ASP.NET Core Project

06/13/2019

Dokumentacija

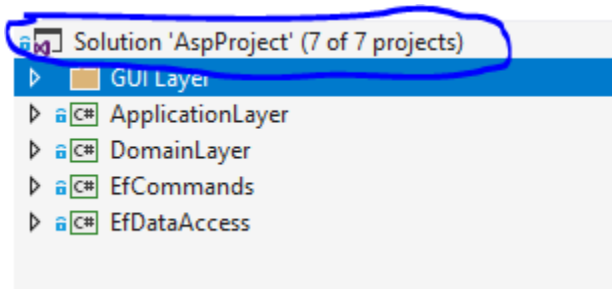
MATERIJA PROJEKTA:

SOLUTIONS AND PROJECT

- ARHITEKTURALNI PRINCIPI RAZVOJA SOFTVERA (DEPENDENCIES)
- VISESLOJNA ARHITEKTURA
- S.O.L.I.D PRINCIPI PROJEKTOVANJA
- -- INTERFEJSI --
- ORM (Objektno relaciono preslikavanje)
- GRAFISKO KORISNICKI SLOJ (WEB API, WEB APLIKACIJA ...)
- .NET Core

0) SOLUTION AND PROJECT

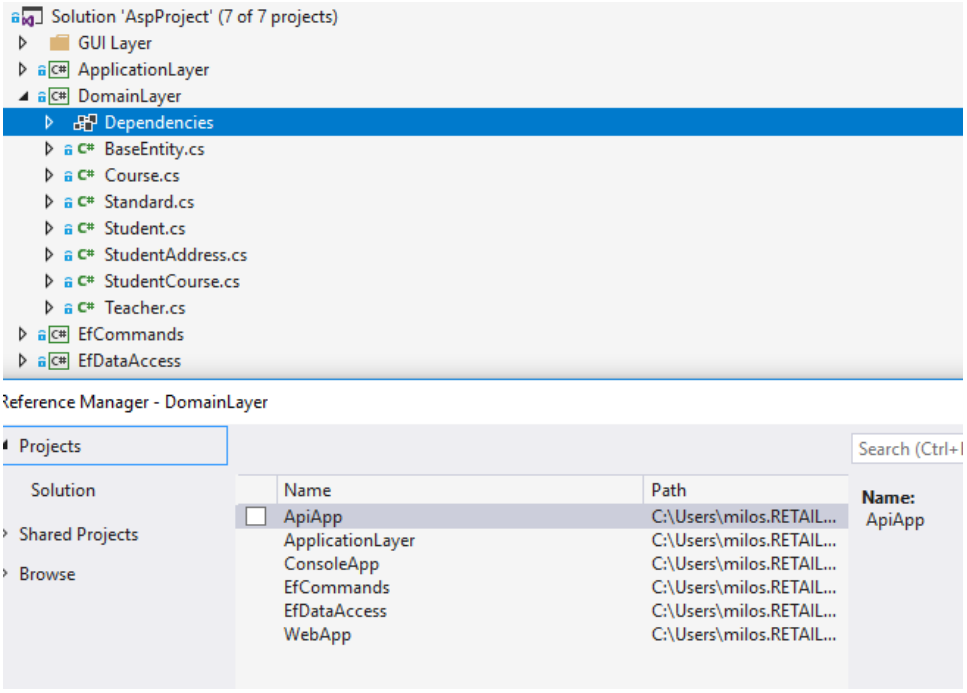
SOLUTION –Predstavlja okvir(folder) za vise projekata. On nam sluzi samo kao okvir za sve projekte koji su neophodni za rad aplikacije.



