МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РАДІОЕЛЕКТРОНІКИ

Кафедра Штучного Інтелекту

Звіт

про виконання лабораторної роботи №2

**«Використання технології Entity Framework Core для**

**організації доступу до баз даних з .NET-застосунків.»**

з дисципліни «Програмування під .NЕТ Соrе»

Виконав:

ст. гр. ІТШІ-20-2

Науменко А.С.

Прийняв:

Бібічков І.Є.

Харків – 2023

**1. НАЗВА РОБОТИ:**

Використання технології Entity Framework Core для організації доступу до баз даних з .NET-застосунків.

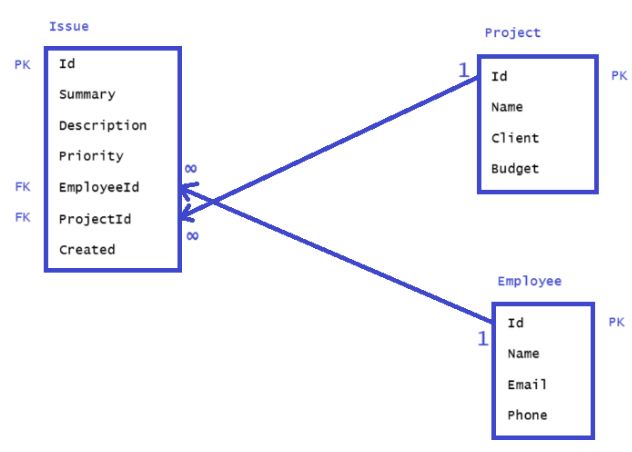
**2. МЕТА РОБОТИ:**

Вивчення особливостей використання технології Entity Framework Core для організації доступу до баз даних з .NET-застосунків.

**ТЕМА:**

Task Management

**3. СХЕМА БАЗИ ДАНИХ:**



**4. ІНТЕРФЕЙСИ КЛАСІВ, ЗОКРЕМА, КЛАСІВ СУТНОСТЕЙ ТА КОНТЕКСТУ ДАНИХ:**

**Project.cs**

public class Project  
{  
 [Key]  
 public Guid Id { get; set; }  
 public string Name { get; set; }  
 public string Client { get; set; }  
 public double Budget { get; set; }  
}

**Issue.cs**

public class Issue  
{  
 [Key]  
 public Guid Id { get; set; }  
 public string Summary { get; set; }  
 public string Description { get; set; }  
 public string Priority { get; set; }  
   
 public Guid? EmployeeId { get; set; }  
 public Employee Employee { get; set; }  
   
 public Guid? ProjectId { get; set; }  
 public Project Project { get; set; }  
 public DateTime Created { get; set; }  
}

**Employee.cs**

public class Employee  
{  
 [Key]  
 public Guid Id { get; set; }  
 public string Name { get; set; }  
 public string Email { get; set; }  
 public string Phone { get; set; }  
}

**TaskDbContext.cs**

public class TaskDbContext : DbContext  
{  
 public DbSet<Employee> Employees { get; set; }  
 public DbSet<Project> Projects { get; set; }  
 public DbSet<Issue> Issues { get; set; }  
 public TaskDbContext(DbContextOptions options) : base(options)  
 protected override void OnModelCreating(ModelBuilder modelBuilder)  
}

**5. ВИХІДНІ ТЕКСТИ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ:**

**Project.cs**

using System.ComponentModel.DataAnnotations;  
  
namespace TaskManagement.Models;  
  
public class Project  
{  
 [Key]  
 public Guid Id { get; set; }  
 public string Name { get; set; }  
 public string Client { get; set; }  
 public double Budget { get; set; }  
}

**Issue.cs**

using System.ComponentModel.DataAnnotations;  
  
namespace TaskManagement.Models;  
  
public class Issue  
{  
 [Key]  
 public Guid Id { get; set; }  
 public string Summary { get; set; }  
 public string Description { get; set; }  
 public string Priority { get; set; }  
   
 public Guid? EmployeeId { get; set; }  
 public Employee Employee { get; set; }  
   
 public Guid? ProjectId { get; set; }  
 public Project Project { get; set; }  
 public DateTime Created { get; set; }  
}

