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

Deel 2: Interfaces maken en Repositories

1. **Voeg het volgende toe aan het Contracts project:**
   Voeg de volgende classes toe :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Linq.Expressions;
using System.Text;

namespace Contracts
{
    public interface IRepositoryBase<T>
    {
        IQueryable<T> FindAll(bool trackChanges);
        IQueryable<T> FindByCondition(Expression<Func<T, bool>> expression,
        bool trackChanges);
        void Create(T entity);
        void Update(T entity);
        void Delete(T entity);
    }
}
```

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

namespace Contracts
{
    public interface IRepositoryManager
    {
        IStudentRepository Student { get; }
        ISubjectRepository Subject { get; }
        IStudentSubjectRepository StudentSubject { get; }
        void Save();

    }
}
```

```
namespace Contracts
{
    public interface IStudentRepository
    {
        IEnumerable<Student> GetAllStudents(bool trackChanges);
        Student GetStudent(Guid studentId, bool trackChanges);
        void CreateStudent(Student student);
        void DeleteStudent(Student company);
    }
}
```

```
using Entities.Models;
using System;
using System.Collections.Generic;
using System.Text;

namespace Contracts
{
    public interface ISubjectRepository
    {
        IEnumerable<Subject> GetAllSubjects( bool trackChanges);
        Subject GetSubject(Guid subjectId, bool trackChanges);
        void DeleteSubject(Subject subject);
        void CreateSubject(Subject subject);

    }
}
```

```
using Entities.Models;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Contracts
{
    public interface IStudentSubjectRepository
    {
        IEnumerable<StudentSubject> GetAllStudentSubjects(bool trackChanges);
        void CreateStudentSubject(StudentSubject studentSubject);
        void DeleteStudentSubject(StudentSubject studentSubject);
        IEnumerable<StudentSubject> GetSubjectsByStudentId(Guid studentId, bool trackChanges);
        IEnumerable<StudentSubject> GetStudentsBySubjectId(Guid subjectId, bool trackChanges);
        StudentSubject GetStudentSubject(StudentSubject studentSubject, bool trackChanges);
    }
}
```

2. Voeg in het project Repositories de volgende classes toe :

```
using Contracts;
using Entities;
using Microsoft.EntityFrameworkCore;
using System;
using System.Linq;
using System.Linq.Expressions;

namespace Repository
{
    public abstract class RepositoryBase<T> : IRepositoryBase<T> where T : class
    {
        protected RepositoryContext RepositoryContext;
        public RepositoryBase(RepositoryContext repositoryContext)
        {
            RepositoryContext = repositoryContext;
        }

        public IQueryable<T> FindAll(bool trackChanges) =>
        !trackChanges ? RepositoryContext.Set<T>() .AsNoTracking() : RepositoryContext.Set<T>();
        public IQueryable<T> FindByCondition(Expression<Func<T, bool>> expression,
        bool trackChanges) =>  !trackChanges ?
RepositoryContext.Set<T>().Where(expression).AsNoTracking() :
RepositoryContext.Set<T>().Where(expression);
        public void Create(T entity) => RepositoryContext.Set<T>().Add(entity);
        public void Update(T entity) => RepositoryContext.Set<T>().Update(entity);
        public void Delete(T entity) => RepositoryContext.Set<T>().Remove(entity);
    }
}
```

```
using Contracts;
using Entities;
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading.Tasks;

namespace Repository
{
    public class RepositoryManager : IRepositoryManager
    {
        private RepositoryContext _repositoryContext;
        private IStudentRepository _studentRepository;
        private ISubjectRepository _subjectRepository;
        private IStudentSubjectRepository _studentSubjectRepository;

        public RepositoryManager(RepositoryContext repositoryContext)
        {
            _repositoryContext = repositoryContext;
        }
        public IStudentRepository Student
        {
            get
            {
                if (_studentRepository == null)
                    _studentRepository = new StudentRepository(_repositoryContext);
                return _studentRepository;
            }
        }
        public ISubjectRepository Subject
        {
            get
            {
                if (_subjectRepository == null)
                    _subjectRepository = new SubjectRepository(_repositoryContext);
                return _subjectRepository;
            }
        }
```

```
        public IStudentSubjectRepository StudentSubject
        {
            get
            {
                if (_studentSubjectRepository == null)
                    _studentSubjectRepository = new StudentSubjectRepository(_repositoryContext);
                return _studentSubjectRepository;
            }
        }

        public void Save()
        {
             _repositoryContext.SaveChanges();
        }
    }

}
```

```
using Contracts;
using Entities;
using Entities.Models;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Repository
{
    public class StudentRepository : RepositoryBase<Student>, IStudentRepository
    {
        public StudentRepository(RepositoryContext repositoryContext)
        : base(repositoryContext)
        {
        }

        public void CreateStudent(Student student)
        {
            Create(student);
        }

        public void DeleteStudent(Student student)
        {
            Delete(student);
        }


        public IEnumerable<Student> GetAllStudents(bool trackChanges)
        {
            return FindAll(trackChanges).OrderBy(s => s.LastName).ThenBy(s => s.FirstName).ToList();

        }

        public Student GetStudent(Guid studentId, bool trackChanges)
        {
           return FindByCondition(c => c.Id.Equals(studentId), trackChanges).SingleOrDefault();
        }

    }

}
```

```
using Contracts;
using Entities;
using Entities.Models;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Repository
{
    public class SubjectRepository : RepositoryBase<Subject>, ISubjectRepository
    {
        public SubjectRepository(RepositoryContext repositoryContext)
        : base(repositoryContext)
        {
        }

        public IEnumerable<Subject> GetAllSubjects(bool trackChanges)
        {
            return FindAll(trackChanges).OrderBy(s => s.Name).ToList();
        }
        public Subject GetSubject(Guid subjectId, bool trackChanges)
        {
            return FindByCondition(c => c.Id.Equals(subjectId), trackChanges)
          .SingleOrDefault();
        }
        public void CreateSubject(Subject subject)
        {
            Create(subject);
        }

        public void DeleteSubject(Subject subject)
        {
            Delete(subject);
        }
    }
}
```

```
using Contracts;
using Entities;
using Entities.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Repository
{
    public class StudentSubjectRepository : RepositoryBase<StudentSubject>,
IStudentSubjectRepository
    {
        public StudentSubjectRepository(RepositoryContext repositoryContext)
        : base(repositoryContext)
        {
        }
        public void CreateStudentSubject(StudentSubject studentSubject)
        {
            Create(studentSubject);
        }
```

```
        public void DeleteStudentSubject(StudentSubject studentSubject)
        {
            Delete(studentSubject);
        }

        public IEnumerable<StudentSubject> GetAllStudentSubjects(bool trackChanges)
        {
            return FindAll(trackChanges).ToList();
        }

        public IEnumerable<StudentSubject> GetStudentsBySubjectId(Guid subjectId, bool
trackChanges)
        {
            return FindByCondition(s => s.SubjectId.Equals(subjectId),
trackChanges).ToList();
        }

        public StudentSubject GetStudentSubject(StudentSubject studentSubject, bool
trackChanges)
        {
            return FindByCondition(s => s.StudentId.Equals(studentSubject.StudentId) &&
s.SubjectId.Equals(studentSubject.SubjectId)
            , trackChanges).SingleOrDefault();
        }

        public IEnumerable<StudentSubject> GetSubjectsByStudentId(Guid studentId, bool
trackChanges)
        {
            return FindByCondition(s => s.StudentId.Equals(studentId),
trackChanges).ToList();

        }
    }
}
```

