

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

# **Кафедра системного програмування та спеціалізованих комп’ютерних систем**

**Лабораторна робота №1**

з дисципліни

**«Бази даних і засоби управління»**

Виконав: студент ІII курсу

ФПМ групи КВ-13

Ольховський Максим Олександрович

Київ – 2023

**Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL**

*Метою роботи* є здобуття вмінь програмування прикладних додатків баз даних PostgreSQL.

*Загальне* *завдання* роботи полягає у наступному:

1. Реалізувати функції перегляду, внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі №1, засобами консольного інтерфейсу.
2. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.
3. Забезпечити реалізацію пошуку за декількома атрибутами з двох та більше сутностей одночасно: для числових атрибутів – у рамках діапазону, для рядкових – як шаблон функції LIKE оператора SELECT SQL, для логічного типу – значення True/False, для дат – у рамках діапазону дат.
4. Програмний код виконати згідно шаблону MVC (модель-подання-контролер).

**Концептуальна модель предметної галузі “Система обліку наявності медичних препаратів у аптеках”**

У даній моделі предметної галузі “Система обліку наявності медичних препаратів у аптеках” (Малюнок 1) я виділив наступні сутності та зв’язки між ними:

1. Сутність “Компанії” з атрибутами: ID, Назва, Країна, Власник;

2. Сутність “Ліки” з атрибутами: ID, Назва, Виробник;

3. Сутність “Аптеки” з атрибутами: ID, Місто, Назва;

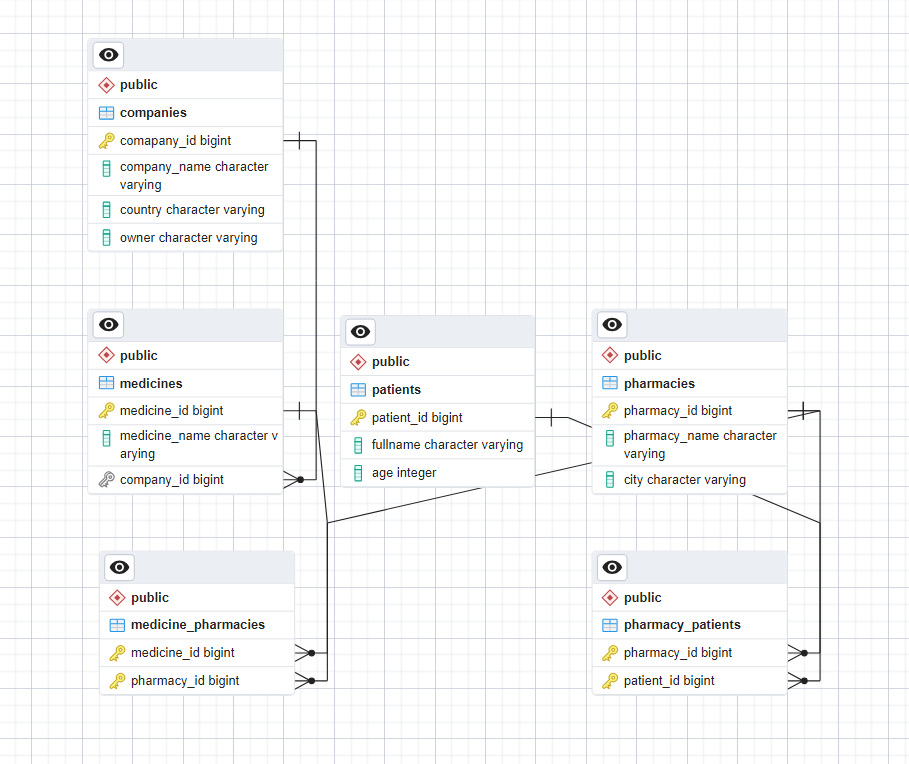
4. Сутність “Пацієнти” з атрибутами: ID, ПІБ, Вік.

Одна компанія може створити декілька варіантів ліків, тому між сутностями “Компанії” та “Ліки” зв’язок R(1:N).

Один вид ліків може направитися в багато аптек і багато видів ліків можуть направитися в одну аптеку, тому зв'язок між сутностями "Ліки" та "Аптеки" R(N:M).

Один пацієнт може відвідати багато аптек і одну аптеку можуть відвідати багато пацієнтів, тому зв'язок між сутностями "Аптеки" та "Пацієнти" R(N:M).

