ER Core Console Application

1. Se creează o aplicție .NET Core Console.

2. Instalare EF Core

În aplicația creată la pasul anterior, vom instala EF Core folosind Package Manager Console. Pentru aceasta vom selecta Tools -> NuGet Package Manager -> Package Manager Console și vom executa comanda

```
PM> Install-Package Microsoft.EntityFrameworkCore.SqlServer
```

3. Definirea modelului

Entity Framework are nevoie de un model (Entity Data Model) pentru a comunica cu baza de date. Modelul EF include 3 părți:

- Modelul conceptual
- Modelul pentru stocarea datelor
- Maparea dintre cele 2 modele anterioare

În abordarea code first, EF construiește modelul conceptual pe baza claselor de domeniu (clasele pentru entități), a clasei de context și a configurărilor.

EF va folosi acest model pentru imeplementarea oprerațiilor CRUD (Create, Read, Update, Delete).

Pentru acest exemplu, vom crea 2 clase: Student și Course.

```
public class Student
{
	public int StudentId { get; set; }
	public string Name { get; set; }
}

public class Course
{
	public int CourseId { get; set; }
	public string CourseName { get; set; }
}
```

4. Definirea clasei context

Vom define o clasă de context care va deriva din DBContext

```
public class SchoolContext : DbContext
{
	public DbSet<Student> Students { get; set; }
	public DbSet<Course> Courses { get; set; }

	protected override void OnConfiguring(
		DbContextOptionsBuilder optionsBuilder)
```

```
        {
            optionsBuilder.UseSqlServer(@"Server=.;Database=SchoolDB;T
                rusted_Connection=True;");
        }
}
```

5. Adăugarea unei migrări

EF Core conține diferite comenzi de migrare pentru crearea sau actualizarea unei baze de date, pe baza modelului definit în aplicație. În acest moment, nu există nici o bază de date cu numele SchoolDB definite. Pentru crearea acesteia pe baza modelului (entități și context) avem nevoie de o migrație inițială.

Comenzile de migrare se pot executa folosind **NuGet Package Manger Console**, sau **CLI** (Command Line Interface) din .NET.

Dacă se folosește **NuGet Package Manger Console**, se introduce comanda

```
PM> add-migration CreateSchoolDB
```

Nota. Daca comanda nu este recunoscută, atunci introduceți următoarea comandă înainete:

```
PM> Install-Package Microsoft.EntityFrameworkCore.Tools
```

Dacă se folosește **CLI**, se folosește comanda:

```
> dotnet ef migrations add CreateSchoolDB
```

După rularea comenzii, EF Core va crea un director cu numele Migrations unde va salva toate migrările care se vor crea.

Pentru crearea efectivă a bazei de date, vom folosi comanda update-database in Package Manager Console.

```
PM> update-database –verbose
```

Dacă se folosește **CLI**, se folosește comanda:

```
> dotnet ef database update
```

6. Implementarea operațiilor CRUD

Toate operațiile CRUD (Create, Read, Update, Delete) au nevoie de contextul în care trebuie să se execute. Pentru aplicația curentă vom adăuga aceste operații în clasa **SchoolContext**.

6.1. Adăugarea de noi înregistrări

```
public void AddStudent(string name)
{
	var student = new Student()
	{
		Name = name
	};

	Students.Add(student);
	SaveChanges();
}


public void AddCourse(string courseName)
{
	var course = new Course()
	{
		CourseName = courseName
	};

	Courses.Add(course);
	SaveChanges();
}
```

6.2. Actualizarea unei înregistrări

```
public void UpdateStudent(string oldName, string newName)
{
	var std = Students.Where(n => n.Name.Equals(oldName)).FirstOrDefault();
	std.Name = newName;

	Students.Update(std);
	SaveChanges();
}


public void UpdateCourse(string oldName, string newName)
{
	var course = Courses.Where(n => n.CourseName.Equals(oldName)).First();
	course.CourseName = newName;

	Courses.Update(course);
	SaveChanges();
}
```

6.3. Ștergerea unei înregistrări

```
public void DeleteStudent(string name)
{
	var std = Students.Where(n => n.Name.Equals(name)).FirstOrDefault();
	std.Name = name;

	Students.Remove(std);
	SaveChanges();
}
```

```
public void DeleteCourse(string name)
{
	var course = Courses.Where(n => n.CourseName.Equals(name)).First();
	course.CourseName = name;

	Courses.Remove(course);
	SaveChanges();
}
```

6.4. Afișarea datelor

```
public void DisplayStudents()
{
	foreach (var student in Students)
	{
		Console.WriteLine("{0} - {1}", student.StudentId, student.Name);
	}
	Console.WriteLine();
}
```

```
public void DisplayCourses()
{
	foreach (var course in Courses)
	{
	     Console.WriteLine("{0} - {1}", course.CourseId, course.CourseName);
	}
	Console.WriteLine();
}
```

7. Testarea aplicației

```
static void Main(string[] args)
{
    var context = new SchoolContext();

    // Add new Students
    context.AddStudent("Popescu");
    context.AddStudent("Ionescu");
    context.AddStudent("Ptutcea");
    context.AddStudent("Gavrila");
    context.DisplayStudents();

    // Add new Courses
    context.AddCourse("Data Structure");
    context.AddCourse("Java");
    context.AddCourse("Algorithms");
    context.AddCourse(".NET");
    context.DisplayCourses();

    Console.WriteLine("Update first student:");
    context.UpdateStudent(context.Students.First().Name, "New student");
    context.DisplayStudents();

    Console.WriteLine("Update first course:");
    context.UpdateCourse(context.Courses.First().CourseName, "New course");
    context.DisplayCourses();
```

```
        Console.WriteLine("Delete first student:");
        context.DeleteStudent(context.Students.First().Name);
        context.DisplayStudents();

        Console.WriteLine("Delete first course:");
        context.DeleteCourse(context.Courses.First().CourseName);
        context.DisplayCourses();

        Console.ReadLine();
}
```