Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №2

за 6 семестр

По дисциплине: «ПИС»

Выполнил:

Студент 3 курса

Группы ПО-4(1)

Иваненко И.Л.

Проверил:

Михняев А.Л.

2022

**Цель:** Познакомиться с предметной областью, реализовать минимальный функционал без детального проектирования согласно паттерну «сценарий транзакции»

**Предметная область:** учет успеваемостис студентов

**Структура базы данных**:

1. Таблица студентов

2. Таблица факультетов

3. Таблица групп

using DAL.Data;  
using DAL.Interfaces;  
using DAL.Models;  
  
namespace DAL.Implementations;  
  
public class StudentRepository : IStudentRepository  
{  
 private readonly ProgressDbContext \_dbContext;  
  
 public StudentRepository(ProgressDbContext dbContext)  
 {  
 \_dbContext = dbContext;  
 }  
  
 public void AddStudent(Student student)  
 {  
 \_dbContext.Add(student);  
 \_dbContext.SaveChanges();  
 }  
  
 public Student FindById(string id)  
 {  
 return \_dbContext.Students.FirstOrDefault(s => s.Id == id)!;  
 }  
  
 public IEnumerable<Student> GetAllStudents()  
 {  
 return \_dbContext.Students;  
 }  
  
 public void AddSubject(string id, Subject subject)  
 {  
 Student student = FindById(id);  
 student.Subjects!.Add(subject);  
 subject.Students!.Add(student);  
 \_dbContext.SaveChanges();  
 }  
  
 public void Dispose()  
 {  
 \_dbContext.Dispose();  
 }  
}

using BL.Interfaces;  
using DAL.Models;  
using Lab4.Models;  
using Microsoft.AspNetCore.Mvc;  
  
namespace Lab4.Controllers;  
  
public class StudentController : Controller  
{  
 private readonly ILogger<StudentController> \_logger;  
 private readonly IStudentManager \_studentManager;  
 private readonly IGroupManager \_groupManager;  
   
 public StudentController(  
 ILogger<StudentController> logger,   
 IStudentManager studentManager,   
 IGroupManager groupManager)  
 {  
 \_logger = logger;  
 \_studentManager = studentManager;  
 \_groupManager = groupManager;  
 }  
  
 public IActionResult Index()  
 **{** List<Group> groups = \_groupManager.GetAllGroups().ToList();  
  
 var viewModel = new CreateStudentViewModel  
 {  
 Groups = groups  
 };  
 return View(viewModel);  
 **}** [HttpPost]  
 public IActionResult Index(CreateStudentViewModel createStudentViewModel)  
 {  
 createStudentViewModel.Groups = \_groupManager.GetAllGroups().ToList();  
 if (!ModelState.IsValid)  
 {  
 return View(createStudentViewModel);  
 }  
  
 \_studentManager.CreateStudent(createStudentViewModel.Name!, createStudentViewModel.SelectedGroupId!);  
 return View(createStudentViewModel);  
 }  
}

using DAL.Data;  
using DAL.Interfaces;  
using DAL.Models;  
  
namespace DAL.Implementations;  
  
public class GroupRepository : IGroupRepository  
{  
 private readonly ProgressDbContext \_dbContext;  
  
 public GroupRepository(ProgressDbContext dbContext)  
 {  
 \_dbContext = dbContext;  
 }  
  
 public void AddGroup(Group group)  
 {  
 \_dbContext.Groups!.Add(group);  
 \_dbContext.SaveChanges();  
 }  
  
 public Group FindById(string id)  
 {  
 return \_dbContext.Groups!.FirstOrDefault(group => group.Id == id)   
 ?? throw new ArgumentException($"Cannot find group with specified Id: {id}");  
 }  
  
 public IEnumerable<Group> FindByName(string name)  
 {  
 return \_dbContext.Groups!.Where(group => group.Name == name);  
 }  
  
 public IEnumerable<Group> GetAllGroups()  
 {  
 return \_dbContext.Groups!;  
 }  
  