**Діаграма сутність-зв’язок та структура бази даних**

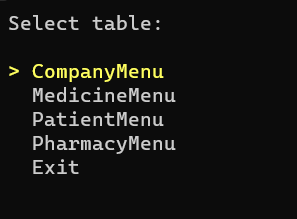


**Використані технології**

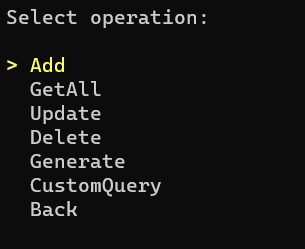
Біліотека Dapper для доступу до бази даних. Для створення інтерфейсу в консолі була використана бібліотека Spectre.Console

**Схема меню користувача**

Головне меню:

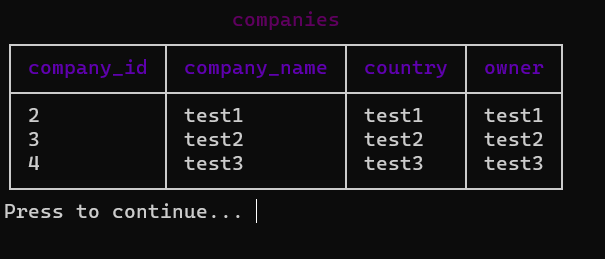


Меню таблиці

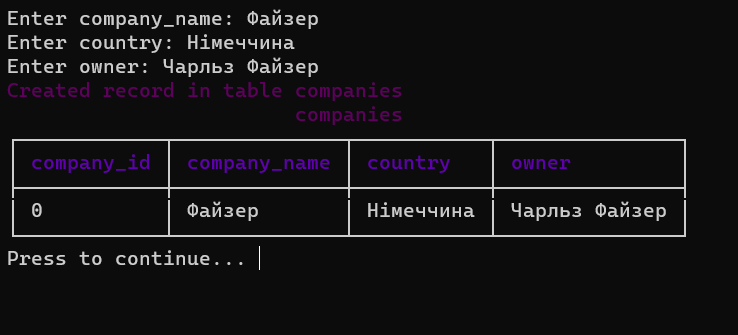


**Тестування операцій**

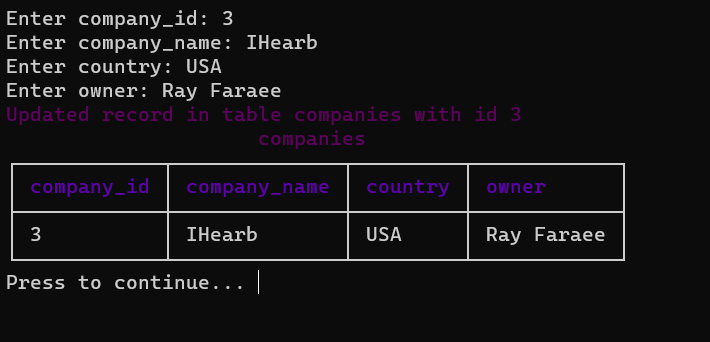
Перевіримо операції на прикладі таблиці CompanyName. Операція отримання всіх значень таблиці:



Створення запису



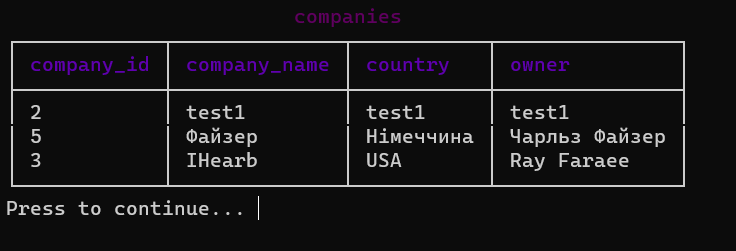
Оновлення запису





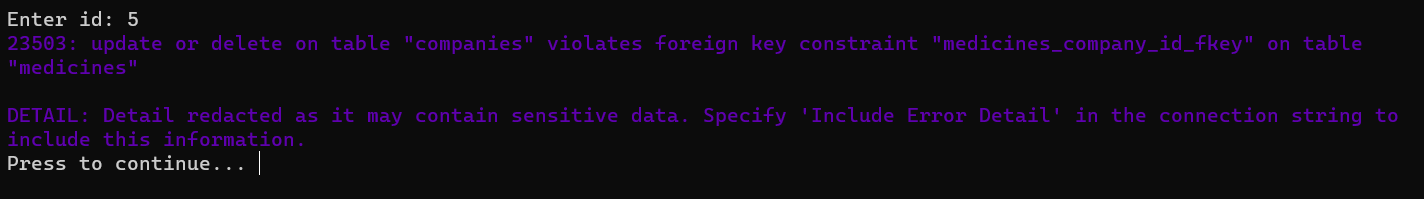
Delete:





