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 III 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:

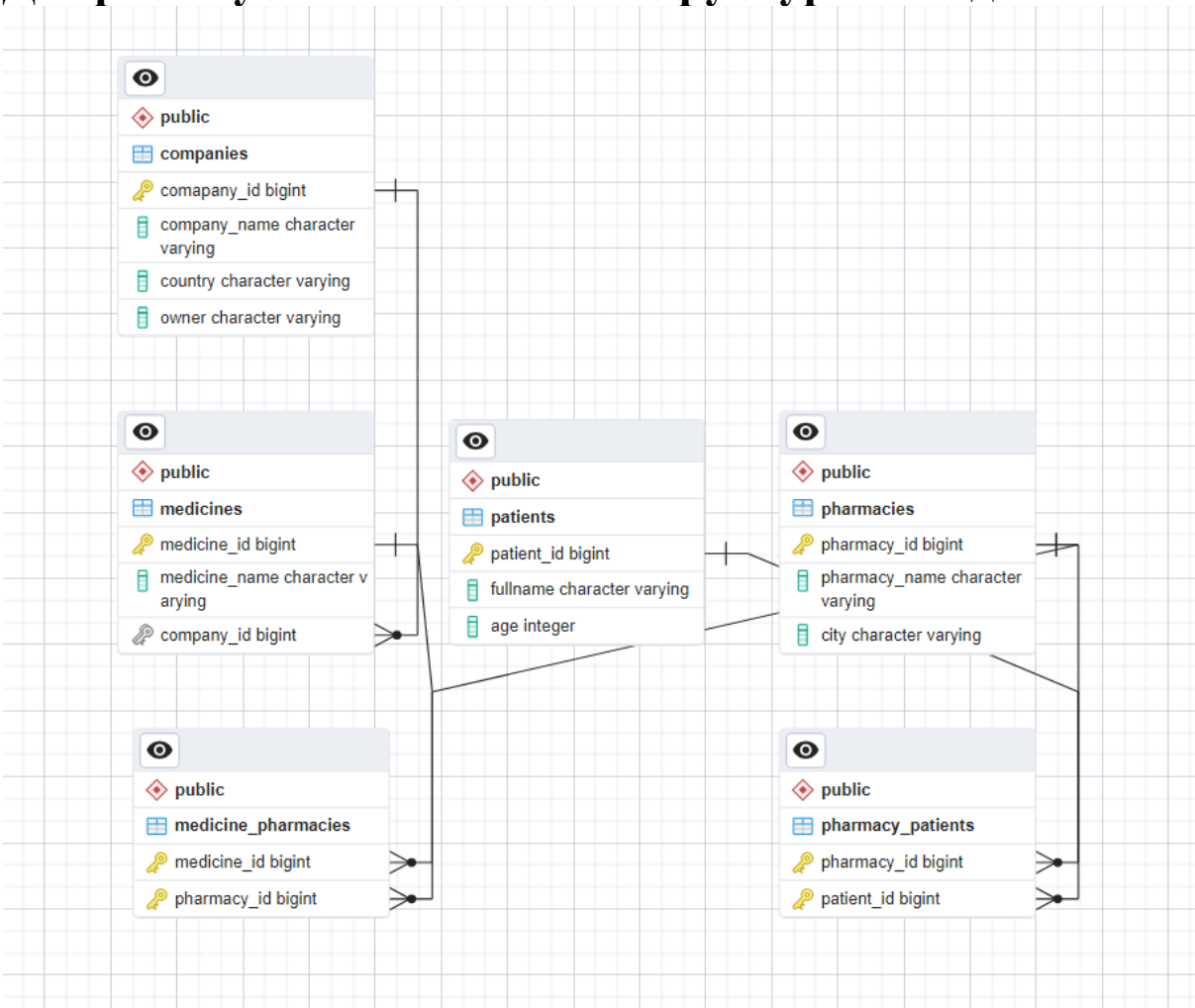
1. Сутність “Компанії” з атрибутами: ID, Назва, Країна, Власник;
2. Сутність “Ліки” з атрибутами: ID, Назва, Виробник;
3. Сутність “Аптеки” з атрибутами: ID, Місто, Назва;
4. Сутність “Пацієнти” з атрибутами: ID, ПІБ, Вік.

