Дисципліни «Бази даних та засоби управління» та
«Бази даних». Осінь 2021 року

Лабораторна робота №2

Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL

Варіант №8: База даних електронних курсів

Виконав:
студент 2 курсу ФПМ,
групи КП-03,
Мишко Роман

Київ 2021

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

Загальне завдання роботи полягає у наступному:

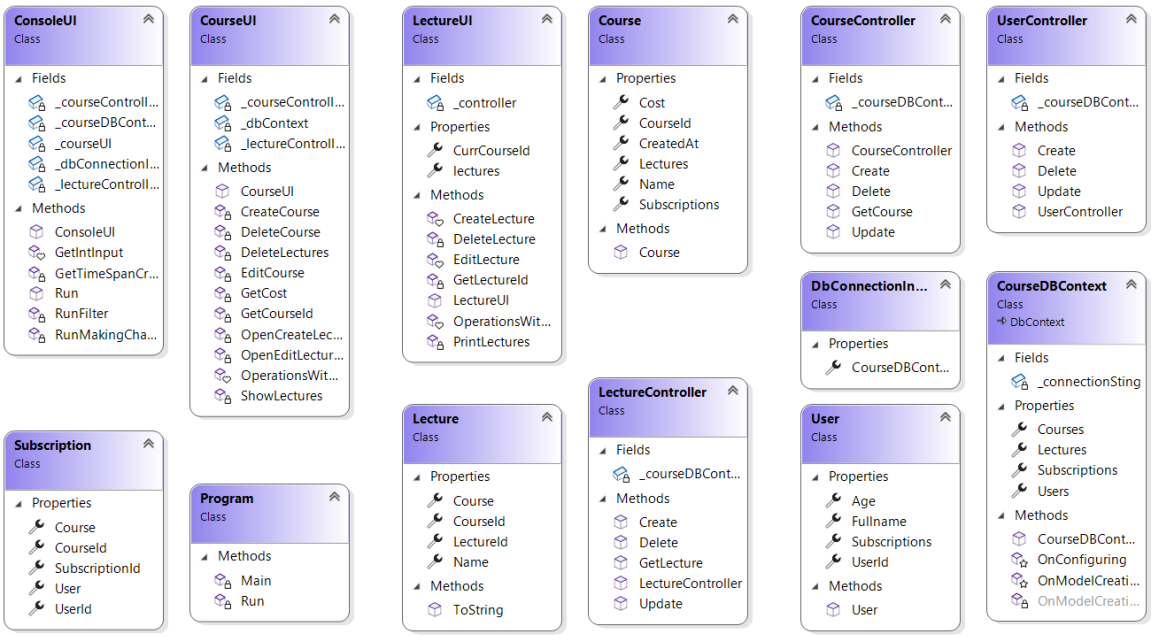
1. Реалізувати функції внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі No1, засобами консольного інтерфейсу.
2. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.
3. Забезпечити реалізацію пошуку за декількома атрибутами з двох та 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).

Деталізоване завдання:

1. Забезпечити можливість введення/редагування/вилучення даних у таблицях бази даних з можливістю контролю відповідності типів даних атрибутів таблиць (рядків, чисел, дати/часу). Для контролю пропонується два варіанти: контроль при введенні (валідація даних) та перехоплення помилок (try..except) від сервера PostgreSQL при виконанні відповідної команди SQL. Особливу увагу варто звернути на дані таблиць, що мають зв’язок 1:N. При цьому з боку батьківської таблиці необхідно контролювати вилучення рядків за умови наявності даних у підлеглій таблиці. З точки зору підлеглої таблиці варто контролювати наявність відповідного рядка у батьківській таблиці при виконанні внесення нових даних. Унеможливити виведення програмою системних помилок на екрані шляхом їх перехоплення і адекватної обробки. Внесення даних виконується користувачем у консольному вікні програми.

2. Забезпечити можливість автоматичної генерації великої кількості даних у таблицях за допомогою вбудованих у PostgreSQL функцій роботи з псевдовипадковими числами. Дані мають бути згенерованими не мовою програмування, а відповідним SQL-запитом!
3. Для реалізації пошуку необхідно підготувати 3 запити, що включають дані з декількох таблиць і фільтрують рядки за 3-4 атрибутами цих таблиць. Забезпечити можливість уведення конкретних значень констант для фільтрації з клавіатури користувачем. Крім того, після виведення даних необхідно вивести час виконання запиту у мілісекундах. Перевірити швидкодію роботи запитів на попередньо згенерованих даних.
4. Програмний код організувати згідно шаблону Model-View-Controller(MVC). При цьому модель, подання та контролер мають бути реалізовані у окремих файлах. Для доступу до бази даних використовувати лише мову SQL (без ORM).