**Видалення батьківських таблиць**

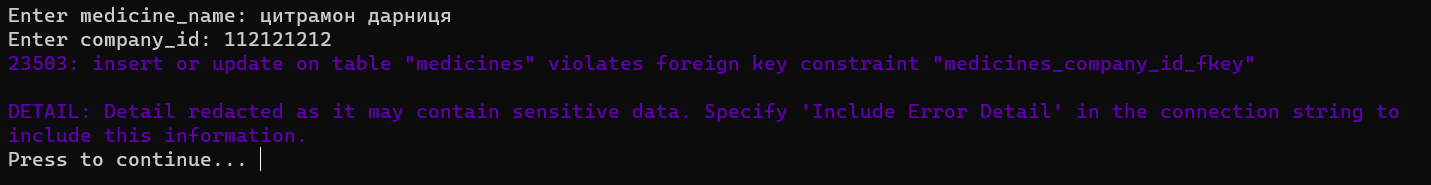
Спробуємо видалити запис із таблиці companies

****

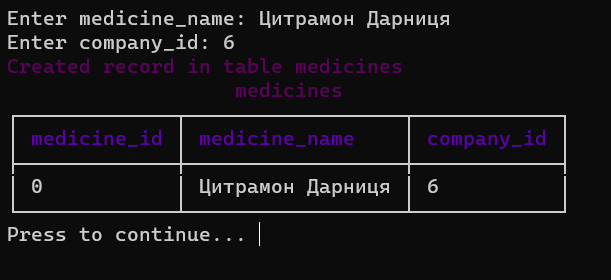
В нас існують ліки, які мають виробника під п'ятим айді. Програма працює справно, видалення заборонене. Часто бувають ситуації, що нам потрібно зберігати ліки, не дивлячись на те, що виробник з тих чи інших причин має бути видалений з бази даних. Якщо все таки ми хочемо видалити виробника, то тоді потрібно наявно видалити всі ліки.

**Вставка дочірніх таблиць**

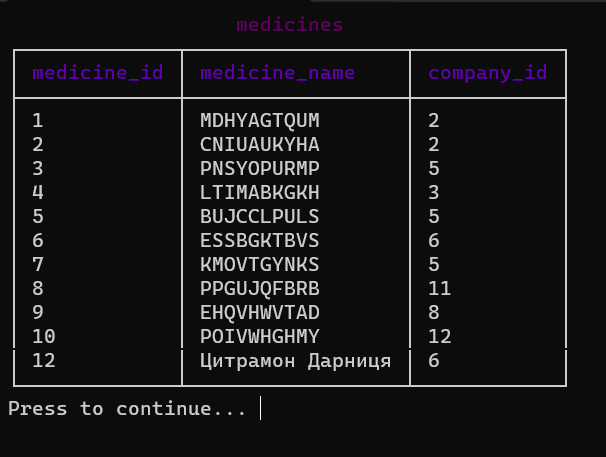
Спробуємо створити запис з невірним ключем



Очевидно, що програма функціонує належним чином. Давайте спробуємо вставити значення з використанням вже існуючого ключа.