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

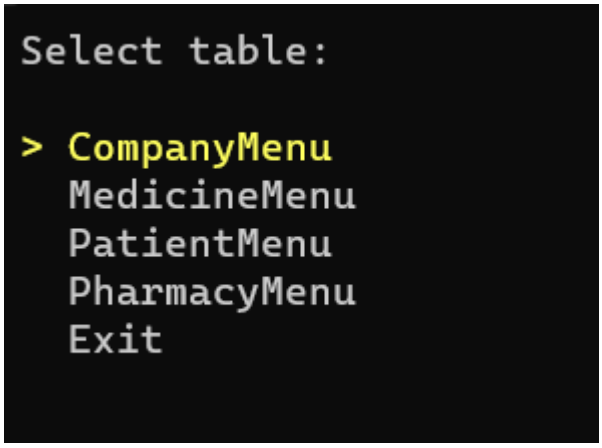


Використані технології

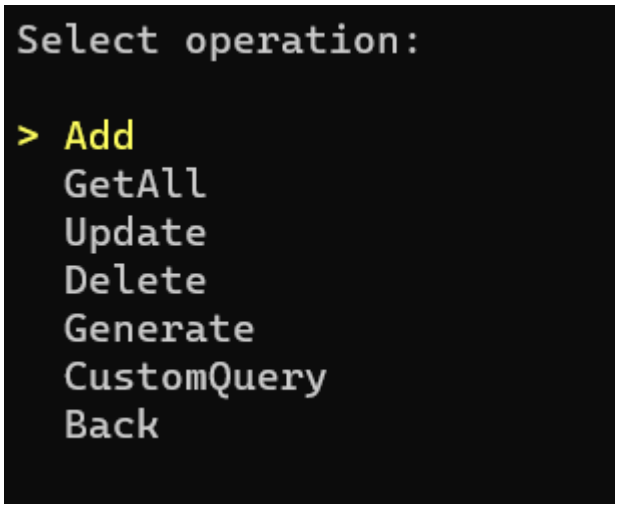
Біліотека Dapper для доступу до бази даних. Для створення інтерфейсу в консолі була використана бібліотека Spectre.Console

Схема меню користувача

Головне меню:

