Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №3

По дисциплине: «Современные платформы программирования»

Выполнил:

Студент 3 курса

Группы ПО-4

Иваненко И. Л.

Проверил:

Монтик Н. С.

Брест 2022

Вариант 10

Цель работы: приобрести практические навыки разработки баз данных и начальной интеграции БД с кодом C# с помощью Microsoft Entity Framework Core

# Общее задание

Реализовать базу данных из не менее 5 таблиц на заданную тематику. При реализации продумать типизацию полей и внешние ключи в таблицах. Визуализировать разработанную БД с помощью схемы, на которой отображены все таблицы и связи между ними. На языке C# с использованием Microsoft Entity Framework Core реализовать подключение к БД и выполнить основные типы запросов, продемонстрировать результаты преподавателю и включить тексты составленных запросов в отчет. Основные типы запросов:

1. На выборку/на выборку с упорядочиванием.

2. На добавление.

3. На удаление.

4. На модификацию.

## Задание по варианту:

10) База данных «Учет успеваемости»

# Дизайн базы данных

Graphical user interface

Description automatically generated

# Код программы:

## Faculty.cs

public class Faculty  
{  
 public string Id { get; set; }  
 public string Name { get; set; }  
 public List<Group> Groups { get; set; } = new();  
  
 public override string ToString()  
 {  
 return $"FACULTY: Id: {Id}; Name: {Name}";  
 }  
}

## Group.cs

public class Group  
{  
 public string Id { get; set; }  
 public string Name { get; set; }  
  
 public string FacultyId { get; set; }  
 public Faculty Faculty { get; set; }  
  
 public List<Student> Students { get; set; } = new();  
  
 public override string ToString()  
 {  
 return $"GROUP: Id: {Id}; Name: {Name}; FacultyId: {FacultyId}; Faculty: {Faculty.ToString()}";  
 }  
}

## Mark.cs

public class Mark : IComparable  
{  
 public string Id { get; set; }  
 public double Value { get; set; }  
  
 public string ProgressId { get; set; }  
 public Progress Progress { get; set; }  
   
 public int CompareTo(object obj)  
 {  
 if (obj is Mark mark)  
 {  
 return (int) (mark.Value - Value);  
 }  
  
 throw new ArgumentException();  
 }  
  
 public override string ToString()  
 {  
 return $"MARK: Id: {Id}; Value: {Value};";  
 }  
}

## Progress.cs

public class Progress  
{  
 public string Id { get; set; }  
  
 public string StudentId { get; set; }  
 public Student Student { get; set; }  
  
 public string SubjectId { get; set; }  
 public Subject Subject { get; set; }  
  
 public ICollection<Mark> Marks { get; set; } = new List<Mark>();  
  
 public double AverageResult => Marks.Select(m => m.Value).Average();  
  
 public override string ToString()  
 {  
 return  
 $"PROGRESS: Id: {Id};";  
 }  
}

## Student.cs

public class Student  
{  
 public string Id { get; set; }  
 public string Name { get; set; }  
  
 public string GroupId { get; set; }  
 public Group Group { get; set; }  
  
 public ICollection<Subject> Subjects { get; set; } = new List<Subject>();  
 public List<Progress> Progresses { get; set; } = new();  
  
 public override string ToString()  
 {  
 return $"STUDENT: Id: {Id}; Name: {Name};";  
 }  
}

## Subject.cs

public class Subject  
{  
 public string Id { get; set; }  
 public string Name { get; set; }  
 public ICollection<Student> Students { get; set; } = new List<Student>();  
  
 public override string ToString()  
 {  
 return $"SUBJECT: Id: {Id}; Name: {Name}";  
 }  
}

## StudentsProgressContext.cs

public class StudentsProgressContext : DbContext  
{  
 public DbSet<Faculty> Faculties { get; set; }  
 public DbSet<Group> Groups { get; set; }  
 public DbSet<Progress> Progresses { get; set; }  
 public DbSet<Student> Students { get; set; }  
 public DbSet<Subject> Subjects { get; set; }  
 public DbSet<Mark> Marks { get; set; }  
  
 protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)  
 {  
 optionsBuilder.UseSqlite(@"");  
 }  
  
 protected override void OnModelCreating(ModelBuilder modelBuilder)  
 {  
 modelBuilder.Entity<Progress>()  
 .HasMany(p => p.Marks)  
 .WithOne(m => m.Progress)  
 .OnDelete(DeleteBehavior.**Cascade**);  
 }  
}