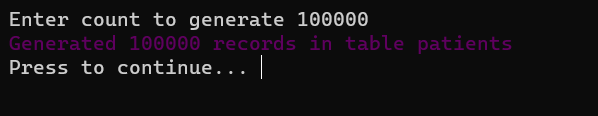


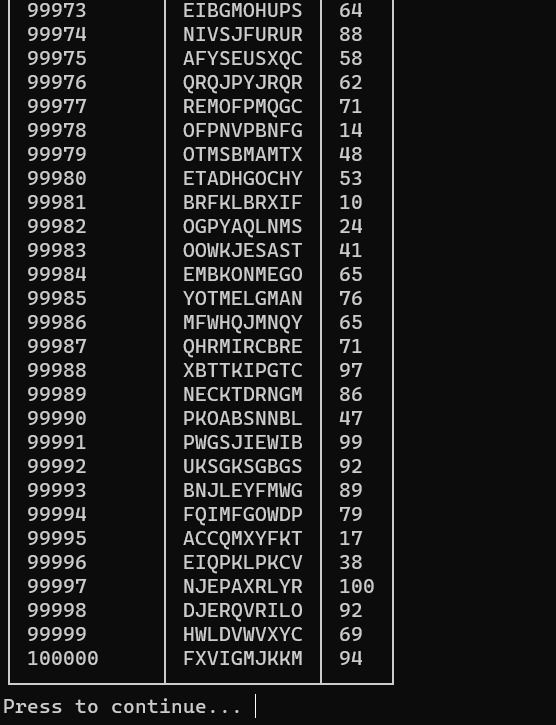
Все пройшло успішно



**Генерація рандомізованих даних**

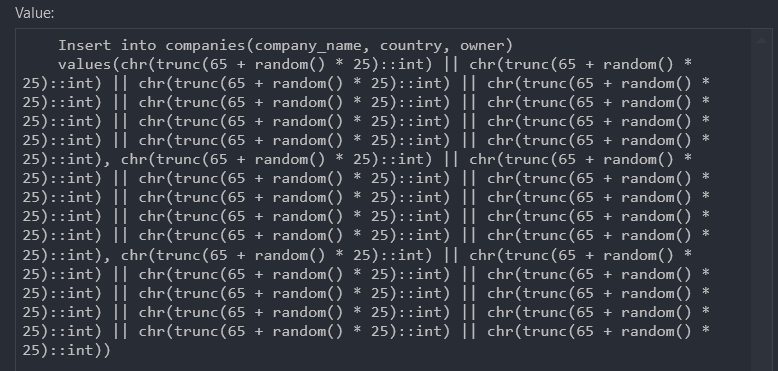
Для перевірки правильності складних запитів створимо 1000 записів для всіх таблиць. Додатково, для тестування генерації великої кількості даних, ми створимо 100000 записів для таблиці "patients" за допомогою опції "generate".



