МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

Лабораторна робота №2

з дисциплін «Бази даних та засоби управління» та «Бази даних»

тема “Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL”

Виконав(ла)

студент(ка) ІІ курсу

групи КП-01

Вишневецька Ольга Денисівна

Київ 2021

Мета роботи
Здобуття вмінь програмування прикладних додатків баз даних PostgreSQL.

Загальне завдання роботи:
1. Реалізувати функції внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі No1, 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).

Текст програми:

Model:

```
using static System.Console;
using System.Collections.Generic;
using Npgsql;
using System;

public class Course
{
   public long id;
   public string name;

   public Course()
   {
       this.id = 0;
       this.name = "";
   }

   public Course(string name)
   {
       this.name = name;
   }
```

```
    public override string ToString()
    {
        return $"[{id}] {name}";
    }
}

public class Student
{
    public long id;
    public long course;
    public string name;
    public string surname;

    public Student()
    {
        this.id = 0;
        this.course = 0;
        this.name = "";
        this.surname = "";
    }

    public Student(long course, string name, string surname)
    {
        this.course = course;
        this.name = name;
        this.surname = surname;
    }

    public override string ToString()
    {
        return $"[{id}] {name} {surname}";
    }
}

public class Lecture
{
    public long id;
    public long course;
    public string topic;
    public long duration;
    public string text;
```

```
    public Lecture()
    {
        this.id = 0;
        this.course = 0;
        this.topic = "";
        this.duration = 0;
        this.text = "";
    }

    public Lecture(long course, string topic, long duration, string
text)
    {
        this.course = course;
        this.topic = topic;
        this.duration = duration;
        this.text = text;
    }

    public override string ToString()
    {
        return $"[{id}] [{course}] {topic} - {duration} \n{text}";
    }
}

public class CourseRep
{
    private NpgsqlConnection connection;

    public CourseRep(NpgsqlConnection connection)
    {
        this.connection = connection;
    }

    public long Insert(Course course)//+
    {
        connection.Open();

        var sql =
        @"INSERT INTO courses (name)
        VALUES (@name);";
        using var command = new NpgsqlCommand(sql, connection);
        command.Parameters.AddWithValue("@name", course.name);
```

```
        long lastId = command.ExecuteNonQuery();

        connection.Close();

        return lastId;
    }

    public bool DeleteById(long id)//++
    {
        connection.Open();

        var sql = @"DELETE FROM courses WHERE id = @id";
        using var command = new NpgsqlCommand(sql, connection);
        command.Parameters.AddWithValue("@id", id);
        int nChanged = command.ExecuteNonQuery();
        connection.Close();
        if (nChanged == 0)
        {
            return false;
        }

        return true;
    }

    public bool Update(long id, Course course)//+
    {
        connection.Open();

        var sql = @"UPDATE courses SET name = @name WHERE id = @id";
        using var command = new NpgsqlCommand(sql, connection);
        command.Parameters.AddWithValue("@id", id);
        command.Parameters.AddWithValue("@name", course.name);
        int rowChange = command.ExecuteNonQuery();
        connection.Close();
        if (rowChange == 0)
        {
            return false;
        }

        return true;
    }
```

```
    public Course GetCourse(NpgsqlDataReader reader)//+
    {
        Course course = new Course();
        course.id = reader.GetInt32(0);
        course.name = reader.GetString(1);

        return course;
    }
}

public class StudentRep
{
    private NpgsqlConnection connection;

    public StudentRep(NpgsqlConnection connection)
    {
        this.connection = connection;
    }

    public long Insert(Student student)//+
    {
        connection.Open();
        var sql =
        @"INSERT INTO students (course, name, surname)
        VALUES (@course, @name, @surname);";

        using var command = new NpgsqlCommand(sql, connection);

        command.Parameters.AddWithValue("@course", student.course);
        command.Parameters.AddWithValue("@name", student.name);
        command.Parameters.AddWithValue("@surname", student.surname);

