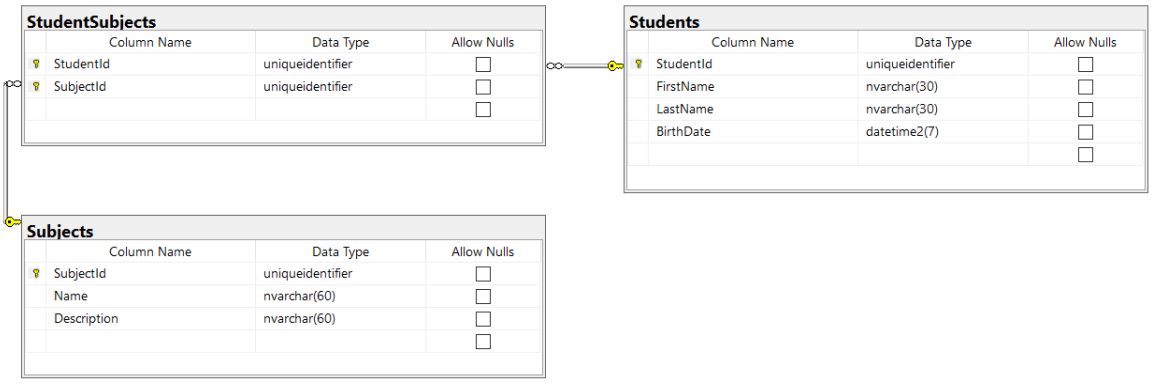
Oefening- Architectuur StudentSubjects Web api maken in ASP.Net Core 3.1 met meerlagen architectuur en EF Core

Deel 1: solution opzetten en EF Core code-first

In deze oefening wordt een meerlagen architectuur gebruikt om een Web app te maken (Web API) om Studenten, Vakken en Inschrijvingen van studenten voor vakken te beheren.

Er wordt gebruik gemaakt van EF Core ORM code-first om een Sql server database aan te maken.

1. Maak een Nieuwe Solution StudentSubjects met hierin de volgende projecten (ASP.NET Core 3.1) :
   - **Web API** (ASP.NET Core 3.1) **project** met naam StudentSubjects
   - **3 class Library** (.NET Core 3.1) **projecten** met namen:
     - Entities
     - Contracts
     - Repository
2. De volgende Tabellen moeten via EF Core code-first aanpak worden gegenereerd :



3. **Installeer de nodig NuGet Packages van EF Core (zowel in Entities project als in StudentSubjects project)**

4. **Voeg het volgende toe aan het Entities project:**
   Voeg aan het project Entities een folder **Models** toe en de volgende classes :

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace Entities.Models
```

```
{
    public class Student
    {
        [Column("StudentId")]
        [Key, DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public Guid Id { get; set; }
        [Required(ErrorMessage = "Student's first name is a required field.")]
        [MaxLength(30, ErrorMessage = "Maximum length for the First Name is 30 characters.")]
        public string FirstName { get; set; }
        [Required(ErrorMessage = "Student's last name is a required field.")]
        [MaxLength(30, ErrorMessage = "Maximum length for the last Name is 30 characters.")]
        public string LastName { get; set; }
        [Required(ErrorMessage = "BirthDate is a required field.")]
        [DataType("date")]
        public DateTime BirthDate { get; set; }
        public ICollection<StudentSubject> StudentSubjects { get; set; }

    }
    public class Subject
    {
        [Column("SubjectId")]
        [Key, DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public Guid Id { get; set; }
        [Required(ErrorMessage = "Subject name is a required field.")]
        [MaxLength(60, ErrorMessage = "Maximum length for the Name is 60 characters.")]
        public string Name { get; set; }
        [Required(ErrorMessage = "Company Description is a required field.")]
        [MaxLength(60, ErrorMessage = "Maximum length for rhe Description is 60 characters")]
        public string Description { get; set; }

        public ICollection<StudentSubject> StudentSubjects { get; set; }
    }

     public class StudentSubject
     {
          public Guid StudentId { get; set; }
          public Student Student { get; set; }

          public Guid SubjectId { get; set; }
          public Subject Subject { get; set; }
     }

}
```

Voeg een folder Configurations toe met hierin de volgende classes :

```
using Entities.Models;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata.Builders;
using System;

namespace Entities.Configuration
{
    public class StudentConfiguration : IEntityTypeConfiguration<Student>
    {
        public void Configure(EntityTypeBuilder<Student> builder)
        {

            {
                builder.HasData
                (
                new Student
                {
                    Id = new Guid("c9d4c053-49b6-410c-bc78-2d54a9991870"),
                    FirstName = "Jos",
                    LastName = "De Klos",
                    BirthDate = DateTime.Now.AddYears(-45)
                },
                new Student
                {
                    Id = new Guid("3d490a70-94ce-4d15-9494-5248280c2ce3"),
```