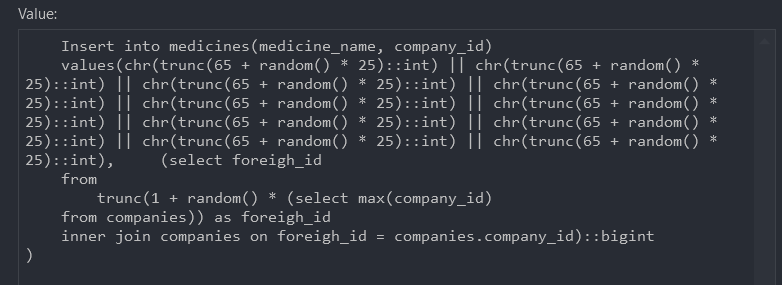


**Запити для генерації даних:**

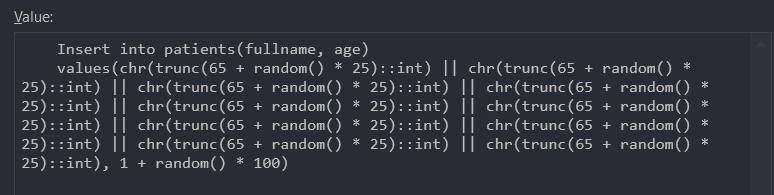
**companies:**



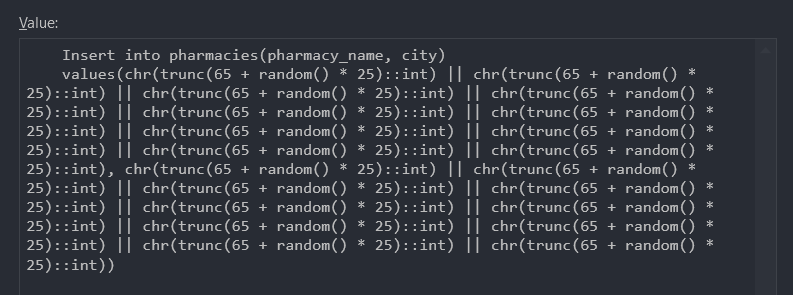
**medicines:**



**patients:**

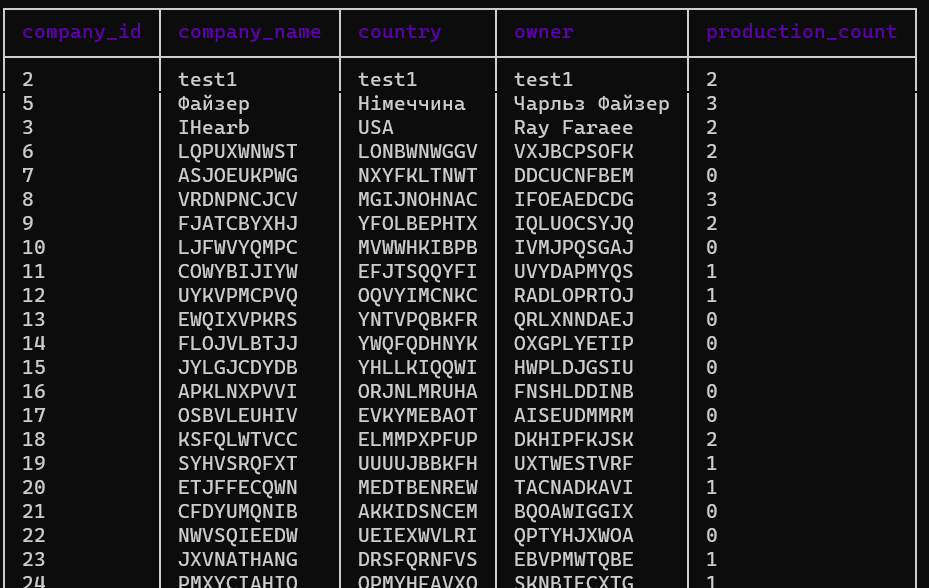


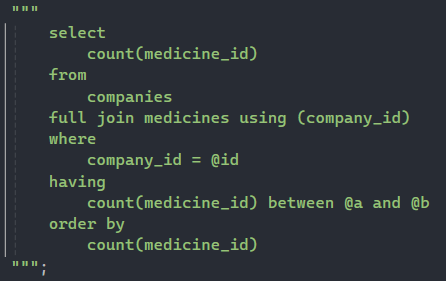
**pharmacies:**



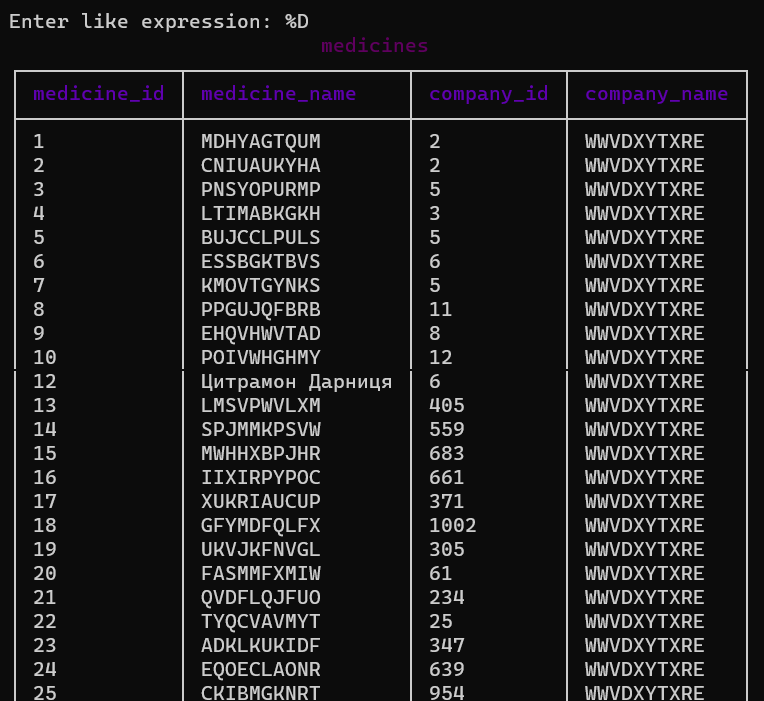
**Ілюстрації уведення пошукового запиту та результатів виконання запитів**

