2023

مشروع Entity Framework

تقدمة الطلاب

1(هادي الصلاج
2(عبد الرحمن طحان
3(عبد الله طيب

بداية نعرف جدول Department وهو جدول كسر العلاقة بين جدولي الـ **Student** والـ **Subject**

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

namespace ProjectEntity
{
    5 references
    public class Department
    {
        1 reference
        public int Id { get; set; }
        3 references
        public string Name { get; set; }
        0 references
        public ICollection<Subject> Subjects { get; set; } = new List<Subject>();
        0 references
        public ICollection<Student> Students { get; } = new List<Student>();
    }
}
```

ثم قمنا بتعريف جدول **Student** وهو مفتاح فرعي من لجدول **Department** وهو مرتبط مع جدول علامات الطالب

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

namespace ProjectEntity
{
    5 references
    public class Student
    {
        1 reference
        public int Id { get; set; }
        3 references
        public string UserName { get; set; }
        3 references
        public string FirstName { get; set; }
        3 references
        public string LastName { get; set; }
        3 references
        public string Email { get; set; }
        3 references
        public string Phone { get; set; }
        1 reference
        public int RegisterDate { get; set; }
        2 references
        public int DepartmentId { get; set; }
        0 references
        public int StudentMarkId { get; set; }
        0 references
        public ICollection<StudentMark> StudentMarks { get; set; } = new List<StudentMark>();
        0 references
        public Department Department { get; set; } = null!;


    }
}
```

جدول **Student** وهو مفتاح فرعي من لجدول **Department** وهو مرتبط مع جدول المحاضرات وجدول الامتحانات

```
namespace ProjectEntity
{
    public class Subject
    {
        public int Id { get; set; }
        public int DepartmentId { get; set; }
        public string Name { get; set; }
        public int MinimumDegree { get; set; }
        public int Term { get; set; }
        public int Year { get; set; }
        public ICollection<Exam> Exams { get; } = new List<Exam>();
        0 references
        public ICollection<SubjectLecture> SubjectLectures { get; } = new List<SubjectLecture>();
        0 references
        public Department Department { get; set; } = null!;
    }
}
```

جدول **Student Mark** وهو جدول يشير لعلامات الطلاب وهو مفتاح فرعي لجدول **Student** وجدول **Exam**

```
namespace ProjectEntity
{
    6 references
    public class StudentMark
    {
        1 reference
        public int Id { get; set; }
        2 references
        public int ExamId { get; set; }
        3 references
        public int Mark { get; set; }

        2 references
        public int StudentId { get; set; }
        0 references
        public Student Student { get; set; } = null!;
        0 references
        public Exam Exam { get; set; } = null!;
    }
}
```

جدول الإمتحانات مربوط مع **StudentMark** وهو مفتاح فرعي للـ **Subject**

```
namespace ProjectEntity
{
    5 references
    public class Exam
    {
        1 reference
        public int Id { get; set; }
        2 references
        public int SubjectId { get; set; }
        3 references
        public string Date { get; set; }
        3 references
        public int Term { get; set; }
        0 references
        public ICollection<StudentMark> StudentMarks { get; } = new List<StudentMark>();
        0 references
        public Subject Subject { get; set; } = null!;
    }
}
```

جدول **SubjectLecture** وهو جدول لمحاضرات الطلاب وهو مفتاح فرعي للجدول **Subject**

```
namespace ProjectEntity
{
    4 references
    public class SubjectLecture
    {
        1 reference
        public int Id { get; set; }
        1 reference
        public int LectureId { get; set; }
        3 references
        public string Title { get; set; }
        3 references
        public string Content { get; set; }
        0 references
        public Subject Subject { get; set; } = null!;

    }
}
```

وبعد أن قمنا بإنشاء كل الجداول تعريفها ربطها، أنشئنا CLASS يمثل الـ**DataBase** التي لدينا

```
namespace ProjectEntity
{
    3 references
    public class ApplicationDbContext : DbContext
    {

        0 references
        protected override void OnConfiguring(DbContextOptionsBuilder optins) =>
            optins.UseSqlServer(@"Data Source=(localdb)\ProjectsV13;Initial Catalog=ProjectEntityFV2;Integrated Security=True;");
        4 references
        public DbSet<Student> Students { get; set; }
        4 references
        public DbSet<Subject> Subjects { get; set; }
        4 references
        public DbSet<Exam> Exams { get; set; }
        4 references
        public DbSet<StudentMark> StudentMarks { get; set; }
        4 references
        public DbSet<SubjectLecture> SubjectLectures { get; set; }
        3 references
        public DbSet<Department> Departments { get; set; }
    }
}
```