Результати роботи



1.1. Ілюстрації обробки виняткових ситуацій (помилок) при введенні/вилучення даних.

```
Console.WriteLine("Write cost of course\r\n\r\n write \"back\" to step back");

try
{
    cost = ConsoleUI.GetIntInput();
}
catch (ArgumentException)
{
    continue;
}
catch (Exception)
{
    return -1;
}
```

1.2. Ілюстрації валідації даних при введенні користувачем.

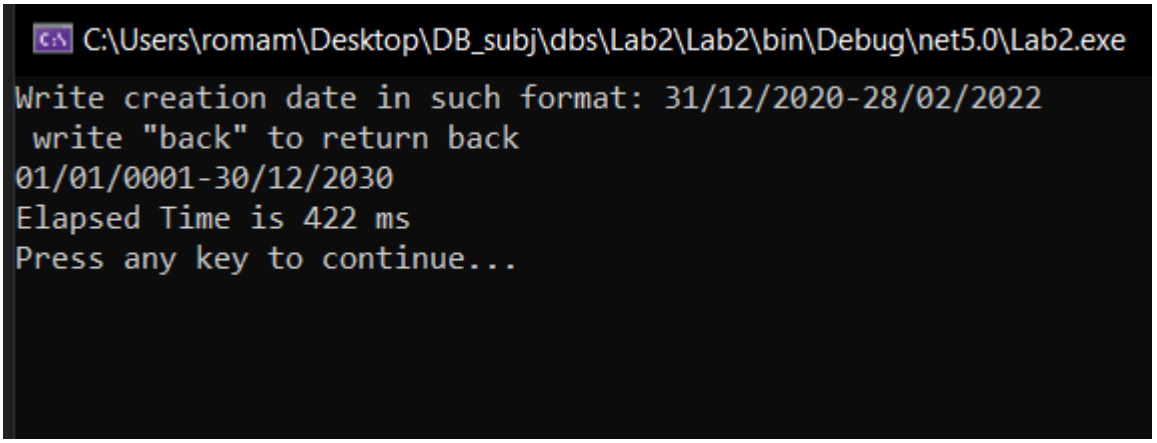
```
7 references | Roman Myshko, 12 hours ago | 1 author, 1 change
internal static int GetIntInput()
{
    string entered = Console.ReadLine();

    if (entered.Equals("exit"))
    {
        Console.WriteLine("Ending session...");
        Environment.Exit(0);
    }
    else if (entered.Equals("back"))
    {
        throw new Exception();
    }

    int option;
    bool isInt = int.TryParse(entered, out option);

    if (!isInt)
    {
        Console.Clear();
        throw new ArgumentException(entered);
    }

    return option;
}
```

2. Копії екрану (ілюстрації) з фрагментами згенерованих даних таблиць.

	name text	created_at date	cost integer	course_id [PK] integer
87	8bdc1690b715d255a0be896e703cc94f	2024-11-09	57	500064
88	54b01b9dad551802bc6f2e36b2698991	2024-05-11	73	500063
89	6cf7af4b43e446964534b83f8a8e2986	2025-05-11	39	500062
90	4fb639876a17e8fae1fc8c2c1638f22a	2025-05-11	23	500061
91	38e4ece696abed15483b68c7d733614e	2022-05-13	21	500060
92	2d4c683fe5f1a924789e32c34cab03b3	2025-05-11	19	500059
93	796179163b3271411f55cdbeba2416a2	2024-05-11	99	500058
94	546ba31db76a874b43b383868408bbb1	2024-05-11	22	500057
95	5500a7a244037213a950ec301b0a47da	2023-05-14	12	500056
96	08c90d192f6c0877bacc4d4b51cdb9cc	2024-05-11	84	500055
97	564841b1a7ad8caccbe65e93290d06fd	2024-05-11	1	500054
98	9eae951816dcf6495595c3e8ec04ee9f	2025-05-11	83	500053
99	d92ef3524878b4c7f47a9b1c1c6d6158	2024-11-09	36	500052
100	7841f76baf6947fde3b5997db0f54fcd	2024-11-09	65	500051