```
                    FirstName = "Joke",
                    LastName = "De Klos",
                    BirthDate = DateTime.Now.AddYears(-25)
                },
                new Student
                {
                    Id = Guid.NewGuid(),
                    FirstName = "Jan",
                    LastName = "Jansens",
                    BirthDate = DateTime.Now.AddYears(-19)
                }
                ) ;
            }
        }
    }
}
```

```
using Entities.Models;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata.Builders;
using System;
using System.Collections.Generic;
using System.Text;

namespace Entities.Configuration
{
    public class SubjectConfiguration : IEntityTypeConfiguration<Subject>
    {
        public void Configure(EntityTypeBuilder<Subject> builder)
        {
            builder.HasData
            (
            new Subject
            {
                Id = new Guid("80abbca8-664d-4b20-b5de-024705497d4a"),
                Name = "C#",
                Description = "Csharp programming language"

            },
            new Subject
            {
                Id = new Guid("86dba8c0-d178-41e7-938c-ed49778fb52a"),
                Name = "Database",
                Description = "Sql server for beginners"
            },
            new Subject
            {
                Id = new Guid("021ca3c1-0deb-4afd-ae94-2159a8479811"),
                Name = "Web apps",
                Description = "Web applications in ASP.NET Core"
            }
            );
        }
    }
}
```

```
using Entities.Models;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata.Builders;
using System;

namespace Entities.Configuration
{

    public class StudentSubjectConfiguration : IEntityTypeConfiguration<StudentSubject>
    {
        public void Configure(EntityTypeBuilder<StudentSubject> builder)
        {

            {
                builder.HasData
```

```
                (
                new StudentSubject
                {
                    StudentId = new Guid("c9d4c053-49b6-410c-bc78-2d54a9991870"), //Jos
                    SubjectId = new Guid("80abbca8-664d-4b20-b5de-024705497d4a")   //C#

                },
                new StudentSubject
                {
                    StudentId = new Guid("c9d4c053-49b6-410c-bc78-2d54a9991870"), //Jos
                    SubjectId = new Guid("86dba8c0-d178-41e7-938c-ed49778fb52a")  //Database

                },
                new StudentSubject
                {
                    StudentId = new Guid("3d490a70-94ce-4d15-9494-5248280c2ce3"), //Joke
                    SubjectId = new Guid("80abbca8-664d-4b20-b5de-024705497d4a")  //C#
                },
                new StudentSubject
                {
                    StudentId = new Guid("3d490a70-94ce-4d15-9494-5248280c2ce3"), //Joke
                    SubjectId = new Guid("021ca3c1-0deb-4afd-ae94-2159a8479811")  //Web apps
                }
                ) ;
            }
        }
    }
}
```

Voeg een afgeleide class van DbContext toe in het Entities project met naam RepositoryContext:

```
using Entities.Configuration;
using Entities.Models;
using Microsoft.EntityFrameworkCore;

namespace Entities
{
    public class RepositoryContext:DbContext
    {
        public RepositoryContext(DbContextOptions options) : base(options)
        {
        }
        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<StudentSubject>().HasKey(sc => new { sc.StudentId, sc.SubjectId
});

            modelBuilder.Entity<StudentSubject>()
                .HasOne<Student>(sc => sc.Student)
                .WithMany(s => s.StudentSubjects)
                .HasForeignKey(sc => sc.StudentId);


            modelBuilder.Entity<StudentSubject>()
                .HasOne<Subject>(sc => sc.Subject)
                .WithMany(s => s.StudentSubjects)
                .HasForeignKey(sc => sc.SubjectId);

            modelBuilder.ApplyConfiguration(new StudentConfiguration());
            modelBuilder.ApplyConfiguration(new SubjectConfiguration());
            modelBuilder.ApplyConfiguration(new StudentSubjectConfiguration());
        }
        public DbSet<Student> Students { get; set; }
        public DbSet<Subject> Subjects { get; set; }
        public DbSet<StudentSubject> StudentSubjects { get; set; }

    }
}
```

5. **Voeg het volgende toe aan het project StudentSubjects:**
   In **appSettings.json** plaats je de querystring, bv :

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information",
      "Microsoft.EntityFrameworkCore": "Information"
    }
  },
  "ConnectionStrings": {
    "sqlConnection": "server=(localDb)\\MssqlLocalDb; database=StudentSubject; Integrated Security=true"
  },
  "AllowedHosts": "*"
}
```

   In **Startup.cs**: Registreer de RepositoryContext class met de built-in IoC container van ASP.NET Core en haal de connectionstring op via Configuration:

```
    public void ConfigureServices(IServiceCollection services)
    {
            services.AddDbContext<RepositoryContext>(opts =>
opts.UseSqlServer(Configuration.GetConnectionString("sqlConnection"), b =>
            b.MigrationsAssembly("StudentSubjects")));

            services.AddControllers();
     }
```

6. **(Re)Build de solution en zet via de Package manager console de eerste migratie op :**

   PM> Add-Migration Init

7. **Indien geen foutmeldingen, genereer de database:**

   PM> update-database

8. **Controleer of de database goed is aangemaakt, dat de relaties goed staan en of er gegevens in elke tabel zijn**