        long lastId =  command.ExecuteNonQuery();
        connection.Close();

        return lastId;
    }

    public bool DeleteById(long id)//+
    {
        connection.Open();
```

```
        var sql = @"DELETE FROM students WHERE id = @id";
        using var command = new NpgsqlCommand(sql, connection);
        command.Parameters.AddWithValue("@id", id);
        int nChanged = command.ExecuteNonQuery();
        connection.Close();
        if (nChanged == 0)
        {
            return false;
        }

        return true;
    }

    public bool Update(long id, Student student)//+
    {
        connection.Open();

        var sql = @"UPDATE students SET course = @course, name =
@name, surname = @surname  WHERE id = @id";
        using var command = new NpgsqlCommand(sql, connection);
        command.Parameters.AddWithValue("@id", student.id);
        command.Parameters.AddWithValue("@course", student.course);
        command.Parameters.AddWithValue("@name", student.name);
        command.Parameters.AddWithValue("@surname", student.surname);
        int rowChange = command.ExecuteNonQuery();
        connection.Close();
        if (rowChange == 0)
        {
            return false;
        }

        return true;
    }

    public Student GetStudent(NpgsqlDataReader reader)//+
    {
        Student student = new Student();
        student.id = reader.GetInt32(0);
        student.course = reader.GetInt32(1);
        student.name = reader.GetString(2);
        student.surname = reader.GetString(3);
        return student;
```

```
    }
}

public class LectureRep
{
    private NpgsqlConnection connection;

    public LectureRep(NpgsqlConnection connection)
    {
        this.connection = connection;
    }

    public long Insert(Lecture lecture)//+
    {
        connection.Open();
        var sql =
        @"INSERT INTO lectures (course, topic, duration, text)
        VALUES (@course, @topic, @duration, @text);";
        using var command = new NpgsqlCommand(sql, connection);
        command.Parameters.AddWithValue("@course", lecture.course);
        command.Parameters.AddWithValue("@topic", lecture.topic);
        command.Parameters.AddWithValue("@duration",
lecture.duration);
        command.Parameters.AddWithValue("@text", lecture.text);
        long lastId = command.ExecuteNonQuery();
        connection.Close();

        return lastId;
    }

    public Lecture GetLecture(NpgsqlDataReader reader)//+
    {
        Lecture lecture = new Lecture();
        lecture.id = reader.GetInt32(0);
        lecture.course = reader.GetInt32(1);
        lecture.topic = reader.GetString(2);
        lecture.duration = reader.GetInt32(3);
        lecture.text = reader.GetString(4);

        return lecture;
    }
```

```
    public bool DeleteById(long id)//+
    {
        connection.Open();

        var sql = @"DELETE FROM lectures WHERE id = @id";
        using var command = new NpgsqlCommand(sql, connection);
        command.Parameters.AddWithValue("@id", id);
        int nChanged = command.ExecuteNonQuery();
        connection.Close();

        if (nChanged == 1)
        {
            return true;
        }
        return false;
    }

    public bool Update(long id, Lecture lecture)//+
    {
        connection.Open();

        var sql = @"UPDATE lectures SET course = @course, topic =
@topic, duration = @duration, text = @text WHERE id = @id";
        using var command = new NpgsqlCommand(sql, connection);
        command.Parameters.AddWithValue("@id", id);
        command.Parameters.AddWithValue("@course", lecture.course);
        command.Parameters.AddWithValue("@topic", lecture.topic);
        command.Parameters.AddWithValue("@duration",
lecture.duration);
        command.Parameters.AddWithValue("@text", lecture.text);
        int nChanged = command.ExecuteNonQuery();
        connection.Close();
        if (nChanged == 1)
        {
            return true;
        }
        return false;
    }
}

public class GenerateData
{
```

```
    public void GenerateCourse(int number, NpgsqlConnection
connection)//+
    {
        string[] names = new string[] {"Maths", "Analitics",
"English", "IT", "Statistics"};
        Random random = new Random();
        CourseRep coursesRep = new CourseRep(connection);
        for (int i = 0; i < number; i++)
        {
            Course course = new Course();
            course.name = names[random.Next(0, names.Length - 1)];
            coursesRep.Insert(course);
        }

    }
}
```