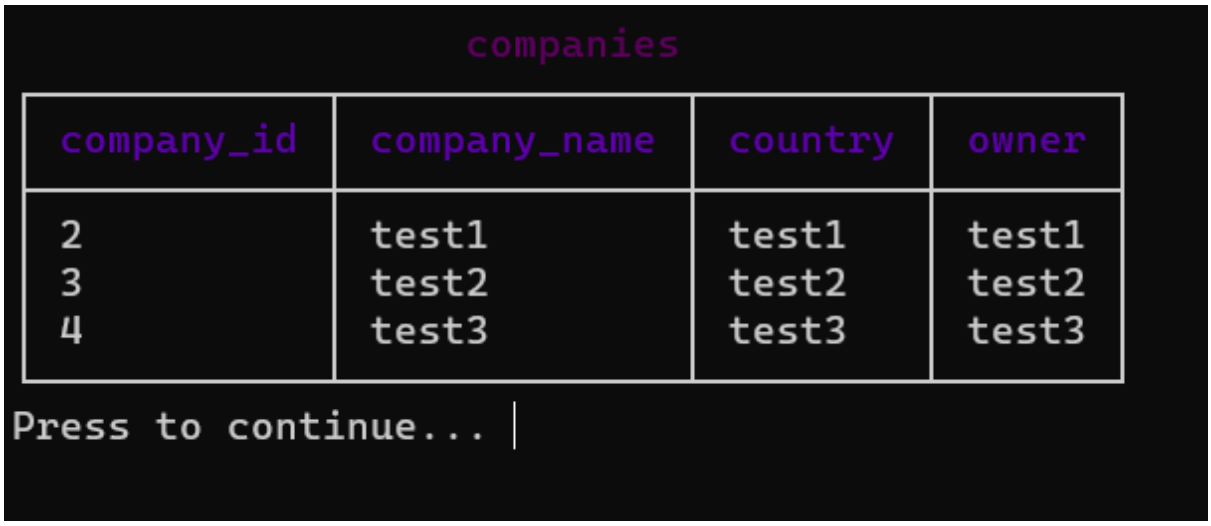


Меню таблиці



Тестування операцій

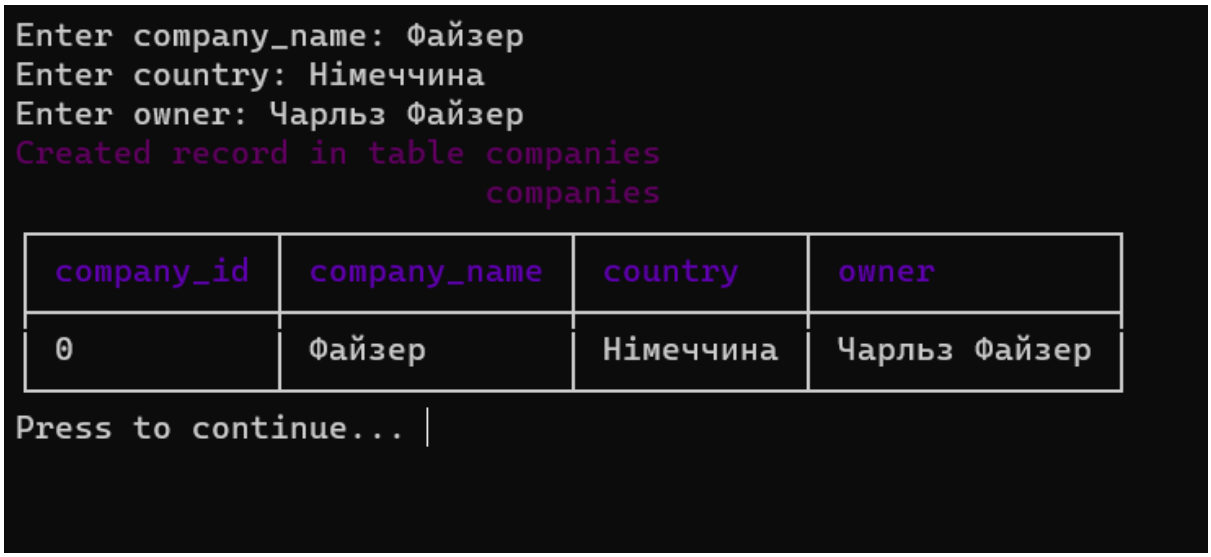
Перевіримо операції на прикладі таблиці CompanyName. Операція отримання всіх значень таблиці:


companies

company_id	company_name	country	owner
2	test1	test1	test1
3	test2	test2	test2
4	test3	test3	test3

Press to continue...

Створення запису


Enter company_name: Файзер
Enter country: Німеччина
Enter owner: Чарльз Файзер
Created record in table companies

companies

company_id	company_name	country	owner
0	Файзер	Німеччина	Чарльз Файзер

Press to continue...

Оновлення запису

```
Enter company_id: 3
Enter company_name: IHearb
Enter country: USA
Enter owner: Ray Faraee
Updated record in table companies with id 3
                companies
```

company_id	company_name	country	owner
3	IHearb	USA	Ray Faraee

```
Press to continue...
```

```
                companies
```

company_id	company_name	country	owner
2	test1	test1	test1
4	test3	test3	test3
5	Файзер	Німеччина	Чарльз Файзер
3	IHearb	USA	Ray Faraee

```
Press to continue...
```

Delete:

```
Enter id: 4
Deleted record in table companies
                companies
```

company_id	company_name	country	owner
4	NULL	NULL	NULL

```
Press to continue...
```

companies

company_id	company_name	country	owner
2	test1	test1	test1
5	Файзер	Німеччина	Чарльз Файзер
3	IHearb	USA	Ray Faraee

Press to continue...