**Employee.cs**

using System.ComponentModel.DataAnnotations;  
  
namespace TaskManagement.Models;  
  
public class Employee  
{  
 [Key]  
 public Guid Id { get; set; }  
 public string Name { get; set; }  
 public string Email { get; set; }  
 public string Phone { get; set; }  
}

**TaskDbContext.cs**

using Microsoft.EntityFrameworkCore;  
using TaskManagement.Models;  
  
namespace TaskManagement.Db;  
  
public class TaskDbContext : DbContext  
{  
 public DbSet<Employee> Employees { get; set; }  
 public DbSet<Project> Projects { get; set; }  
 public DbSet<Issue> Issues { get; set; }  
  
 public TaskDbContext(DbContextOptions options) : base(options)  
 {  
 }  
  
 protected override void OnModelCreating(ModelBuilder modelBuilder)  
 {  
 base.OnModelCreating(modelBuilder);  
  
 modelBuilder.Entity<Issue>(entry =>  
 {  
 entry.ToTable("task");  
   
 entry.HasOne(t => t.Employee)  
 .WithMany()  
 .HasForeignKey(t => t.EmployeeId)  
 .OnDelete(DeleteBehavior.**Cascade**);  
   
 entry.HasOne(t => t.Project)  
 .WithMany()  
 .HasForeignKey(t => t.ProjectId)  
 .OnDelete(DeleteBehavior.**Cascade**);  
 });  
  
 modelBuilder.Entity<Project>(entry => { entry.ToTable("project"); });  
 modelBuilder.Entity<Employee>(entry => { entry.ToTable("employee"); });  
  
 }  
}

**Form1.cs**

using Microsoft.EntityFrameworkCore;  
using TaskManagement.Db;  
  
namespace TaskManagement  
{  
 public partial class Form1 : Form  
 {  
 private readonly TaskDbContext \_taskDbContext;  
 public Form1(TaskDbContext taskDbContext)  
 {  
 \_taskDbContext = taskDbContext;  
 InitializeComponent();  
  
 ShowIssues();  
 ShowEmployees();  
 ShowProjects();  
 }  
  
 public void ShowIssues()  
 {  
 tasksListView.Items.Clear();  
 var issues = \_taskDbContext.Issues.AsNoTracking()  
 .Include(i => i.Employee)  
 .Include(i => i.Project)  
 .OrderByDescending(i => i.Created);  
  
 foreach (var issue in issues)  
 {  
 ListViewItem item = new(issue.Id.ToString());  
 item.SubItems.Add(issue.Summary);  
 item.SubItems.Add(issue.Description);  
 item.SubItems.Add(issue.Priority);  
 item.SubItems.Add(issue.Employee?.Name);  
 item.SubItems.Add(issue.Project?.Name);  
 tasksListView.Items.Add(item);  
 }  
 }  
  
 public void ShowProjects()  
 {  
 projectsListView.Items.Clear();  
 var projects = \_taskDbContext.Projects.AsNoTracking()  
 .OrderByDescending(p => p.Name);  
  
 foreach (var project in projects)  
 {  
 ListViewItem item = new(project.Id.ToString());  
 item.SubItems.Add(project.Name);  
 item.SubItems.Add(project.Client);  
 item.SubItems.Add(project.Budget.ToString());  
 projectsListView.Items.Add(item);  
 }  
 }  
  
 public void ShowEmployees()  
 {  
 employeesListView.Items.Clear();  
 var employees = \_taskDbContext.Employees.AsNoTracking()  
 .OrderByDescending(p => p.Name);  
  
 foreach (var employee in employees)  
 {  
 ListViewItem item = new(employee.Id.ToString());  
 item.SubItems.Add(employee.Name);  
 item.SubItems.Add(employee.Email);  
 item.SubItems.Add(employee.Phone);  
 employeesListView.Items.Add(item);  
 }  
 }  
  
 private void addBtn\_Click(object sender, EventArgs e)  
 {  
 IssueForm issueForm = new(Guid.Empty, \_taskDbContext);  
 issueForm.Closed += (send, evt) => ShowIssues();  
 issueForm.Show();  
 }  
  
 private void delBtn\_Click(object sender, EventArgs e)  
 {  
 if (tasksListView.SelectedItems.Count != 0)  
 {  
 var id =