View:

```
using static System.Console;
using System.Collections.Generic;
using Npgsql;
using System;

public class View
{
    public void SendHelp()
    {
        WriteLine("Your possible options are: insert, get, update,
delete or generate.");
    }

    public void SendError()
    {
        WriteLine("Error! You didn't enter the correct value!");
    }

    public void PrintInsertStudent(long id)
    {
        WriteLine("Created new student with id: {0}", id);
    }
    public void PrintInsertCourse(long id)
```

```
{
    WriteLine("Created new course with id: {0}", id);
}
public void PrintInsertLecture(long id)
{
    WriteLine("Created new lecture with id: {0}", id);
}

public void PrintGetStudent(Student st)
{
    WriteLine("Got: " + st.ToString());
}
public void PrintGetCourse(Course c)
{
    WriteLine("Got: " + c.ToString());
}
public void PrintGetLecture(Lecture l)
{
    WriteLine("Got: " + l.ToString());
}

public void PrintUpdateStudent(bool check)
{
    if(check)
    {
        WriteLine("Student was successfully updated!");
    }
    else
    {
        WriteLine("Student couldn't update.");
    }
}
public void PrintUpdateCourse(bool check)
{
    if(check)
    {
        WriteLine("Course was successfully updated!");
    }
    else
    {
        WriteLine("Course couldn't update.");
    }
```

```
    }
    public void PrintUpdateLecture(bool check)
    {
        if(check)
        {
            WriteLine("Lecture was successfully updated!");
        }
        else
        {
            WriteLine("Lecture couldn't update.");
        }
    }

    public void PrintDeleteStudent(bool check)
    {
        if(check)
        {
            WriteLine("Student was successfully deleted!");
        }
        else
        {
            WriteLine("Student wasn't deleted.");
        }
    }
    public void PrintDeleteCourse(bool check)
    {
        if(check)
        {
            WriteLine("Course was successfully deleted!");
        }
        else
        {
            WriteLine("Course wasn't deleted.");
        }
    }
    public void PrintDeleteLecture(bool check)
    {
        if(check)
        {
            WriteLine("Lecture was successfully deleted!");
        }
        else
```

```
        {
            WriteLine("Lecture wasn't deleted.");
        }
    }
}
```