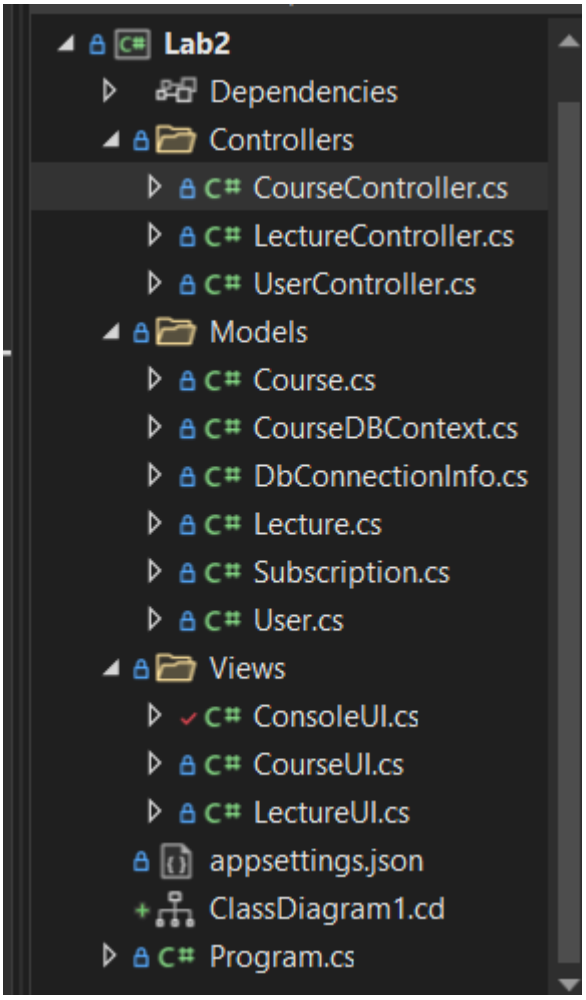
	user_id [PK] integer	fullname text	age integer
45	199983	ff7beb35eb28ced22d4ba8a9ebbdcc2c	40
46	199982	032416d555aad3af468c1a25dbf16e50	59
47	199981	56dfb65ef313cceb9afd5e076e84b550	31
48	199980	ebca7cb4e1bfc133e1120701c1c36323	25
49	199979	529cd054697048d6f7d4ade4952c3c5c	63
50	199978	95213bd67354cc482be207333065aec2	15
51	199977	a27573cdef9bbb285212a5d16d33cc44	23
52	199976	4f135dcac971705df89ffee25f1a2bb2	24
53	199975	9c4f4ef1e3a78289738d5b721701e21f	16
54	199974	9838f8654e95c07d983cb3e28de1c112	64
55	199973	7e494c3514e1603e53a51097aab356c1	34
56	199972	f6be633f6cd00efc6cddb310df194070	62
57	199971	df755ddd0c0aee98ebca882302acbe18	38
58	199970	8edaaa45c89cb460b883412eacaf4cad	58
59	199969	7959e06c0fa15c7dec8b9a7496a0ef67	42

3. Ілюстрації введення пошукового запиту та результатів виконання запитів.




```
e(course => course.Name.Contains("a") &&
  course.Cost < 50 && course.CreatedAt >= fromTime
  && course.CreatedAt <= toTime);
```

4. Ілюстрації програмного коду з репозиторію Git.



Program.cs

```
using System;
using Lab2.Models;
using Lab2.Views;
using Lab2.Controllers;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata;
using Microsoft.Extensions.Options;
using Microsoft.Extensions.Configuration;

namespace Lab2
{
    internal class Program
    {
        static void Main(string[] args)
        {
                    const string databaseConnection = "Server=127.0.0.1;Port=5432;Database=student01_DB;User
Id=postgres;Password=1;Include Error Detail=True";

            Run(databaseConnection);
        }

        static void Run(string databaseConnection)
        {
            CourseDBContext _courseDBContext;
            DbConnectionInfo _dbConnectionInfo = new DbConnectionInfo();
```

```
            _dbConnectionInfo.CourseDBContext = databaseConnection;
            _courseDBContext = new CourseDBContext(_dbConnectionInfo);

            ConsoleUI console = new ConsoleUI(databaseConnection);
            console.Run();
        }
    }
}
```