I) ARHITEKTURALNI PRINCIPI RAZVOJA SOFTVERA

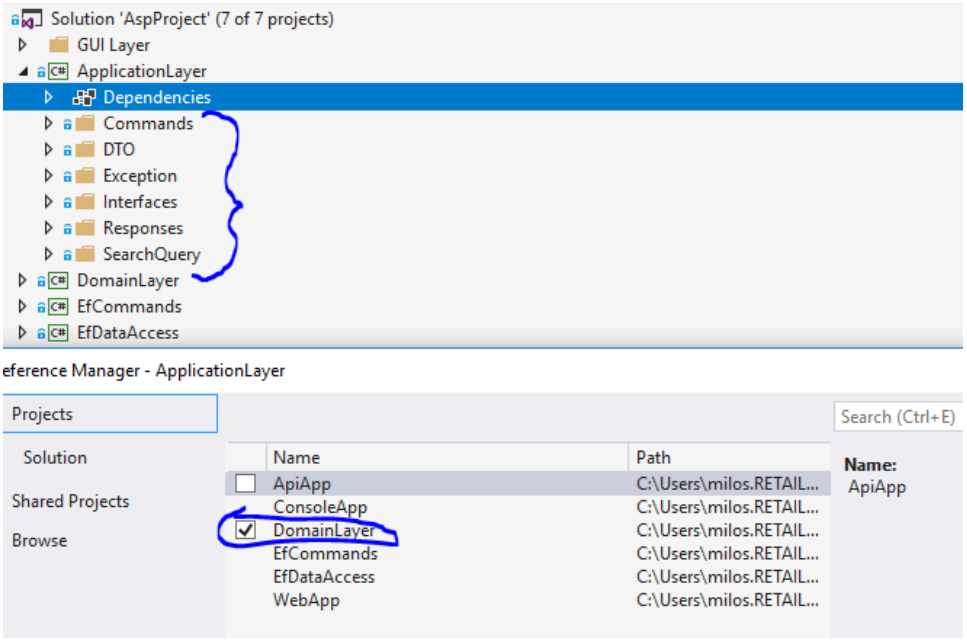
POD OVIM TERMIN SE SMATRA DIREKTNA ZAVISNOST PROJEKATA.
Svaki deo projekta treba da bude radvojen od onog drugog I da bude zaduzen samo za jednu celinu u celom projektu. Ceo softver treba da bude projektovan tako da najbitniji delovi aplikacije nimaju nikakvu zavisnost od onih koji su manje bitni. Obicno se kreira po projekat za d sa domenima, bazom, poslovnu logiku, grafiski sloj...
Projekat je najmanja jedinica kompaliranja. Sastoji se od klasa, interfejsa, struktura - .cs fajlova. Prilikom kompaliranja projekat se svodi na .dll ili .exe fajl I svi se oni nalaze u okviru solution-a.

1) Deo aplikacije koje nezavisi niodkoga je DOMAIN_LAYER:



Na ovoj slici vidimo da DOMAIN_LAYER sloj je naj nezavisniji I samim tim sloj koji je u samom centru arhitektute softvera.
U njemu se nalaze Entiteti tj., tabele. I on se jos naziva domenski sloj.

2) APPLICATION-LAYER - Sledeci deo aplikacije po zavisnosti je deo za POSLOVNU LOGIKU



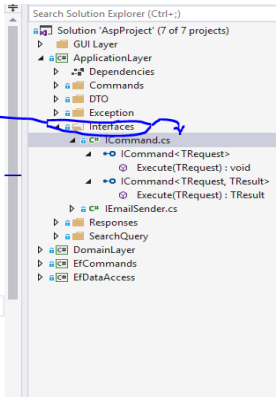
APLICATION_LAYER sloj ima zavisnost samo prema DOMAIN_LAYER sloju.

INTERFEJSI

Sadrze samo potpise metoda, svojstava, dogadjaja I indeksera. Klasa koja implementira interfejs mora da implementira sve clanove definisane interfejosm. Svi clanovi propisani interfejsom su javni. Klasa koja implementira interfejs se smatra primerom tog interfejsa. Referencu klase koja implementira interfejs moguce je cuvati u referenci pomenutog interfejsa. U nastavku cemo videti klasicnu primenu interfejsa...

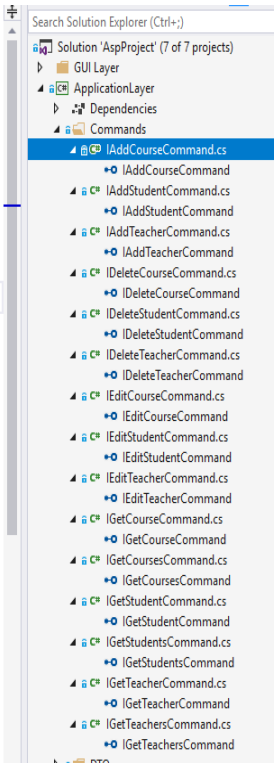
- OVAJ sloj u sebi ima deo koji se odnosi na implementaciju dve vrse komandi. I to one koje nece vratiti nista (INSERT, UPDATE I DELETE), I one druge koje ce I primiti podatke I vratiti nazad kao sto je (SELECT) *2