## StudentsProgressRepository:

public class StudentsProgressRepository : IDisposable  
{  
 private readonly StudentsProgressContext \_dbContext;  
  
 public StudentsProgressRepository(StudentsProgressContext dbContext)  
 {  
 \_dbContext = dbContext;  
 }  
   
 public List<Progress> GetAllProgresses()  
 {  
 Console.WriteLine("Getting all progresses");  
 return \_dbContext.Progresses.ToList();  
 }  
  
 public List<Mark> GetAllMarks()  
 {  
 Console.WriteLine("Getting all marks");  
 return \_dbContext.Marks.ToList();  
 }  
  
 public List<Mark> GetAllMarksSorted()  
 {  
 Console.WriteLine("Getting sorted marks");  
 List<Mark> marksToSort = \_dbContext.Marks.ToList();  
 marksToSort.Sort();  
 return marksToSort;  
 }  
  
 public void AddFaculty(Faculty faculty)  
 {  
 Console.WriteLine($"Adding {faculty}");  
 \_dbContext.Faculties.Add(faculty);  
 \_dbContext.SaveChanges();  
 }  
  
 public void AddGroup(Group group)  
 {  
 Console.WriteLine($"Adding {group}");  
 \_dbContext.Groups.Add(group);  
 \_dbContext.SaveChanges();  
 }  
  
 public void AddStudent(Student student)  
 {  
 Console.WriteLine($"Adding {student}");  
 \_dbContext.Students.Add(student);  
 \_dbContext.SaveChanges();  
 }  
  
 public void AddSubject(Subject subject)  
 {  
 Console.WriteLine($"Adding subject");  
 \_dbContext.Subjects.Add(subject);  
 \_dbContext.SaveChanges();  
 }  
  
 public void InitStudent(Student student, List<Subject> subjects)  
 {  
 Console.WriteLine($"Initializing {student}");  
 foreach (var subject in subjects)  
 {  
 Console.WriteLine($"Adding {subject}");  
 subject.Students.Add(student);  
 student.Subjects.Add(subject);  
 \_dbContext.Subjects.Add(subject);  
 }  
  
 \_dbContext.Students.Add(student);  
 \_dbContext.SaveChanges();  
 }  
  
 public void AddProgress(Progress progress)  
 {  
 Console.WriteLine($"Adding {progress}");  
 \_dbContext.Progresses.Add(progress);  
 \_dbContext.SaveChanges();  
 }  
  
 public void AddMark(Mark mark)  
 {  
 Console.WriteLine($"Adding {mark}");  
 \_dbContext.Marks.Add(mark);  
 \_dbContext.SaveChanges();  
 }  
  
 public void InitProgress(Progress progress, List<Mark> marks)  
 {  
 Console.WriteLine($"Initializing {progress}");  
 foreach (var mark in marks)  
 {  
 Console.WriteLine($"Adding {mark}");  
 mark.ProgressId = progress.Id;  
 mark.Progress = progress;  
 progress.Marks.Add(mark);  
 \_dbContext.Marks.Add(mark);  
 }  
  
 \_dbContext.Progresses.Add(progress);  
 \_dbContext.SaveChanges();  
 }  
  
 public void DeleteProgressesLessThan(double averageResult)  
{  
 Console.WriteLine($"Deleting progress less than {averageResult}");  
 var progressesToDelete =  
 \_dbContext.Progresses.Where(p => p.Marks.Select(m => m.Value).Average() < averageResult).ToList();  
 foreach (var progress in progressesToDelete)  
 {  
 Console.WriteLine($"Deleting {progress}");  
 }  
 \_dbContext.Progresses.RemoveRange(progressesToDelete);  
 \_dbContext.SaveChanges();  
}

public void ChangeStudentName(string studentId, string newName)  
 {  
 Console.WriteLine($"Updating student with id {studentId}");  
 Student studentToUpdate = \_dbContext.Students.FirstOrDefault(s => s.Id == studentId);  
 Console.WriteLine($"Student to update: {studentToUpdate}");  
 if (studentToUpdate != null)  
 {  
 studentToUpdate.Name = newName;  
 \_dbContext.SaveChanges();  
 }  
 Console.WriteLine($"New name {newName}");  
 }  
  
 public void Dispose()  
 {  
 \_dbContext?.Dispose();  
 }  
}

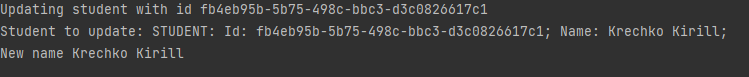
# Результат выполнения:

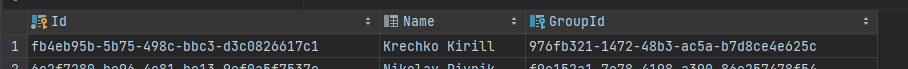
## Добавление данных:

Text

Description automatically generated

## Обновление данных:





## Удаление по условию:

Graphical user interface, text

Description automatically generated

## Выборка отсортированных данных:

Text

Description automatically generated

## Выборка данных:

Text

Description automatically generated

# Вывод

В ходе выполнения данной лабораторной работы были приобретены практические навыки работы с базами данных с помощью таких инструментов как язык программирования C# и Microsoft Entity Framework Core.