Controller:

```
using static System.Console;
using System.Collections.Generic;
using Npgsql;
using System;

namespace test
{
    class Program
    {
        static void Main(string[] args)
        {
            string db_path = "Server=localhost; Port=5432; User
Id=postgres; Password=123456; Database=db"; //@"/home/katrin/Рабочий
стол/КПИ/progbase3/Progbase3.sln/ConsoleProject/data/Database.db";
            using var connection = new NpgsqlConnection(db_path);

            CourseRep courserep = new CourseRep(connection);
            LectureRep lecturerep = new LectureRep(connection);
            StudentRep studentrep = new StudentRep(connection);

            string command;
            string type;
            do
            {
                WriteLine("What do you want to do?: \nEnter help to
see possible options.");
                command = ReadLine();
                View v = new View();
                if (command == "insert")
                {
                    WriteLine("Choose and type in: student, course or
lecture");
                    type = ReadLine();
                    ProcessInsert prin = new ProcessInsert();
```

```
                    switch(type)
                    {
                        case "student":
prin.ProcessInsertStudent(studentrep); break;
                        case "course":
prin.ProcessInsertCourse(courserep); break;
                        case "lecture":
prin.ProcessInsertLecture(lecturerep); break;
                        default: v.SendError(); break;
                    }
                }
                else if (command == "get")
                {
                    WriteLine("Choose and type in: student, course or
lecture");
                    type = ReadLine();
                    NpgsqlCommand cmd = connection.CreateCommand();
                    connection.Open();
                    NpgsqlDataReader reader = cmd.ExecuteReader();
                    ProcessGet prg = new ProcessGet();
                    while (reader.Read())
                    {
                        switch(type)
                        {
                            case "student":
prg.ProcessGetStudent(studentrep, reader); break;
                            case "course":
prg.ProcessGetCourse(courserep, reader); break;
                            case "lecture":
prg.ProcessGetLecture(lecturerep, reader); break;
                            default: v.SendError(); break;
                        }
                    }
                }
                else if (command == "update")
                {
                    WriteLine("Choose and type in: student, course or
lecture");
                    type = ReadLine();
                    ProcessUpdate prup = new ProcessUpdate();
                    switch(type)
                    {
```

```
                        case "student":
prup.ProcessUpdateStudent(studentrep); break;
                        case "course":
prup.ProcessUpdateCourse(courserep); break;
                        case "lecture":
prup.ProcessUpdateLecture(lecturerep); break;
                        default: v.SendError(); break;
                    }
                }
                else if (command == "delete")
                {
                    WriteLine("Choose and type in: student, course or
lecture");
                    type = ReadLine();
                    ProcessDelete prdel = new ProcessDelete();
                    switch(type)
                    {
                        case "student":
prdel.ProcessDeleteStudent(studentrep); break;
                        case "course":
prdel.ProcessDeleteCourse(courserep); break;
                        case "lecture":
prdel.ProcessDeleteLecture(lecturerep); break;
                        default: v.SendError(); break;
                    }
                }
                else if (command == "generate")
                {
                    ProcessGeneration pg = new ProcessGeneration();
                    pg.ProcessGenerate(connection);
                }
                else if(command == "help")
                {
                    v.SendHelp();
                }
                else if (command != "quit")
                {
                    WriteLine("{0} not found. Try another command.",
command);
                }

            } while(command !=  "quit");
```

```
            WriteLine("Bye! :)");
        }
    }

    public class ProcessGeneration
    {
        public void ProcessGenerate(NpgsqlConnection connection)
        {
            WriteLine("How many do you want to generate? Enter n:");
            string temp = ReadLine();
            if(int.TryParse(temp, out int n))
            {
                GenerateData g = new GenerateData();
                g.GenerateCourse(n, connection);
            }
            else
            {
                View v = new View();
                v.SendError();
            }
        }
    }

    public class ProcessInsert
    {
        public void ProcessInsertStudent(StudentRep studentrep)
        {
            WriteLine("Type in the id of students course:");
            string temp = ReadLine();
            View v = new View();
            if(!(Int64.TryParse(temp, out long course)))
            {
                v.SendError();
            }
            else
            {
                WriteLine("Type in students name:");
                string name = ReadLine();
                WriteLine("Type in students surname: ");
                string surname = ReadLine();

                Student st = new Student(course, name, surname);
```

```
                long newId = studentrep.Insert(st);
                v.PrintInsertStudent(newId);
            }
        }

        public void ProcessInsertCourse(CourseRep courserep)
        {
            WriteLine("Type in name of the course:");
            string name = ReadLine();

            Course c = new Course(name);
            long newId = courserep.Insert(c);
            View v = new View();
            v.PrintInsertCourse(newId);
        }

        public void ProcessInsertLecture(LectureRep lecturerep)
        {
            WriteLine("Type in the id of course the lecture will be
for:");
            string temp = ReadLine();
            View v = new View();