Видалення батьківських таблиць

Спробуємо видалити запис із таблиці companies

```
Enter id: 5
23503: update or delete on table "companies" violates foreign key constraint "medicines_company_id_fkey" on table
"medicines"

DETAIL: Detail redacted as it may contain sensitive data. Specify 'Include Error Detail' in the connection string to
include this information.
Press to continue...
```

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

```
Enter medicine_name: цитрамон дарниця
Enter company_id: 112121212
23503: insert or update on table "medicines" violates foreign key constraint "medicines_company_id_fkey"

DETAIL: Detail redacted as it may contain sensitive data. Specify 'Include Error Detail' in the connection string to
include this information.
Press to continue...
```

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

```
Enter medicine_name: Цитрамон Дарниця
Enter company_id: 6
Created record in table medicines
                  medicines
```

medicine_id	medicine_name	company_id
0	Цитрамон Дарниця	6

```
Press to continue...
```

Все пройшло успішно

```
                  medicines
```

medicine_id	medicine_name	company_id
1	MDHYAGTQUM	2
2	CNIUAUKYHA	2
3	PNSYOPURMP	5
4	LTIMABKGKH	3
5	BUJCCLPULS	5
6	ESSBGKTBVS	6
7	KMOVTGYNKS	5
8	PPGUJQFBRB	11
9	EHQVHWVTAD	8
10	POIVWHGHMY	12
12	Цитрамон Дарниця	6

```
Press to continue...
```

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".

```
Enter count to generate 100000
Generated 100000 records in table patients
Press to continue...
```

99973	EIBGMOHUPS	64
99974	NIVSJFURUR	88
99975	AFYSEUSXQC	58
99976	QRQJPYJRQR	62
99977	REMOFPMQGC	71
99978	OFPNVPBNFG	14
99979	OTMSBMAMTX	48
99980	ETADHGOCHY	53
99981	BRFKLBRXIF	10
99982	OGPYAQLNMS	24
99983	OOWKJESAST	41
99984	EMBKONMEGO	65
99985	YOTMELGMAN	76
99986	MFWHQJMNQY	65
99987	QHRMIRCBRE	71
99988	XBTTKIPGTC	97
99989	NECKTDRNGM	86
99990	PKOABSNNBL	47
99991	PWGSJIEWIB	99
99992	UKSGKSGBGS	92
99993	BNJLEYFMWG	89
99994	FQIMFGOWDP	79
99995	ACCQMXYFKT	17
99996	EIQPKLPKCV	38
99997	NJEPAXRLYR	100
99998	DJERQVRILO	92
99999	HWLDVWVXYC	69
100000	FXVIGMJKKM	94

```
Press to continue...
```

Запити для генерації даних:

companies:

Value:

```
    Insert into companies(company_name, country, owner)
    values(chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() *
25)::int) || chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() *
25)::int) || chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() *
25)::int) || chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() *
25)::int), chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() *
25)::int) || chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() *
25)::int) || chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() *
25)::int) || chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() *
25)::int), chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() *
25)::int) || chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() *
25)::int) || chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() *
25)::int) || chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() *
25)::int) || chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() *
25)::int))
```

medicines:

Value:

```
    Insert into medicines(medicine_name, company_id)
    values(chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() *
25)::int) || chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() *
25)::int) || chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() *
25)::int) || chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() *
25)::int),     (select foreigh_id
    from
        trunc(1 + random() * (select max(company_id)
    from companies)) as foreigh_id
    inner join companies on foreigh_id = companies.company_id)::bigint
)
```

patients:

Value:

```
    Insert into patients(fullname, age)
    values(chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() *
25)::int) || chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() *
25)::int) || chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() *
25)::int) || chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() *
25)::int), 1 + random() * 100)
```

pharmacies:

Value:

```
    Insert into pharmacies(pharmacy_name, city)
    values(chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() *
25)::int) || chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() *
25)::int) || chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() *
25)::int) || chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() *
25)::int) || chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() *
25)::int), chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() *
25)::int) || chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() *
25)::int) || chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() *
25)::int) || chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() *
25)::int) || chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() *
25)::int))
```