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

namespace ApplicationLayer.Interfaces
{
    9 references
    public interface ICommand<TRequest>
    {
        21 references | 0 exceptions
        void Execute(TRequest request);
    }

    6 references
    public interface ICommand<TRequest, TResult> where TResult : class
    {
        15 references | 0 exceptions
        TResult Execute(TRequest request);
    }
}
```



- COMMANDS – Jedan deo biznis logike koji je zaduzen da kaze sta aplikacija treba da radi.

```
using ApplicationLayer.DTO;
using ApplicationLayer.Interfaces;
using System;
using System.Collections.Generic;
using System.Text;

namespace ApplicationLayer.Commands
{
    4 references
    public interface IAddCourseCommand : ICommand<CreateCourseDto>
    {
    }
}
```

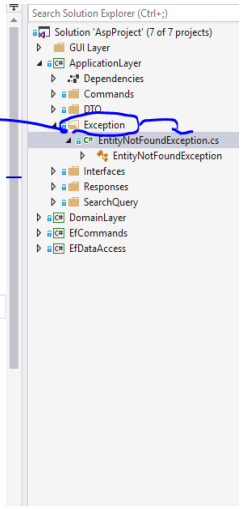


- DTO – (Data transfer object) Sloj koji je zaduzen za prenos podatak.

```
using System.Text;

namespace ApplicationLayer.DTO
{
    25 references
    public class StudentDto
    {

        [Required(ErrorMessage = "This filed is required! Please, try again!")]
        6 references | 0 exceptions
        public int Id { get; set; }

        [Required(ErrorMessage = "This filed is required! Please, try again!")]
        [RegularExpression("^[A-Z][a-z]{2,20}$", ErrorMessage = "The name isn't correct. Please, t
        12 references | 0 exceptions
        public string StudentName { get; set; }

        [Required(ErrorMessage = "This filed is required!")]
        [RegularExpression("^[0-9]{3,8}$", ErrorMessage = "The index number isn't correct. Please,
        12 references | 0 exceptions
        public int NumberIndex { get; set; }

        12 references | 0 exceptions
        public int StudyYear { get; set; }

        0 references | 0 exceptions
        public int NumberPhone { get; set; }
        0 references | 0 exceptions
        public string Natioanality { get; set; }
        0 references | 0 exceptions
        public int BirthDate { get; set; }
```

* EXCEPTION – zaduzen samo da prihvati bacene izuzetke.

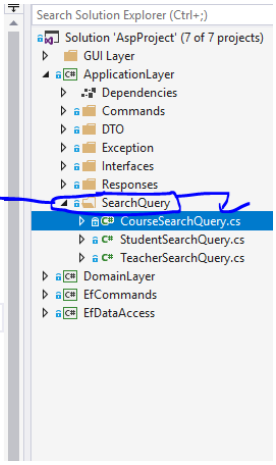
```
using System;
using System.Collections.Generic;
using System.Text;

namespace ApplicationLayer.Exceptions
{
    28 references
    public class EntityNotFoundException : Exception
    {
        8 references | 0 exceptions
        public EntityNotFoundException(string entity) : base($"{entity} doesn't exist")
        {

        }

        10 references | 0 exceptions
        public EntityNotFoundException()
        {

        }
    }
}
```



- SEARCHQUERY – za filtriranje podataka preko searhquery

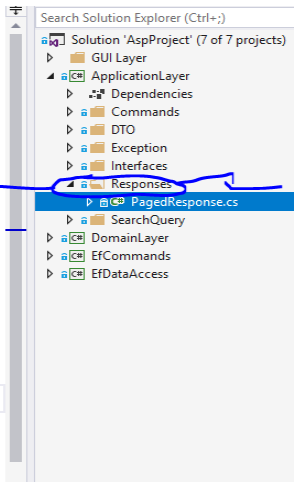
Ono sto cemo vratiti kao rezultat.

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

namespace ApplicationLayer.SearchQuery
{
    3 references
    public class CourseSearchQuery
    {
        2 references | 0 exceptions
        public string CourseName { get; set; }
        3 references | 0 exceptions
        public int PerPage { get; set; } = 1;
        2 references | 0 exceptions
        public int PageNumber { get; set; } = 1;
    }
}
```