new Guid(tasksListView.SelectedItems[0].SubItems[0].Text);  
 \_taskDbContext.Remove(\_taskDbContext.Issues.Find(id));  
 \_taskDbContext.SaveChanges();  
 ShowIssues();  
 }  
 }  
  
 private void tasksListView\_ItemActivate(object sender, EventArgs e)  
 {  
 var id =

new Guid((sender as ListView).SelectedItems[0].SubItems[0].Text);  
 IssueForm issueForm = new(id, \_taskDbContext);  
 issueForm.Closed += (send, evt) => ShowIssues();  
 issueForm.Show();  
 }  
 }  
}

**IssueForm.cs**

using Microsoft.EntityFrameworkCore;  
using TaskManagement.Db;  
using TaskManagement.Models;  
  
namespace TaskManagement  
{  
 public partial class IssueForm : Form  
 {  
 private readonly TaskDbContext \_taskDbContext;  
 private Issue \_issue = new();  
   
   
 public IssueForm(Guid id, TaskDbContext taskDbContext)  
 {  
 InitializeComponent();  
  
 \_taskDbContext = taskDbContext;  
   
 var employees =

\_taskDbContext.Employees.OrderByDescending(x => x.Name).ToList();  
 var projects =

\_taskDbContext.Projects.OrderByDescending(x => x.Name).ToList();  
   
 employeesCb.DataSource = employees;  
 projectsCb.DataSource = projects;  
 employeesCb.ValueMember = "Id";  
 projectsCb.ValueMember = "Id";  
 employeesCb.DisplayMember = "Name";  
 projectsCb.DisplayMember = "Name";  
 employeesCb.SelectedIndex = -1;  
 projectsCb.SelectedIndex = -1;  
  
 if (id == Guid.Empty)  
 {  
 return;  
 }  
   
 \_issue = \_taskDbContext.Issues.AsNoTracking()  
 .Include(i => i.Employee)  
 .Include(i => i.Project)  
 .FirstOrDefault(x => x.Id == id);  
   
 employeesCb.SelectedIndex =

employees.FindIndex(x => x.Id == \_issue.EmployeeId);  
 projectsCb.SelectedIndex =

projects.FindIndex(x => x.Id == \_issue.ProjectId);  
   
 summaryTb.Text = \_issue!.Summary;  
 descriptionTb.Text = \_issue.Description;  
 priorityTb.Text = \_issue.Priority;  
 }  
  
 private void saveBtn\_Click(object sender, EventArgs e)  
 {  
 \_issue.Summary = summaryTb.Text;  
 \_issue.Description = descriptionTb.Text;  
 \_issue.Priority = priorityTb.Text;  
   
 var employees =

\_taskDbContext.Employees.OrderByDescending(x => x.Name).ToList();  
 var projects =

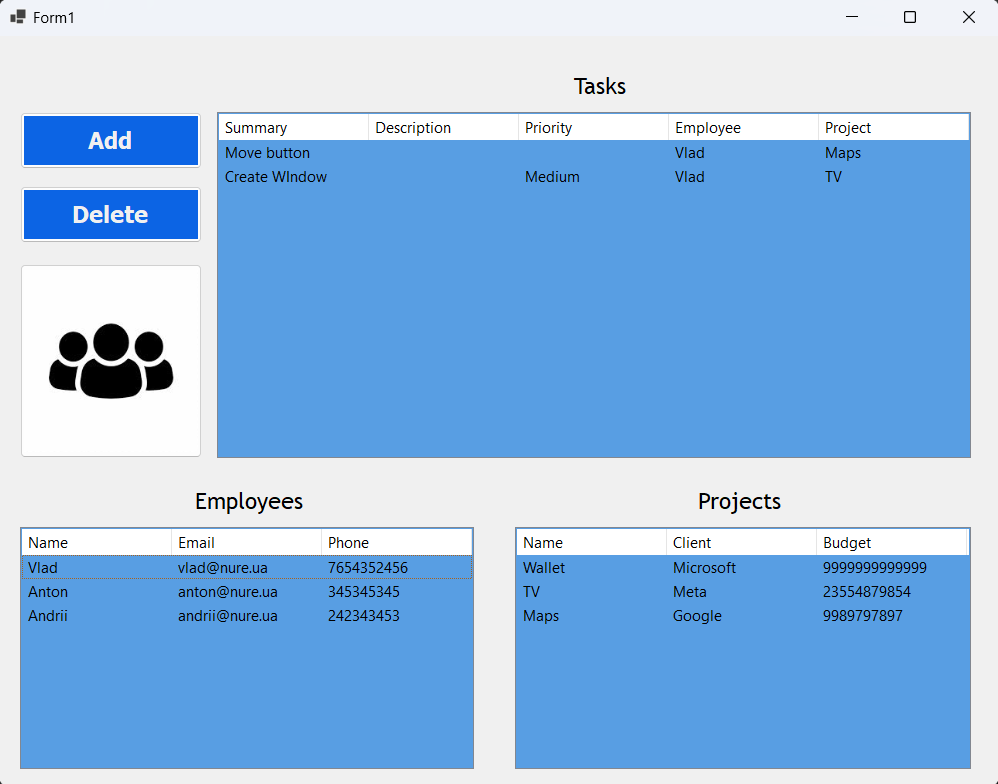
\_taskDbContext.Projects.OrderByDescending(x => x.Name).ToList();  
  