3. **Voeg aan het project StudentSubjects de volgende controllers toe (API Controller Empty Template) onder de folder Controllers:**

```
namespace StudentSubjects.Controllers
{
    [Route("api/students")]
    [ApiController]
    public class StudentsController : ControllerBase
    {
        private readonly IRepositoryManager _repository;
        public StudentsController(IRepositoryManager repository)
        {
            _repository = repository;
        }
        [HttpGet]
        public IActionResult GetStudents()
        {
            var students =  _repository.Student.GetAllStudents(trackChanges: false);
            return Ok(students);

        }
        [HttpGet("{id}", Name = "StudentById")]
        public IActionResult GetStudent(Guid id)
        {
            var student = _repository.Student.GetStudent(id, trackChanges: false);
            if (student == null)
            {
                return NotFound();
            }
            else
            {

               return Ok(student);
            }
        }

    }

}
```

```
namespace SubjectSubjects.Controllers
{
    [Route("api/subjects")]
    [ApiController]
    public class SubjectsController : ControllerBase
    {
        private readonly IRepositoryManager _repository;
        public SubjectsController(IRepositoryManager repository)
        {
            _repository = repository;
        }
        [HttpGet]
        public  IActionResult GetAllSubjects()
        {

            var subjects =  _repository.Subject.GetAllSubjects(trackChanges: false);

            return Ok(subjects);
        }
        [HttpGet("{id}", Name = "SubjectById")]
        public IActionResult GetSubject(Guid id)
        {
            var subject = _repository.Subject.GetSubject(id, trackChanges: false);
            if (subject == null)
            {
                return NotFound();
            }
            else
            {
                return Ok(subject);
            }
        }

   }

}
```

```
namespace StudentSubjects.Controllers
{
    [Route("api/studentsubjects")]
    [ApiController]
    public class StudentSubjectsController : ControllerBase
    {
        private readonly IRepositoryManager _repository;

        public StudentSubjectsController(IRepositoryManager repository)
        {
            _repository = repository;
        }
        [HttpGet]
        public IActionResult GetAllStudentSubjects()
        {
            var studentSubjects =
_repository.StudentSubject.GetAllStudentSubjects(trackChanges: false);
            return Ok(studentSubjects);
        }
```

```
 [HttpGet("{subjectId}", Name = "StudentsBySubjectId")]
        public IActionResult StudentsBySubjectId(Guid subjectId)
        {
            var subjectStudents =
_repository.StudentSubject.GetStudentsBySubjectId(subjectId, trackChanges: false);

            return Ok(subjectStudents);

        }
    }
}
```

4. (Re)Build en Start je Web API op en test met Postman de volgende http GET Requests :

https://localhost:xxxxx/api/students
https://localhost:xxxxx/api/students/3d490a70-94ce-4d15-9494-5248280c2ce3
https://localhost:xxxxx/api/subjects
https://localhost:xxxxx/api/subjects/80abbca8-664d-4b20-b5de-024705497d4a
https://localhost:xxxxx/api/studentsubjects
https://localhost:xxxxx/api/studentsubjects/80abbca8-664d-4b20-b5de-024705497d4a