            if(!(Int64.TryParse(temp, out long course)))
            {
                v.SendError();
            }
            else
            {
                WriteLine("Type in topic of the lecture:");
                string topic = ReadLine();
                WriteLine("Type in duration of the lecture:");
                temp = ReadLine();
                if(!(Int64.TryParse(temp, out long duration)))
                {
                    v.SendError();
                }
                else
                {
                    WriteLine("Type in lectures text: ");
                    string text = ReadLine();
```

```
                    Lecture l = new Lecture(course, topic, duration,
text);
                    long newId = lecturerep.Insert(l);
                    v.PrintInsertLecture(newId);
                }
            }
        }
    }

    public class ProcessGet
    {
        public void ProcessGetStudent(StudentRep studentrep,
NpgsqlDataReader reader)
        {
            Student st = studentrep.GetStudent(reader);
            View v = new View();
            v.PrintGetStudent(st);
        }
        public void ProcessGetCourse(CourseRep courserep,
NpgsqlDataReader reader)
        {
            Course c = courserep.GetCourse(reader);
            View v = new View();
            v.PrintGetCourse(c);
        }
        public void ProcessGetLecture(LectureRep lecturerep,
NpgsqlDataReader reader)
        {
            Lecture l = lecturerep.GetLecture(reader);
            View v = new View();
            v.PrintGetLecture(l);
        }
    }

    public class ProcessUpdate
    {
        public void ProcessUpdateStudent(StudentRep studentrep)
        {
            WriteLine("Type in the id of student you want to
update:");
            string temp = ReadLine();
            View v = new View();
```

```
            if(!(Int64.TryParse(temp, out long id)))
            {
                v.SendError();
            }
            else
            {
                WriteLine("Type in new id of students course:");
                temp = ReadLine();
                if(!(Int64.TryParse(temp, out long course)))
                {
                    v.SendError();
                }
                else
                {
                    WriteLine("Type in new students name:");
                    string name = ReadLine();
                    WriteLine("Type in new students surname: ");
                    string surname = ReadLine();

                    Student st = new Student(course, name, surname);
                    bool check = studentrep.Update(id, st);
                    v.PrintUpdateStudent(check);
                }
            }
        }

        public void ProcessUpdateCourse(CourseRep courserep)
        {
            WriteLine("Type in the id of course you want to update:");
            string temp = ReadLine();
            View v = new View();
            if(!(Int64.TryParse(temp, out long id)))
            {
                v.SendError();
            }
            else
            {
                WriteLine("Type in new name of the course:");
                string name = ReadLine();

                Course c = new Course(name);
                bool check = courserep.Update(id, c);
```

```
                v.PrintUpdateCourse(check);
            }
        }

        public void ProcessUpdateLecture(LectureRep lecturerep)
        {
            WriteLine("Type in the id of lecture you want to
update:");
            string temp = ReadLine();
            View v = new View();
            if(!(Int64.TryParse(temp, out long id)))
            {
                v.SendError();
            }
            else
            {
                WriteLine("Type in new id of course the lecture will
be for:");
                temp = ReadLine();
                if(!(Int64.TryParse(temp, out long course)))
                {
                    v.SendError();
                }
                else
                {
                     WriteLine("Type in new topic of the lecture:");
                    string topic = ReadLine();
                    WriteLine("Type in new duration of the lecture:");
                    temp = ReadLine();
                    if(!(Int64.TryParse(temp, out long duration)))
                    {
                        v.SendError();
                    }
                    else
                    {
                        WriteLine("Type in new lectures text: ");
                        string text = ReadLine();

                        Lecture l = new Lecture(course, topic,
duration, text);
                        bool check = lecturerep.Update(id, l);
                        v.PrintUpdateLecture(check);
```

```
                    }
                }
            }
        }
    }

    public class ProcessDelete
    {
        public void ProcessDeleteStudent(StudentRep studentrep)
        {
            WriteLine("Type in the id of student you want to
delete:");
            string temp = ReadLine();
            View v = new View();
            if(!(Int64.TryParse(temp, out long id)))
            {
                v.SendError();
            }
            else
            {
                bool check = studentrep.DeleteById(id);
                v.PrintDeleteStudent(check);
            }
        }

        public void ProcessDeleteCourse(CourseRep courserep)
        {
            WriteLine("Type in the id of course you want to delete:");
            string temp = ReadLine();
            View v = new View();
            if(!(Int64.TryParse(temp, out long id)))
            {
                v.SendError();
            }
            else
            {
                bool check = courserep.DeleteById(id);
                v.PrintDeleteCourse(check);
            }
        }

        public void ProcessDeleteLecture(LectureRep lecturerep)
```

```
        {
            WriteLine("Type in the id of lecture you want to
delete:");
            string temp = ReadLine();
            View v = new View();
            if(!(Int64.TryParse(temp, out long id)))
            {
                v.SendError();
            }
            else
            {
                bool check = lecturerep.DeleteById(id);
                v.PrintDeleteLecture(check);
            }
        }
    }
}
```