 public bool Exists(string id)  
 {  
 return \_dbContext.Groups!.Any(group => group.Id == id);  
 }  
   
 public void Dispose()  
 {  
 \_dbContext.Dispose();  
 }  
}

using BL.Interfaces;  
using DAL.Models;  
using Lab4.Models;  
using Microsoft.AspNetCore.Mvc;  
  
namespace Lab4.Controllers;  
  
public class GroupController : Controller  
{  
 private readonly ILogger<GroupController> \_logger;  
 private readonly IFacultyManager \_facultyManager;  
 private readonly IGroupManager \_groupManager;  
  
 public GroupController(  
 ILogger<GroupController> logger,   
 IFacultyManager facultyManager,   
 IGroupManager groupManager)  
 {  
 \_logger = logger;  
 \_facultyManager = facultyManager;  
 \_groupManager = groupManager;  
 }  
  
 public IActionResult Index()  
 {  
 List<Faculty> allFaculties = \_facultyManager.GetAllFaculties().ToList();  
   
 var createGroupViewModel = new CreateGroupViewModel  
 {  
 Faculties = allFaculties  
 };  
   
 return View(createGroupViewModel);  
 }  
  
 [HttpPost]  
 public IActionResult Index(CreateGroupViewModel groupViewModel)  
 {  
 groupViewModel.Faculties = \_facultyManager.GetAllFaculties().ToList();  
 if (!ModelState.IsValid)  
 {  
 return View(groupViewModel);  
 }  
  
 \_groupManager.CreateGroup(groupViewModel.Name!, groupViewModel.SelectedFacultyId!);  
   
 return View(groupViewModel);  
 }  
}

using DAL.Data;  
using DAL.Interfaces;  
using DAL.Models;  
  
namespace DAL.Implementations;  
  
public class FacultyRepository : IFacultyRepository  
{  
 private readonly ProgressDbContext \_dbContext;  
   
 public FacultyRepository(ProgressDbContext dbContext)  
 {  
 \_dbContext = dbContext;  
 }  
  
 public void AddFaculty(Faculty faculty)  
 {  
 \_dbContext.Faculties!.Add(faculty);  
 \_dbContext.SaveChanges();  
 }  
  
 public Faculty FindById(string id)  
 {  
 return \_dbContext.Faculties!.FirstOrDefault(faculty => faculty.Id == id)   
 ?? throw new ArgumentException($"Cannot find faculty with specified Id: {id}");  
 }  
  
 public IEnumerable<Faculty> FindByName(string name)  
 {  
 return \_dbContext.Faculties!.Where(faculty => faculty.Name == name);  
 }  
  
 public IEnumerable<Faculty> GetAllFaculties()  
 {  
 return \_dbContext.Faculties!;  
 }  
  
 public bool Exists(string id)  
 {  
 return \_dbContext.Faculties!.Any(faculty => faculty.Id == id);  
 }  
  
 public void Dispose()  
 {  
 \_dbContext.Dispose();  
 }  
}

using BL.Interfaces;  
using Lab4.Models;  
using Microsoft.AspNetCore.Mvc;  
  
namespace Lab4.Controllers;  
  
public class FacultyController : Controller  
{  
 private readonly ILogger<FacultyController> \_logger;  
 private readonly IFacultyManager \_facultyManager;  
   
 public FacultyController(ILogger<FacultyController> logger,   
 IFacultyManager facultyManager)  
 {  
 \_logger = logger;  
 \_facultyManager = facultyManager;  
 }  
  
 public IActionResult Index()  
 {  
 return View();  
 }  
   
 [HttpPost]  
 public IActionResult Index(CreateFacultyViewModel facultyViewModel)  
 {  
 if (!ModelState.IsValid)  
 {  
 return View(facultyViewModel);  
 }  
  
 \_facultyManager.CreateFaculty(facultyViewModel.Name!);  
 return View(null);  
 }  
}

**Вывод:** познакомился с предметной областью, реализовал минимальный функционал

без детального проектирования согласно паттерну «сценарий транзакции».