CourseController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Lab2.Models;
using System.Threading.Tasks;

namespace Lab2.Controllers
{
    public class CourseController
    {
        private readonly CourseDBContext _courseDBContext;

        public CourseController(CourseDBContext courseDBContext)
        {
            _courseDBContext = courseDBContext;
        }

        public int Create(Course course)
        {
            try
            {
                _courseDBContext.Add(course);
                return _courseDBContext.SaveChanges();
            }
            catch (Exception)
            {
                return -1;
            }
        }

        public int Update(Course updatedCourse)
        {
            try
            {
                _courseDBContext.Update(updatedCourse);
                 return _courseDBContext.SaveChanges();
            }
            catch
            {
                return -1;
            }
        }

        public Course GetCourse(int id)
        {
            return _courseDBContext.Courses.Find(id);
        }

        public int Delete(int? id)
        {
            if (id == null)
            {
                return 0;
            }

            try
            {
                Course course = _courseDBContext.Courses.Find(id);
                _courseDBContext.Courses.Remove(course);
                return _courseDBContext.SaveChanges();
            }
            catch
            {
```

```
                return -1;
            }
        }
    }
}
```

LectureController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Lab2.Models;

namespace Lab2.Controllers
{
    public class LectureController
    {
        private readonly CourseDBContext _courseDBContext;

        public LectureController(CourseDBContext courseDBContext)
        {
            _courseDBContext = courseDBContext;
        }

        public int Create(Lecture lecture)
        {
            try
            {
                _courseDBContext.Add(lecture);
                return _courseDBContext.SaveChanges();
            }
            catch (Exception)
            {
                return -1;
            }
        }

        public int Update(Lecture updatedLecture)
        {
            try
            {
                _courseDBContext.Update(updatedLecture);
                return _courseDBContext.SaveChanges();
            }
            catch
            {
                return -1;
            }
        }

        public Lecture GetLecture(int id)
        {
            return _courseDBContext.Lectures.Find(id);
        }

        public int Delete(int? id)
        {
            if (id == null)
            {
                return 0;
            }

            try
            {
                Lecture lecture = _courseDBContext.Lectures.Find(id);
                _courseDBContext.Lectures.Remove(lecture);
                return _courseDBContext.SaveChanges();
            }
            catch
            {
                return -1;
            }
        }
    }
}
```

Course.cs

```
using System;
using System.Collections.Generic;

#nullable disable

namespace Lab2.Models
{
    public partial class Course
    {
        public Course()
        {
            Lectures = new HashSet<Lecture>();
            Subscriptions = new HashSet<Subscription>();
        }

        public int CourseId { get; set; }
        public string Name { get; set; }
        public DateTime? CreatedAt { get; set; }
        public int Cost { get; set; }

        public virtual ICollection<Lecture> Lectures { get; set; }
        public virtual ICollection<Subscription> Subscriptions { get; set; }
    }
}
```

CourseDBContext.cs

```
using System;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata;
using Microsoft.Extensions.Options;

#nullable disable

namespace Lab2.Models
{
    public partial class CourseDBContext : DbContext
    {
        private readonly string _connectionSting;

        public CourseDBContext()
        {
        }

        public CourseDBContext(DbContextOptions<CourseDBContext> options)
            : base(options)
        {
        }

        public CourseDBContext(DbConnectionInfo dbConnectionInfo)
        {
            _connectionSting = dbConnectionInfo.CourseDBContext;
        }

        public virtual DbSet<Course> Courses { get; set; }
        public virtual DbSet<Lecture> Lectures { get; set; }
        public virtual DbSet<Subscription> Subscriptions { get; set; }
        public virtual DbSet<User> Users { get; set; }

        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            if (!optionsBuilder.IsConfigured)
            {
                optionsBuilder.UseNpgsql(_connectionSting);
            }
        }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<Course>(entity =>
            {
                entity.ToTable("courses");
```

```
            entity.Property(e => e.CourseId)
                .UseSerialColumn()
                .HasColumnName("course_id");

            entity.Property(e => e.CreatedAt)
                .HasColumnType("date")
                .HasColumnName("created_at");

            entity.Property(e => e.Name)
                .IsRequired()
                .HasColumnName("name");

            entity.Property(e => e.Cost)
                .IsRequired()
                .HasColumnName("cost");
        });

        modelBuilder.Entity<Lecture>(entity =>
        {
            entity.ToTable("lectures");

            entity.Property(e => e.LectureId)
                .UseSerialColumn()
                .HasColumnName("lecture_id");

            entity.Property(e => e.CourseId).HasColumnName("course_id");

            entity.Property(e => e.Name)
                .IsRequired()
                .HasColumnName("name");

            entity.HasOne(d => d.Course)
                .WithMany(p => p.Lectures)
                .HasForeignKey(d => d.CourseId)
                .OnDelete(DeleteBehavior.ClientSetNull)
                .HasConstraintName("course_id");
        });

        modelBuilder.Entity<Subscription>(entity =>
        {
            entity.ToTable("subscriptions");

            entity.Property(e => e.SubscriptionId)
                .UseSerialColumn()
                .HasColumnName("subscription_id");

            entity.Property(e => e.CourseId).HasColumnName("course_id");

            entity.Property(e => e.UserId).HasColumnName("user_id");

            entity.HasOne(d => d.Course)
                .WithMany(p => p.Subscriptions)
                .HasForeignKey(d => d.CourseId)
                .OnDelete(DeleteBehavior.ClientSetNull)
                .HasConstraintName("course_id");

            entity.HasOne(d => d.User)
                .WithMany(p => p.Subscriptions)
                .HasForeignKey(d => d.UserId)
                .OnDelete(DeleteBehavior.ClientSetNull)
                .HasConstraintName("user_id");
        });

        modelBuilder.Entity<User>(entity =>
        {
            entity.ToTable("users");

            entity.Property(e => e.UserId)
                .UseSerialColumn()
                .HasColumnName("user_id");

            entity.Property(e => e.Age).HasColumnName("age");

            entity.Property(e => e.Fullname)
                .IsRequired()
                .HasColumnName("fullname");
        });
```

```
            OnModelCreatingPartial(modelBuilder);
        }

        partial void OnModelCreatingPartial(ModelBuilder modelBuilder);
    }
}
```