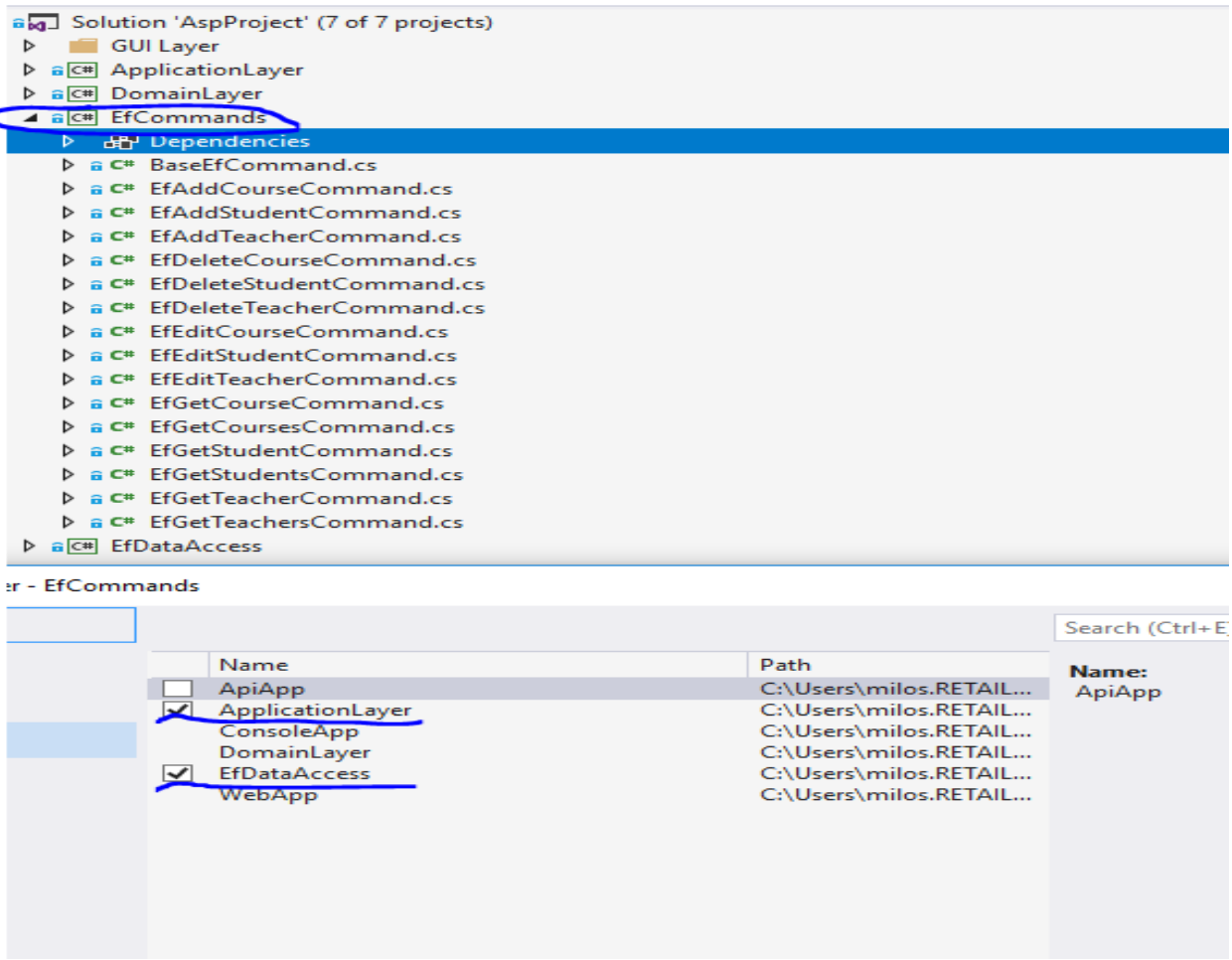
- RESPONSES – PAGINACIJA I PRETRAGA PREKO GET-ENT POINT-A

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

namespace ApplicationLayer.Responses
{
    3 references
    public class PagedResponse<T>
    {
        1 reference | 0 exceptions
        public int TotalCount { get; set; }
        1 reference | 0 exceptions
        public int PagesCount { get; set; }