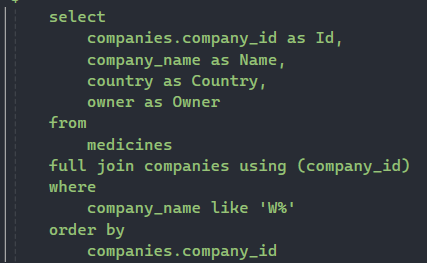
**companies:**



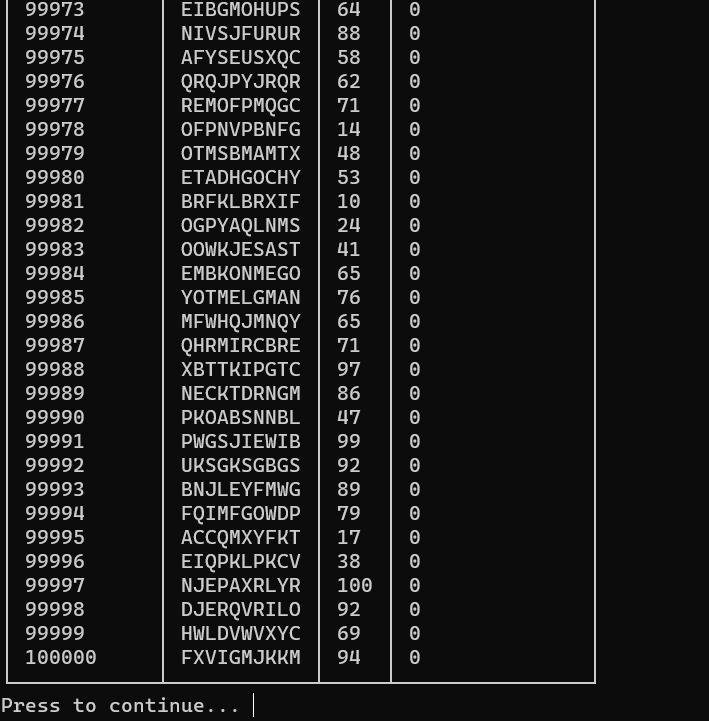


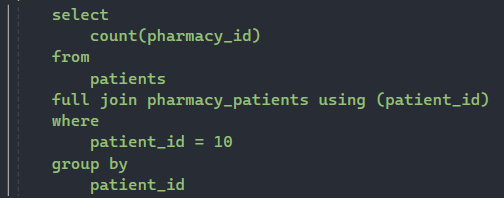
**medicines:**

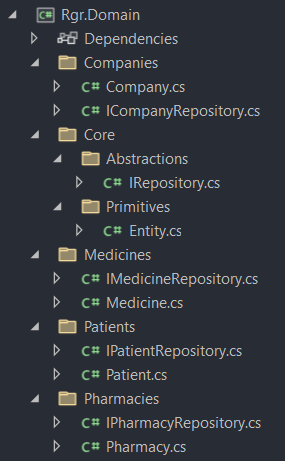
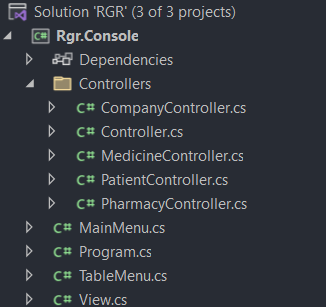
****

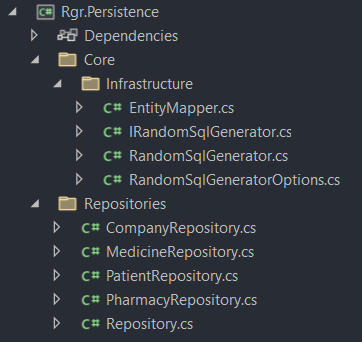
****

**patients:**

****



**Ілюстрації програмного коду**



У дизайні MVC моделлю виступає Repository pattern за допомогою якого ми і звертаємось до баз даних. “V” у нашому проекті це статичний клас View<T> і за контролери відповідає однойменний клас Controllers

