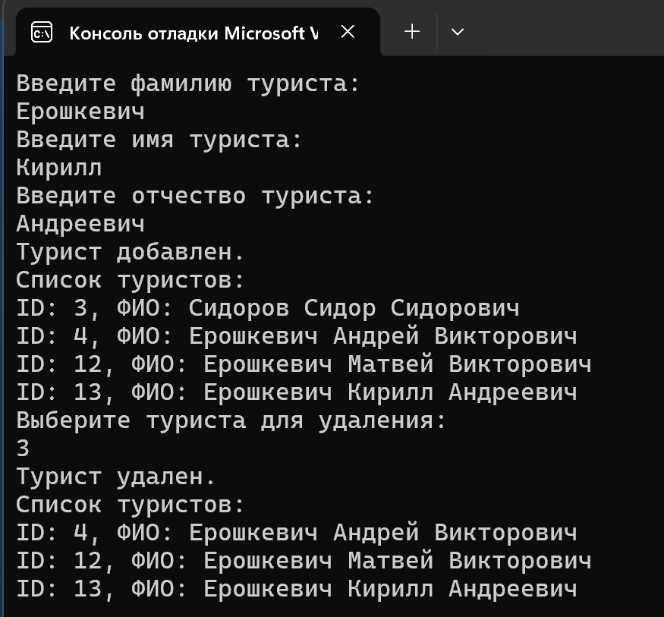


Рисунок 33.1 – Результат работы программы