        1 reference | 0 exceptions
        public int CurrentPage { get; set; }
        1 reference | 0 exceptions
        public IEnumerable<T> Data { get; set; }
    }
}
```

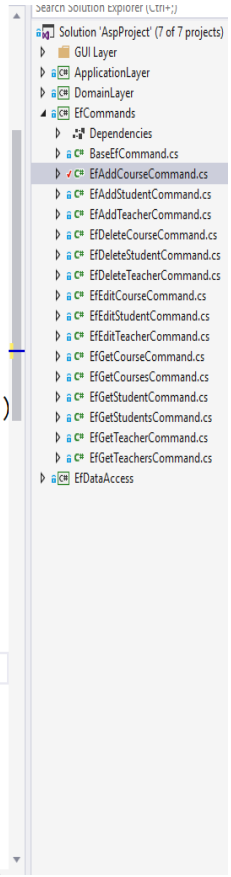


3) EFCOMMANDS – Sloj koji je zaduzen za implementaju. On zavisi od APPLICATION_LAYER I zavisi od EFDATAACCES_LAYER.

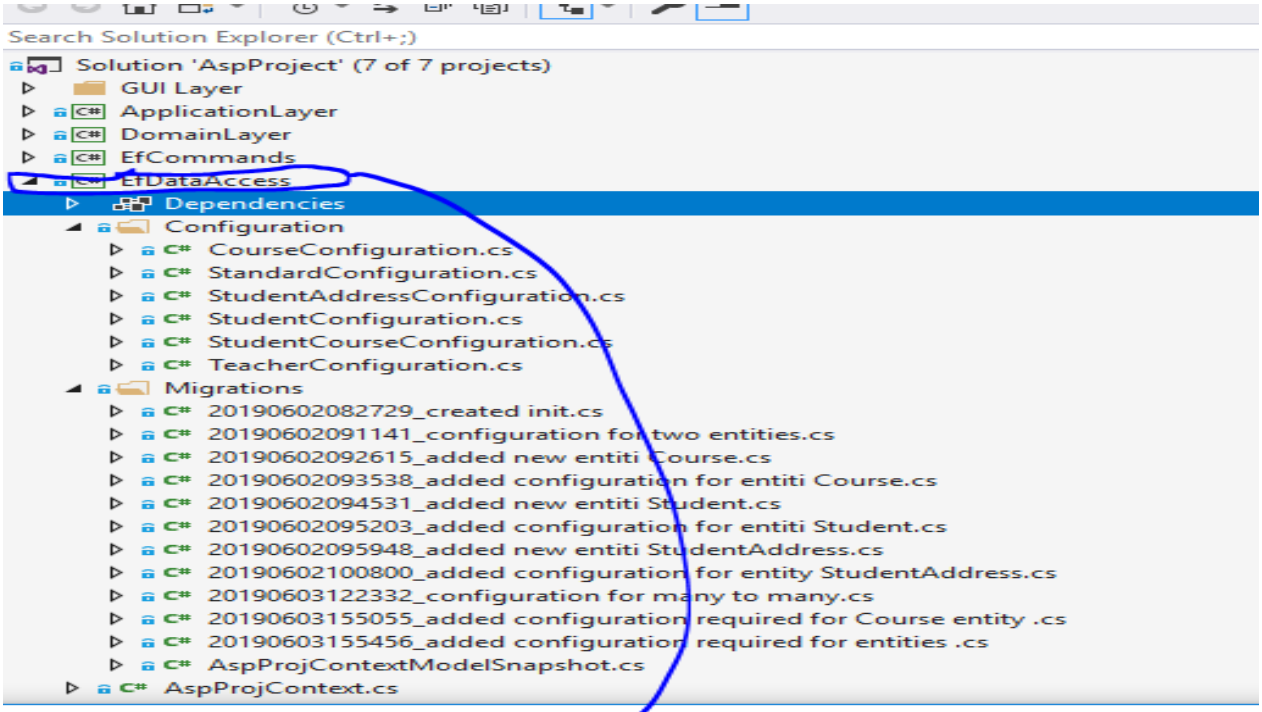