Можливі результати(вимоги до деталізованих завдань):

1.

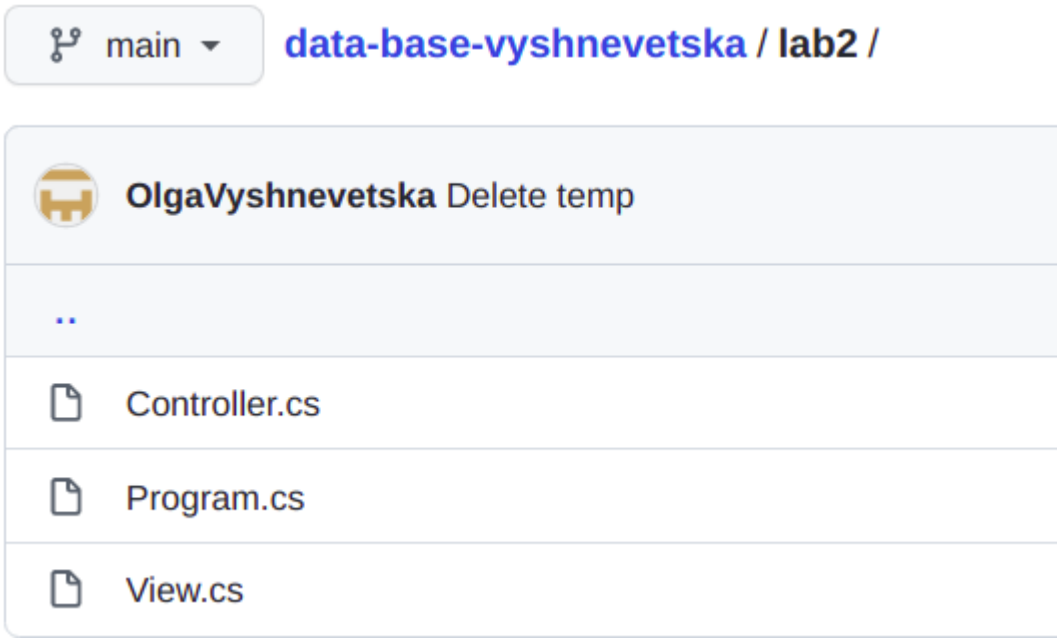
```
volcha@volcha:~/Рабочий стол/KPI/БД/lab/MyAppSolution/ConsoleProject$ What do you want to do?:
Enter help to see possible options.
insert
Choose and type in: student, course or lecture
student
Type in the id of students course:
3
Type in students name:
testing
Type in students surname:
student
Created new student with id: 1
What do you want to do?:
Enter help to see possible options.
create
create not found. Try another command.
What do you want to do?:
Enter help to see possible options.
insert
Choose and type in: student, course or lecture
student
Type in the id of students course:
id
Error! You didn't enter the correct value!
What do you want to do?:
Enter help to see possible options.
insert
Choose and type in: student, course or lecture

Error! You didn't enter the correct value!
```

2.

```
1  SELECT * FROM courses
```

Data Output | Explain | Messages | Notifications

	id [PK] integer	name text
1	1	IT
2	2	English
3	3	Analitics
4	4	Maths
5	5	IT
6	6	Analitics
7	7	Analitics
8	8	English
9	9	English
10	10	Maths
11	11	Maths
12	12	IT
13	13	Maths
14	14	English
15	15	Maths
16	16	Maths
17	17	English
18	18	Maths

4. 

Висновки:

Я здобула вміння програмування прикладних додатків баз даних PostgreSQL.