DbConnectionInfo.cs

```
#nullable disable

namespace Lab2.Models
{
    public class DbConnectionInfo
    {
        public string CourseDBContext { get; set; }
    }
}
```

Lecture.cs

```
using System;
using System.Collections.Generic;

#nullable disable

namespace Lab2.Models
{
    public partial class Lecture
    {
        public int LectureId { get; set; }
        public string Name { get; set; }
        public int CourseId { get; set; }

        public virtual Course Course { get; set; }

        public override string ToString()
        {
            return $"Lecture Id: {LectureId}, name: {Name}, course Id: {CourseId}";
        }
    }
}
```

Subscription.cs

```
using System;
using System.Collections.Generic;

#nullable disable

namespace Lab2.Models
{
    public partial class Subscription
    {
        public int SubscriptionId { get; set; }
        public int UserId { get; set; }
        public int CourseId { get; set; }

        public virtual Course Course { get; set; }
        public virtual User User { get; set; }
    }
}
```

User.cs

```
using System;
using System.Collections.Generic;

#nullable disable
```

```
namespace Lab2.Models
{
    public partial class User
    {
        public User()
        {
            Subscriptions = new HashSet<Subscription>();
        }

        public int UserId { get; set; }
        public string Fullname { get; set; }
        public int? Age { get; set; }

        public virtual ICollection<Subscription> Subscriptions { get; set; }
    }
}
```

ConsoleUI.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Lab2.Models;
using Lab2.Controllers;
using Microsoft.Extensions.Options;
using System.Diagnostics;

namespace Lab2.Views
{
    public class ConsoleUI
    {
        private readonly CourseDBContext _courseDBContext;
        private readonly DbConnectionInfo _dbConnectionInfo;
        private readonly CourseController _courseController;
        private readonly LectureController _lectureController;
        private readonly CourseUI _courseUI;

        public ConsoleUI(string databaseConnection)
        {
            _dbConnectionInfo = new DbConnectionInfo();
            _dbConnectionInfo.CourseDBContext = databaseConnection;
            _courseDBContext = new CourseDBContext(_dbConnectionInfo);
            _courseController = new CourseController(_courseDBContext);
            _lectureController = new LectureController(_courseDBContext);
            _courseUI = new CourseUI(_courseController, _lectureController, _courseDBContext);
        }

        public void Run()
        {
            while (true)
            {
                Console.Clear();
                Console.WriteLine("Choose working option:\r\n[1] filter data in the database\r\n[2] make changes
in the database\r\n\r\nwrite \"exit\" to exit the program");

                try
                {
                    int option = GetIntInput();

                    switch (option)
                    {
                        case 1:
                            RunFilter();
                            break;
                        case 2:
                            RunMakingChanges();
                            break;
                        default:
                            break;
                    }
                }
                catch (ArgumentException)
```