ثم أنشئنا نسخة من Class الداتا بيز لأستطيع التعامل مع الجداول وقمنا باستدعاء كلاس Department واجرينا عليه عملية إضافة وحفظ التغيرات وبعدها أنشئنا حلقة للتعديل وشرط لفحص إذا رقم القسم موجود أولا ومن ثم عمليتي التعديل وحفظ التغيرات

```
class Program
{
    0 references
    static void Main(string[] args)
    {
        var _context = new ApplicationDbContext();
        Department department = new Department
        {

            Name = "برمجيات"
        };
        _context.Departments.Add(department);
        _context.SaveChanges();
        while (true)
        {
            Console.WriteLine("department id");
            int id = int.Parse(Console.ReadLine());
            var department1 = _context.Departments.Find(id);
            if (department1 is null)
            {
                Console.WriteLine("department is not exsiting ");
                continue;
            }
            Console.WriteLine("enter the new Namedepartment");
            string named = Console.ReadLine();
            department1.Name = named;
            _context.Update(department1);
            _context.SaveChanges();
            break;
        }
```

وهنا انشئا نسخة من **Student** وقمنا بالإضافة على البيانات وحفظنا التغيرات وبعدها حلقة للتعديل وشرط لفحص ID الطالب إذا كان موجود او عند الانتهاء منها يتم التعديل وحفظ التغيرات ثم قمنا بعملية حذف

```
var student = new Student
{
    UserName = "hadi",
    FirstName = "Ahmad",
    LastName = "altobaje",
    Email = "ahmadt@gmail.com",
    Phone = "0932244654",
    RegisterDate = 2023,
    DepartmentId = 1
};
_context.Students.Add(student);
_context.SaveChanges();
while (true)
{
    Console.WriteLine("std id");
    int id = int.Parse(Console.ReadLine());
    var std1 = _context.Students.Find(id);
    if (std1 is null)
    {
        Console.WriteLine("student is not exsiting ");
        continue;
    }
    Console.WriteLine("enter the new userName");
    string uname = Console.ReadLine();
    std1.UserName = uname;
    Console.WriteLine("enter the new firstName");
    string fname = Console.ReadLine();
    std1.FirstName = fname;

    Console.WriteLine("enter the new lastName");
    string lname = Console.ReadLine();
    std1.LastName = lname;
    Console.WriteLine("enter the new email");
    string email = Console.ReadLine();
    std1.Email = email;
    Console.WriteLine("enter the new DepartmentId");
    int departmentid1 = int.Parse(Console.ReadLine());
    std1.DepartmentId = departmentid1;
    Console.WriteLine("enter the new phone");
    string phone1 = Console.ReadLine();
    std1.Phone = phone1;
    _context.Update(std1);
    _context.SaveChanges();
    Console.WriteLine("enter id student to delete");
    int id1 = int.Parse(Console.ReadLine());
    _context.Students.DeleteByKey(id1);
    _context.SaveChanges();
    break;
}
```

وبعدها نكتب نفس الكود لباقي الجداول

وأخيرا قمنا بعرض معلومات كل الجداول

```
}
var dep = _context.Departments.ToList();
foreach (var item in dep)
{
    Console.WriteLine("id"+item.Id+"name"+item.Name);
}
Console.WriteLine("--------------------------------------------------------------------------------");
    var std = _context.Students.ToList();
foreach (var item in std)
{
    Console.WriteLine(" stdID : " + item.Id + "user name" +item.UserName+"std name"  + item.FirstName + "last name"+item.LastName+"email"+item.Email+"phone"+item.Phone);
}
Console.WriteLine("--------------------------------------------------------------------------------");
var sub = _context.Subjects.ToList();
foreach (var item in sub)
{
    Console.WriteLine("subID: " + item.Id+"departmentid"+item.DepartmentId+" name " + item.Name +"minimumdegree"+item.MinimumDegree+"term"+item.Term+"year"+item.Year );
}
Console.WriteLine("--------------------------------------------------------------------------------");
var ex = _context.Exams.ToList();
foreach (var item in ex)
{
    Console.WriteLine("exam id"+item.Id+"subjec id"+item.SubjectId+"date"+item.Date+"term"+item.Term);
}
Console.WriteLine("--------------------------------------------------------------------------------");
var stdm = _context.StudentMarks.ToList();
foreach (var item in stdm)
{
    Console.WriteLine("id"+item.Id+"student id"+item.StudentId+"exam id"+item.ExamId+"mark"+item.Mark);
}
Console.WriteLine("--------------------------------------------------------------------------------");
var subl = _context.SubjectLectures.ToList();
foreach (var item in subl)
{
    Console.WriteLine("id"+item.Id+"lecture id"+item.LectureId+"title"+item.Title+"content"+item.Content);
}
Console.WriteLine("--------------------------------------------------------------------------------");
```