Ispred naziva stoji Ef zato sto se radi sa entity framvorkom ali ako bi promenili ovaj sloj je podlozan promeni na neku drugu tehnologiju.

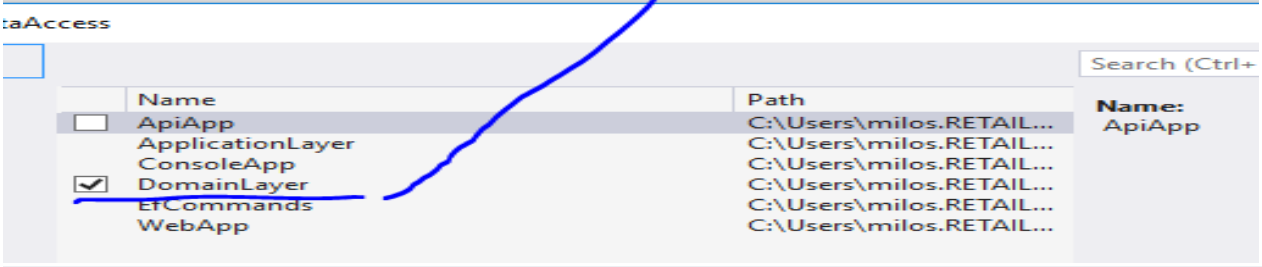


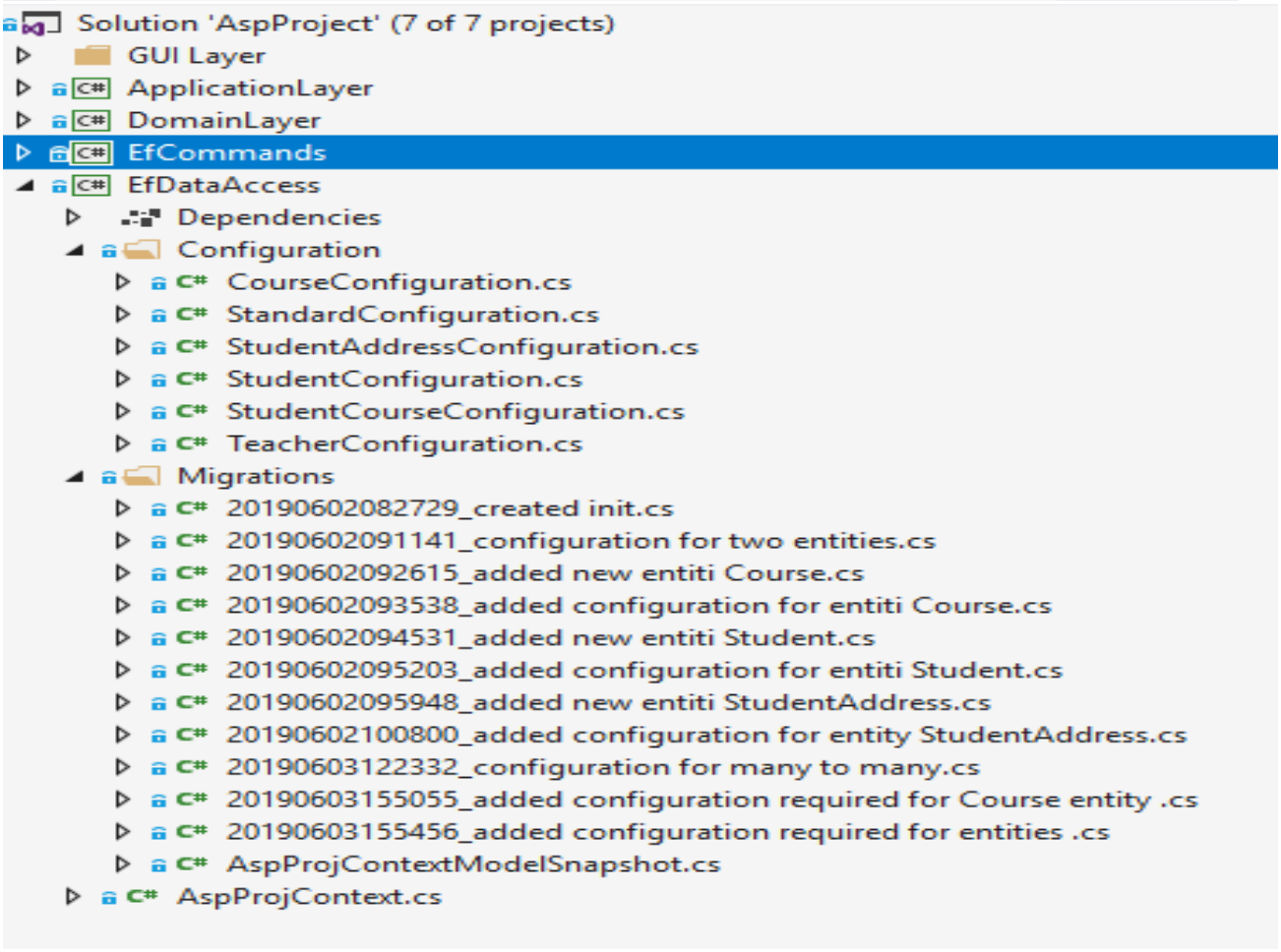
```
namespace EfCommands
{
    2 references
    public class EfAddCourseCommand : BaseEfCommand, IAddCourseCommand
    {
        0 references | 0 exceptions
        public EfAddCourseCommand(AspProjContext context): base(context)
        {