```
                {
                    continue;
                }
                catch (Exception)
                {
                    break;
                }
            }
        }

        internal static int GetIntInput()
        {
            string entered = Console.ReadLine();

            if (entered.Equals("exit"))
            {
                Console.WriteLine("Ending session...");
                Environment.Exit(0);
            }
            else if (entered.Equals("back"))
            {
                throw new Exception();
            }

            int option;
            bool isInt = int.TryParse(entered, out option);

            if (!isInt)
            {
                Console.Clear();
                throw new ArgumentException(entered);
            }

            return option;
        }

        private void RunMakingChanges()
        {
            while (true)
            {
                Console.Clear();
                   Console.WriteLine("Choose table:\r\n[1] courses\r\n\r\n write \"back\" to return to the main
menu");

                try
                {
                    int option = GetIntInput();

                    switch (option)
                    {
                        case 1:
                            _courseUI.OperationsWithCourses();
                            break;
                        default:
                            break;
                    }
                }
                catch (ArgumentException)
                {
                    continue;
                }
                catch (Exception)
                {
                    break;
                }
            }
        }

        private void RunFilter()
        {
            Console.Clear();

            Console.WriteLine("Filtering the course table by created day, cost and name");

            DateTime fromTime, toTime;

            try
            {
                (fromTime, toTime) = GetTimeSpanCreationCourses();
```

```
            }
            catch
            {
                return;
            }

            Stopwatch stopwatch = Stopwatch.StartNew();

            stopwatch.Start();
                   _courseDBContext.Courses.Where(course => course.Name.Contains("a") && course.Cost < 50 &&
course.CreatedAt >= fromTime && course.CreatedAt <= toTime);
            stopwatch.Stop();

            Console.WriteLine($"Elapsed Time is {stopwatch.ElapsedMilliseconds} ms");
            Console.WriteLine("Press any key to continue...");
            if (Console.ReadLine() != "")
                return;
        }

        private (DateTime fromTime, DateTime toTime) GetTimeSpanCreationCourses()
        {
            while (true)
            {
                Console.Clear();

                 Console.WriteLine("Write creation date in such format: 31/12/2020-28/02/2022\r\n write \"back\"
to return back");

                string[] entered = Console.ReadLine().Split('-');

                if (entered[0].Equals("back"))
                    throw new Exception();

                DateTime fromTime, toTime;
                try
                {
                    fromTime = DateTime.Parse(entered[0]);
                    toTime = DateTime.Parse(entered[1]);
                }
                catch
                {
                    continue;
                }