Ілюстрації уведення пошукового запиту та результатів виконання запитів

companies:

company_id	company_name	country	owner	production_count
2	test1	test1	test1	2
5	Файзер	Німеччина	Чарльз Файзер	3
3	IHearb	USA	Ray Faraee	2
6	LQPUXWNWST	LONBWNWGGV	VXJBCPSOFK	2
7	ASJOEUKPWG	NXYFKLTNWT	DDCUCNFBEM	0
8	VRDNPNCJCV	MGIJNOHNAC	IFOEAEDCDG	3
9	FJATCBYXHJ	YFOLBEPHTX	IQLUOCSYJQ	2
10	LJFWVYQMPC	MVWWHKIBPB	IVMJPQSGAJ	0
11	COWYBIJIYW	EFJTSQQYFI	UVYDAPMYQS	1
12	UYKVPMCPVQ	OQVYIMCNKC	RADLOPRTOJ	1
13	EWQIXVPKRS	YNTVPQBKFR	QRLXNNDAEJ	0
14	FLOJVLBTJJ	YWQFQDHNYK	OXGPLYETIP	0
15	JYLGJCDYDB	YHLLKIQQWI	HWPLDJGSIU	0
16	APKLNXPVVI	ORJNLMRUHA	FNSHLDDINB	0
17	OSBVLEUHIV	EVKYMEBAOT	AISEUDMMRM	0
18	KSFQLWTVCC	ELMMPXPFUP	DKHIPFKJSK	2
19	SYHVSRQFXT	UUUUJBBKFH	UXTWESTVRF	1
20	ETJFFECQWN	MEDTBENREW	TACNADKAVI	1
21	CFDYUMQNIB	AKKIDSNCEM	BQOAWIGGIX	0
22	NWVSQIEEDW	UEIEXWVLRI	QPTYHJXWOA	0
23	JXVNATHANG	DRSFQRNFVS	EBVPMWTQBE	1
24	PMXYCIAHIO	OPMYHEAVXO	SKNBIECXIG	1

```
"""
    select
        count(medicine_id)
    from
        companies
    full join medicines using (company_id)
    where
        company_id = @id
    having
        count(medicine_id) between @a and @b
    order by
        count(medicine_id)
""";
```

medicines:

```
Enter like expression: %D
```

medicines

medicine_id	medicine_name	company_id	company_name
1	MDHYAGTQUM	2	WWVDXYTXRE
2	CNIUAUKYHA	2	WWVDXYTXRE
3	PNSYOPURMP	5	WWVDXYTXRE
4	LTIMABKGKH	3	WWVDXYTXRE
5	BUJCCLPULS	5	WWVDXYTXRE
6	ESSBGKTBVS	6	WWVDXYTXRE
7	KMOVTGYNKS	5	WWVDXYTXRE
8	PPGUJQFBRB	11	WWVDXYTXRE
9	EHQVHWVTAD	8	WWVDXYTXRE
10	POIVWHGHMY	12	WWVDXYTXRE
12	Цитрамон Дарниця	6	WWVDXYTXRE
13	LMSVPWVLXM	405	WWVDXYTXRE
14	SPJMMKPSVW	559	WWVDXYTXRE
15	MWHHXBPJHR	683	WWVDXYTXRE
16	IIXIRPYPOC	661	WWVDXYTXRE
17	XUKRIAUCUP	371	WWVDXYTXRE
18	GFYMDFQLFX	1002	WWVDXYTXRE
19	UKVJKFNVGL	305	WWVDXYTXRE
20	FASMMFXMIW	61	WWVDXYTXRE
21	QVDFLQJFUO	234	WWVDXYTXRE
22	TYQCVAVMYT	25	WWVDXYTXRE
23	ADKLKUKIDF	347	WWVDXYTXRE
24	EQOECLAONR	639	WWVDXYTXRE
25	CKIBMGKNRT	954	WWVDXYTXRE

```
select
    companies.company_id as Id,
    company_name as Name,
    country as Country,
    owner as Owner
from
    medicines
full join companies using (company_id)
where
    company_name like 'W%'
order by
    companies.company_id
```