**Model(Repository)**

**using Dapper;**

**using Rgr.Domain.Core.Abstractions;**

**using Rgr.Domain.Core.Primitives;**

**using RGR.Persistence.Core.Infrastructure;**

**using System.Data;**

**using System.Linq;**

**using static Dapper.SqlMapper;**

**namespace RGR.Persistence.Repositories;**

**public abstract class Repository<TEntity> : IRepository<TEntity> where TEntity : IEntity, new()**

**{**

**protected readonly IDbConnection \_connection;**

**protected readonly IRandomSqlGenerator \_sqlGenerator;**

**public Repository(IDbConnection connection)**

**{**

**\_connection = connection;**

**\_sqlGenerator = new RandomSqlGenerator<TEntity>(10, 100);**

**}**

**protected string ColumnsString(bool hasPrimaryKeys = false) =>**

**string.Join(", ", EntityMapper<TEntity>.GetColumnNames(hasPrimaryKeys));**

**protected string ParametersString(bool hasPrimaryKeys = false) =>**

**string.Join(", ", EntityMapper<TEntity>.GetParameterNames(hasPrimaryKeys));**

**protected string ColumnAsString(bool hasPrimaryKeys = false) =>**

**string.Join(", ", EntityMapper<TEntity>.GetColumnNames(hasPrimaryKeys)**

**.Zip(EntityMapper<TEntity>.GetPropertyNames(hasPrimaryKeys), (column, prop) => new { column, prop })**

**.Select(e => e.column + " as " + e.prop));**

**protected string GetParametrColumnString(bool hasPrimaryKeys = false) =>**

**string.Join(", ", EntityMapper<TEntity>.GetParameterNames(hasPrimaryKeys)**

**.Zip(EntityMapper<TEntity>.GetColumnNames(hasPrimaryKeys), (param, column) => new { param, column })**

**.Select(e => e.column + " = " + e.param));**

**public void Add(TEntity entity)**

**{**

**string sql =**

**$"""**

**Insert into {EntityMapper<TEntity>.GetTableName()}({ColumnsString()})**

**values({ParametersString()})**

**""";**

**\_connection.Open();**

**try**

**{**

**\_connection.Execute(sql, entity);**

**}**

**finally**

**{**

**\_connection.Close();**

**}**

**}**

**public void Delete(TEntity entity)**

**{**

**string sql =**

**$"""**

**Delete from {EntityMapper<TEntity>.GetTableName()}**

**where {EntityMapper<TEntity>.GetPrimaryKeyNames().FirstOrDefault()} = @Id**

**""";**

**\_connection.Open();**

**try**

**{**

**\_connection.Execute(sql, new { entity.Id });**

**}**

**finally**

**{**

**\_connection.Close();**

**}**

**}**

**public void Update(TEntity entity)**

**{**

**string sql =**

**$"""**

**update {EntityMapper<TEntity>.GetTableName()}**

**set {GetParametrColumnString()}**

**where {EntityMapper<TEntity>.GetPrimaryKeyNames().FirstOrDefault()} = @Id**

**""";**

**\_connection.Open();**

**try**

**{**

**\_connection.Execute(sql, entity);**

**}**

**finally**

**{**

**\_connection.Close();**

**}**

**}**

**public IEnumerable<TEntity> GetAll()**

**{**

**string sql =**

**$"""**

**select {ColumnAsString(true)} from {EntityMapper<TEntity>.GetTableName()}**

**""";**

**\_connection.Open();**

**IEnumerable<TEntity> result;**

**try**

**{**

**result = \_connection.Query<TEntity>(sql);**

**}**

**finally**

**{**

**\_connection.Close();**

**}**

**return result;**

**}**

**public void Generate(long count)**

**{**

**string sql =**

**$"""**

**Insert into {EntityMapper<TEntity>.GetTableName()}({ColumnsString()})**

**values({\_sqlGenerator.GenerateSql()})**

**""";**

**\_connection.Open();**

**try**

**{**

**for (int i = 0; i < count; i++)**

**\_connection.Execute(sql);**

**}**

**finally**

**{**

**\_connection.Close();**

**}**

**}**

**}**

**Model**

**using Npgsql;**

**using Rgr.Domain.Core.Primitives;**

**using RGR.Persistence.Core.Infrastructure;**

**using Spectre.Console;**

**namespace Rgr.Console;**

**public static class View<TEntity> where TEntity : IEntity, new()**

**{**

**public static void PrintTable(IEnumerable<TEntity> entities)**

**{**

**Table table = new Table();**

**table.Title = new TableTitle(EntityMapper<TEntity>.GetTableName(), new(Color.DeepPink4));**

**table.AddColumns(EntityMapper<TEntity>.GetColumnNames(true)**

**.Select(c => new TableColumn(new Markup(c, Color.Purple4\_1))).ToArray());**

**foreach (var entity in entities)**

**{**

**var values = typeof(TEntity).GetProperties()**

**.Select(p => p.GetValue(entity)?.ToString() ?? "NULL");**

**table.AddRow(values.ToArray());**

**}**

**AnsiConsole.Write(table);**

**AnsiConsole.Prompt(**

**new TextPrompt<string>("Press to continue...")**

**.AllowEmpty());**

**AnsiConsole.Clear();**

**}**

**public static void PrintCreate(TEntity entity)**

**{**

**AnsiConsole.Write(new Markup($"Created record in table {EntityMapper<TEntity>.GetTableName()}\n", new(Color.DeepPink4)));**

**PrintTable([entity]);**

**AnsiConsole.Prompt(**

**new TextPrompt<string>("Press to continue...")**

**.AllowEmpty());**

**AnsiConsole.Clear();**

**}**

**public static void PrintUpdate(TEntity entity)**

**{**

**AnsiConsole.Write(new Markup($"Updated record in table {EntityMapper<TEntity>.GetTableName()} with id {entity.Id}\n", new(Color.DeepPink4)));**

**PrintTable([entity]);**

**AnsiConsole.Prompt(**

**new TextPrompt<string>("Press to continue...")**

**.AllowEmpty());**

**AnsiConsole.Clear();**

**}**

**public static void PrintDelete(TEntity entity)**

**{**

**AnsiConsole.Write(new Markup($"Deleted record in table {EntityMapper<TEntity>.GetTableName()}\n", new(Color.DeepPink4)));**

**PrintTable([entity]);**

**AnsiConsole.Prompt(**

**new TextPrompt<string>("Press to continue...")**

**.AllowEmpty());**

**AnsiConsole.Clear();**

**}**

**public static void PrintError(Exception exception)**

**{**

**AnsiConsole.Write(new Markup($"{exception.Message}\n", new(Color.Purple4\_1)));**

**AnsiConsole.Prompt(**

**new TextPrompt<string>("Press to continue...")**

**.AllowEmpty());**

**AnsiConsole.Clear();**

**}**

**public static void PrintGenerate(int count)**

**{**

**AnsiConsole.Write(new Markup($"Generated {count} records in table {EntityMapper<TEntity>.GetTableName()}\n", new(Color.DeepPink4)));**