 if (employees.Count != 0 && employeesCb.SelectedIndex != -1)  
 {  
 \_issue.Employee = employees[employeesCb.SelectedIndex];  
 }  
   
 if (projects.Count != 0 && projectsCb.SelectedIndex != -1)  
 {  
 \_issue.Project = projects[projectsCb.SelectedIndex];  
 }

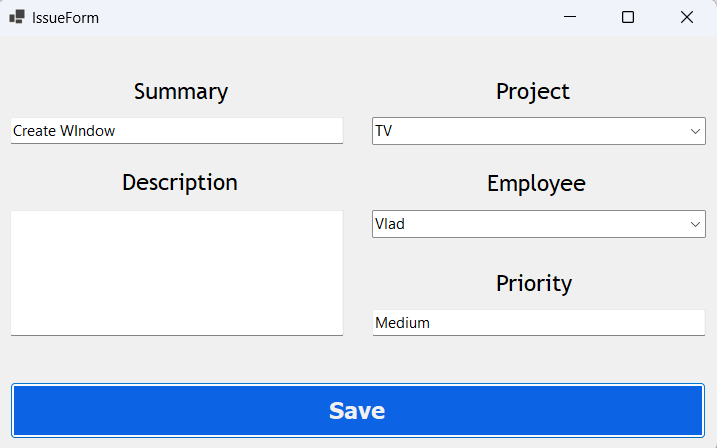
if (\_issue.Id == Guid.Empty)  
 {  
 \_issue.Id = Guid.NewGuid();  
 \_issue.Created = DateTime.Now;  
 \_taskDbContext.Add(\_issue);  
 }  
 else  
 {  
 \_taskDbContext.Update(\_issue);  
 }  
 \_taskDbContext.SaveChanges();  
 Close();  
 }  
 }  
}

**Program.cs**

using Microsoft.EntityFrameworkCore;  
using TaskManagement.Db;  
  
namespace TaskManagement  
{  
 internal static class Program  
 {  
 [STAThread]  
 static void Main()  
 {  
 ApplicationConfiguration.Initialize();  
   
 var optionsBuilder = new DbContextOptionsBuilder<TaskDbContext>();  
 string connectionString = "Server=localhost;Database=task\_management;Uid=root;Pwd=mysecretpassword;";  
 ServerVersion serverVersion = ServerVersion.AutoDetect(connectionString);  
 optionsBuilder.UseMySql(connectionString, serverVersion);  
 var taskDbContext = new TaskDbContext(optionsBuilder.Options);  
   
 Application.Run(new Form1(taskDbContext));  
 }  
 }  
}

**6. СКРІНИ ВІЗУАЛЬНОГО ІНТЕРФЕЙСУ:**

****

****

**7. ВИСНОВКИ:**

Під час виконання даної роботи я вивчив особливості використання технології Entity Framework Core для організації доступу до баз даних з .NET-застосунків. Було розроблено WinForms застосунок на тему «Task Management». Застосунок має такі вікна: для відображення даних за допомогою таблиць та для редагування та створення тасків.