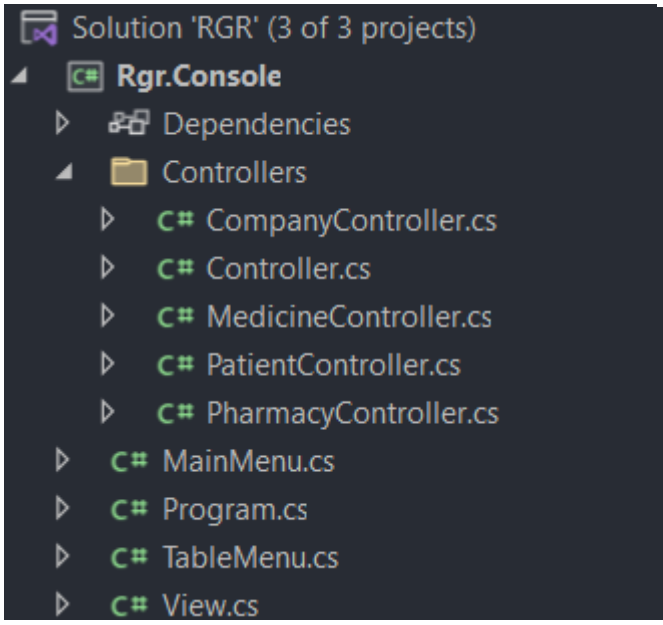
patients:

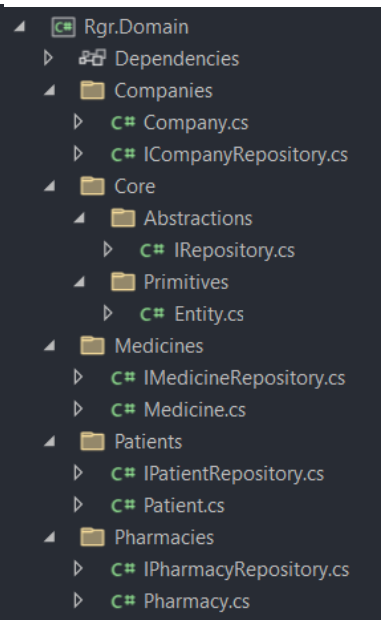
99973	EIBGMOHUPS	64	0
99974	NIVSJFURUR	88	0
99975	AFYSEUSXQC	58	0
99976	QRQJPYJRQR	62	0
99977	REMOFPMQGC	71	0
99978	OFPNVPBNFG	14	0
99979	OTMSBMAMTX	48	0
99980	ETADHGOCHY	53	0
99981	BRFKLBRXIF	10	0
99982	OGPYAQLNMS	24	0
99983	OOWKJESAST	41	0
99984	EMBKONMEGO	65	0
99985	YOTMELGMAN	76	0
99986	MFWHQJMNQY	65	0
99987	QHRMIRCBRE	71	0
99988	XBTTKIPGTC	97	0
99989	NECKTDRNGM	86	0
99990	PKOABSNNBL	47	0
99991	PWGSJIEWIB	99	0
99992	UKSGKSGBGS	92	0
99993	BNJLEYFMWG	89	0
99994	FQIMFGOWDP	79	0
99995	ACCQMXYFKT	17	0
99996	EIQPKLPKCV	38	0
99997	NJEPAXRLYR	100	0
99998	DJERQVRILO	92	0
99999	HWLDVWVXYC	69	0
100000	FXVIGMJKKM	94	0

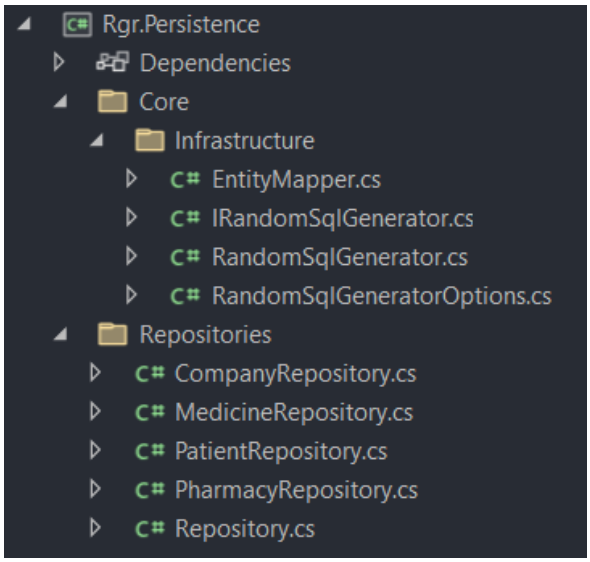
```
Press to continue...
```

```
select
    count(pharmacy_id)
from
    patients
full join pharmacy_patients using (patient_id)
where
    patient_id = 10
group by
    patient_id
```

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