                return (fromTime, toTime);
            }
        }
    }
}
```

CourseUI.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Threading;
using Lab2.Models;
using Lab2.Controllers;

namespace Lab2.Views
{
    internal class CourseUI
    {
        private readonly CourseController _courseController;
        private readonly LectureController _lectureController;
        private readonly CourseDBContext _dbContext;

         public CourseUI(CourseController courseController, LectureController lectureController, CourseDBContext
courseDBContext)
        {
            _courseController = courseController;
            _lectureController = lectureController;
```

```
            _dbContext = courseDBContext;
        }

        internal void OperationsWithCourses()
        {
            while (true)
            {
                Console.Clear();
                   Console.WriteLine("Choose operation [course]:\r\n[1] create\r\n[2] edit\r\n[3] delete\r\n\r\n
write \"back\" to step back");

                try
                {
                    int option = ConsoleUI.GetIntInput();

                    switch (option)
                    {
                        case 1:
                            CreateCourse();
                            break;
                        case 2:
                            EditCourse();
                            break;
                        case 3:
                            DeleteCourse();
                            break;
                        default:
                            break;
                    }
                }
                catch (ArgumentException)
                {
                    continue;
                }
                catch (Exception)
                {
                    break;
                }
            }
        }

        private void CreateCourse()
        {
            while (true)
            {
                Console.Clear();
                Console.WriteLine("Write name of course\r\n\r\n write \"back\" to step back");

                string name = Console.ReadLine();
                if (name.Equals("back"))
                    break;

                int intCost = -1;
                intCost = GetCost();

                if (intCost == -1)
                    break;

                Course course = new Course();
                course.Name = name;
                course.Cost = intCost;
                course.CreatedAt = DateTime.Now;

                if (_courseController.Create(course) == 1)
                {
                    Console.WriteLine("Course was added successfully. Do you want to add lectures? [y/n]");
                    string answer = Console.ReadLine();

                    if(answer.Equals("y"))
                    {
                        OpenCreateLectureUI(course.CourseId);
                    }
                    return;
                }

                return;
            }
        }
```

```
        private void OpenCreateLectureUI(int courseId)
        {
            LectureUI lectureUI = new LectureUI(_lectureController);
            lectureUI.CurrCourseId = courseId;
            lectureUI.CreateLecture();
            lectureUI.OperationsWithLectures();
        }

        private void OpenEditLectureUI(int courseId)
        {
            LectureUI lectureUI = new LectureUI(_lectureController);
            lectureUI.CurrCourseId = courseId;
                                            lectureUI.lectures    =    _dbContext.Lectures.Where(lecture    =>
lecture.CourseId.Equals(courseId)).ToList<Lecture>();
            lectureUI.EditLecture();
            lectureUI.OperationsWithLectures();
        }

        private void ShowLectures(List<Lecture> lectures)
        {
            foreach (Lecture l in lectures)
            {
                Console.WriteLine(l);
            }
        }

        private int GetCost()
        {
            int cost = 0;

            while (true)
            {
                Console.Clear();
                Console.WriteLine("Write cost of course\r\n\r\n write \"back\" to step back");

                try
                {
                    cost = ConsoleUI.GetIntInput();
                }
                catch (ArgumentException)
                {
                    continue;
                }
                catch (Exception)
                {
                    return -1;
                }

                if (cost < 0)
                    continue;

                break;
            }

            return cost;
        }

        private void EditCourse()
        {
            while (true)
            {
                int courseId = -1;
                try
                {
                    courseId = GetCourseId();
                }
                catch
                {
                    return;
                }

                if (courseId == -1)
                    continue;

                Course course = _courseController.GetCourse(courseId);

                if (course == null)
                {
                    Console.WriteLine("Course was not found");
```

```
                    Thread.Sleep(1000);
                    continue;
                }

                  Console.WriteLine("Course by Id #{0}, name: {1}, cost: {2}, created at: {3}", course.CourseId,
course.Name, course.Cost, course.CreatedAt);
                Console.WriteLine("\r\nWrite new name and new cost in such format\r\nnewName;cost");
                string[] enteredData = Console.ReadLine().Split(';');

                if (enteredData.Length != 2)
                    continue;
                try
                {
                    course.Name = enteredData[0];
                    course.Cost = int.Parse(enteredData[1]);
                    if (_courseController.Update(course) == 1)
                    {
                              Console.WriteLine("Course was edited successfully. Do you want to edit lectures?
[y/n]");
                        string answer = Console.ReadLine();

                        if (answer.Equals("y"))
                        {
                            OpenEditLectureUI(course.CourseId);
                        }
                        return;
                    }
                }
                catch
                {
                    continue;
                }

                return;
            }
        }

        private void DeleteLectures(List<Lecture> lectures)
        {
            foreach (Lecture l in lectures)
            {
                _lectureController.Delete(l.LectureId);
            }
        }

        private void DeleteCourse()
        {
            while (true)
            {
                int courseId = -1;
                try
                {
                    courseId = GetCourseId();
                }
                catch
                {
                    return;
                }

                if (courseId == -1)
                    continue;
                try
                {
                                            List<Lecture>  lectures  =  _dbContext.Lectures.Where(lecture  =>
lecture.CourseId.Equals(courseId)).ToList<Lecture>();

                    ShowLectures(lectures);

                    Console.WriteLine("Lectures will be deleted. Do you want to delete course? [y/n]");
                    string answer = Console.ReadLine();

                    if (answer.Equals("y"))
                    {
                        DeleteLectures(lectures);
                    }

                    if (_courseController.Delete(courseId) == 1)
                    {
```

```
                        Console.WriteLine("Operation is successfull. Press any key to continue...");
                        if (Console.ReadLine() != "")
                            return;
                    }
                    else
                    {
                        Console.WriteLine("Course was not found");
                        Thread.Sleep(1000);
                        continue;
                    }
                }
                catch
                {
                    Console.WriteLine("Some error occur");
                    Thread.Sleep(1000);
                    return;
                }

                return;
            }
        }

        private int GetCourseId()
        {
            Console.Clear();
            Console.WriteLine("Enter course id\r\n\r\n write \"back\" to step back");

            int courseId = -1;

            try
            {
                courseId = ConsoleUI.GetIntInput();
            }
            catch (ArgumentException)
            {
                return courseId;
            }

            return courseId;
        }
    }
}
```