**AnsiConsole.Prompt(**

**new TextPrompt<string>("Press to continue...")**

**.AllowEmpty());**

**AnsiConsole.Clear();**

**}**

**}**

**Controller**

**using Rgr.Domain.Core.Abstractions;**

**using Rgr.Domain.Core.Primitives;**

**using Spectre.Console;**

**using System.ComponentModel;**

**using System.ComponentModel.DataAnnotations;**

**using System.ComponentModel.DataAnnotations.Schema;**

**using System.Reflection;**

**using ValidationResult = Spectre.Console.ValidationResult;**

**namespace Rgr.Console.Controllers;**

**public class Controller<TEntity> where TEntity : IEntity, new()**

**{**

**protected readonly IRepository<TEntity> \_repository;**

**public Controller(IRepository<TEntity> repository)**

**{**

**\_repository = repository;**

**}**

**public void GetAll()**

**{**

**try**

**{**

**View<TEntity>.PrintTable(\_repository.GetAll());**

**}**

**catch (Exception ex)**

**{**

**View<TEntity>.PrintError(ex);**

**}**

**}**

**public void Add()**

**{**

**TEntity entity = new TEntity();**

**foreach (var prop in typeof(TEntity).GetProperties())**

**{**

**if (prop.GetCustomAttribute<KeyAttribute>() != null)**

**continue;**

**var converter = TypeDescriptor.GetConverter(prop.PropertyType);**

**object? result;**

**do**

**{**

**var entered = AnsiConsole.Prompt(**

**new TextPrompt<string>($"Enter {prop.GetCustomAttribute<ColumnAttribute>()?.Name ?? prop.Name}:")**

**.ValidationErrorMessage("Not valid value"));**

**try**

**{**

**result = converter.ConvertFrom(entered);**

**break;**

**}**

**catch**

**{**

**AnsiConsole.WriteLine("Not valid value type");**

**continue;**

**}**

**}**

**while(true);**

**prop.SetValue(entity, result);**

**}**

**try**

**{**

**\_repository.Add(entity);**

**View<TEntity>.PrintCreate(entity);**

**}**

**catch (Exception ex)**

**{**

**View<TEntity>.PrintError(ex);**

**}**

**}**

**public void Update()**

**{**

**TEntity entity = new TEntity();**

**foreach (var prop in typeof(TEntity).GetProperties())**

**{**

**var converter = TypeDescriptor.GetConverter(prop.PropertyType);**

**object? result;**

**do**

**{**

**var entered = AnsiConsole.Prompt(**

**new TextPrompt<string>($"Enter {prop.GetCustomAttribute<ColumnAttribute>()?.Name ?? prop.Name}:")**

**.ValidationErrorMessage("Not valid value"));**

**try**

**{**

**result = converter.ConvertFrom(entered);**

**break;**

**}**

**catch**

**{**

**AnsiConsole.WriteLine("Not valid value type");**

**continue;**

**}**

**}**

**while (true);**

**prop.SetValue(entity, result);**

**}**

**try**

**{**

**\_repository.Update(entity);**

**View<TEntity>.PrintUpdate(entity);**

**}**

**catch (Exception ex)**

**{**

**View<TEntity>.PrintError(ex);**

**}**

**}**

**public void Delete()**

**{**

**TEntity entity = new TEntity() { Id = AnsiConsole.Prompt(**

**new TextPrompt<long>($"Enter id:")**

**.ValidationErrorMessage("Not valid value")**

**)**

**};**

**try**

**{**

**\_repository.Delete(entity);**

**View<TEntity>.PrintDelete(entity);**

**}**

**catch (Exception ex)**

**{**

**View<TEntity>.PrintError(ex);**

**}**

**}**

**public void Generate()**

**{**

**int count = AnsiConsole.Prompt(**

**new TextPrompt<int>("Enter count to generate")**

**.ValidationErrorMessage("Not valid value"));**

**try**

**{**

**\_repository.Generate(count);**

**View<TEntity>.PrintGenerate(count);**

**}**

**catch (Exception ex)**

**{**

**View<TEntity>.PrintError(ex);**

**}**

**}**

**}**

Мій телеграм: <https://t.me/Maks_01111>

Мій GitHub: https://github.com/fantomas00/RGR\_DB