LectureUI.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using Lab2.Models;
using Lab2.Controllers;

namespace Lab2.Views
{
    internal class LectureUI
    {
        private LectureController _controller;
        public int CurrCourseId { get; set; }
        public List<Lecture> lectures { get; set; }

        public LectureUI(LectureController controller) => _controller = controller;

        internal void OperationsWithLectures()
        {
            while (true)
            {
                Console.Clear();
                Console.WriteLine($"Choose operation [lecture of course [#{CurrCourseId}]]:\r\n[1] create\r\n[2] edit\r\n[3] delete\r\n\r\n write \"back\" to step back");

                try
                {
                    int option = ConsoleUI.GetIntInput();
```

```
                switch (option)
                {
                    case 1:
                        CreateLecture();
                        break;
                    case 2:
                        EditLecture();
                        break;
                    case 3:
                        DeleteLecture();
                        break;
                    default:
                        break;
                }
            }
            catch (ArgumentException)
            {
                continue;
            }
            catch (Exception)
            {
                return;
            }
        }
    }

    internal void CreateLecture()
    {
        while (true)
        {
            Console.Clear();
            Console.WriteLine("Write name of lecture\r\n\r\n write \"back\" to step back");

            string name = Console.ReadLine();
            if (name.Equals("back"))
                break;

            Lecture lecture = new Lecture();
            lecture.Name = name;
            lecture.CourseId = CurrCourseId;

            if (_controller.Create(lecture) == 1)
            {
                Console.WriteLine(lecture.ToString()+"\r\n"+"Press any key to continue...");
                if (Console.ReadKey().Equals(new ConsoleKeyInfo()))
                    return;
            }

            return;
        }
    }

    private void PrintLectures()
    {
        foreach(var lecture in lectures)
        {
            Console.WriteLine(lecture);
        }
    }

    internal void EditLecture()
    {
        while (true)
        {
            Console.Clear();

            try
            {
                PrintLectures();
            }
            catch
            {
                Console.WriteLine("Error printing lectures");
                Thread.Sleep(4000);
            }

            int lectureId = -1;
            try
            {
```

```
                    lectureId = GetLectureId();
                }
                catch
                {
                    return;
                }

                if (lectureId == -1)
                    continue;

                Lecture lecture = _controller.GetLecture(lectureId);

                if (lecture == null)
                {
                    Console.WriteLine("Lecture was not found");
                    Thread.Sleep(1000);
                    continue;
                }

                        Console.WriteLine("Lecture by Id #{0}, name: {1}, courseId: {2}", lecture.LectureId,
lecture.Name, lecture.CourseId);
                Console.WriteLine("\r\nWrite new name");
                string enteredData = Console.ReadLine();

                try
                {
                    lecture.Name = enteredData;

                    if (_controller.Update(lecture) == 1)
                    {
                        Console.WriteLine("Operation is successfull. Press any key to continue...");
                        if (Console.ReadKey().Equals(new ConsoleKeyInfo()))
                            return;
                    }
                }
                catch
                {
                    continue;
                }

                return;
            }
        }

        private void DeleteLecture()
        {
            while (true)
            {
                int courseId = -1;
                try
                {
                    courseId = GetLectureId();
                }
                catch
                {
                    return;
                }

                if (courseId == -1)
                    continue;
                try
                {
                    if (_controller.Delete(courseId) == 1)
                    {
                        Console.WriteLine("Operation is successfull. Press any key to continue...");
                        if (Console.ReadKey().Equals(new ConsoleKeyInfo()))
                            return;
                    }
                    else
                    {
                        Console.WriteLine("Course was not found");
                        Thread.Sleep(1000);
                        continue;
                    }
                }
                catch
                {
                    Console.WriteLine("Some error occur");
                    Thread.Sleep(1000);
```

```
                    return;
                }

                return;
            }
        }

        private int GetLectureId()
        {

            Console.WriteLine("Enter lecture id\r\n\r\n write \"back\" to step back");

            int lectureId  = -1;

            try
            {
                lectureId = ConsoleUI.GetIntInput();
            }
            catch (ArgumentException)
            {
                return lectureId;
            }

            return lectureId;
        }
    }
}
```

Висновки

Виконавши дану лабораторну роботу, ми ознайомилися з можливостями PostgreSQL. Здобули вміння програмування прикладних додатків баз даних PostgreSQL.