        }
        21 references | 0 exceptions
        public void Execute(CreateCourseDto request)
        {
            if (_context.Courses.Any(c => c.CourseName == request.CourseName && c.Location == request.Location ))
            {
                throw new EntityNotFoundException();
            }

            if (!_context.Teachers.Any(t => t.Id == request.TeacherId))
            {
                throw new EntityNotFoundException("Teachers");
                // Message -> Teacher doesn't exist
            }

            _context.Courses.Add(new Course
            {
                CourseName = request.CourseName,
                Description = request.Description,
```



4) EFDATA_ACCES – Sloj koje je zaduzen za komunikaciju sa provajderom tj., vendorom date baze podataka. U ovom slucaju mi koristiomo SQL kao relacionu bazu podataka, ali moze se implementirati I bilo koje druge tehnologije.


Search Solution Explorer (Ctrl+;)
Solution 'AspProject' (7 of 7 projects)
GUI Layer
ApplicationLayer
DomainLayer
EfCommands
EfDataAccess
Dependencies
Configuration
CourseConfiguration.cs
StandardConfiguration.cs
StudentAddressConfiguration.cs
StudentConfiguration.cs
StudentCourseConfiguration.cs
TeacherConfiguration.cs
Migrations
20190602082729_created init.cs
20190602091141_configuration for two entities.cs
20190602092615_added new entiti Course.cs
20190602093538_added configuration for entiti Course.cs
20190602094531_added new entiti Student.cs
20190602095203_added configuration for entiti Student.cs
20190602095948_added new entiti StudentAddress.cs
20190602100800_added configuration for entity StudentAddress.cs
20190603122332_configuration for many to many.cs
20190603155055_added configuration required for Course entity .cs
20190603155456_added configuration required for entities .cs
AspProjContextModelSnapshot.cs
AspProjContext.cs


taAccess
Search (Ctrl+
Name
Path
ApiApp
C:\Users\milos.RETAIL...
ApplicationLayer
C:\Users\milos.RETAIL...
ConsoleApp
C:\Users\milos.RETAIL...
DomainLayer
C:\Users\milos.RETAIL...
EfCommands
C:\Users\milos.RETAIL...
WebApp
C:\Users\milos.RETAIL...
Name:
ApiApp


Solution 'AspProject' (7 of 7 projects)
GUI Layer
ApplicationLayer
DomainLayer
EfCommands
EfDataAccess
Dependencies
Configuration
CourseConfiguration.cs
StandardConfiguration.cs
StudentAddressConfiguration.cs
StudentConfiguration.cs
StudentCourseConfiguration.cs
TeacherConfiguration.cs
Migrations
20190602082729_created init.cs
20190602091141_configuration for two entities.cs
20190602092615_added new entiti Course.cs
20190602093538_added configuration for entiti Course.cs
20190602094531_added new entiti Student.cs
20190602095203_added configuration for entiti Student.cs
20190602095948_added new entiti StudentAddress.cs
20190602100800_added configuration for entity StudentAddress.cs
20190603122332_configuration for many to many.cs
20190603155055_added configuration required for Course entity .cs
20190603155456_added configuration required for entities .cs
AspProjContextModelSnapshot.cs
AspProjContext.cs

------- AspProjContext -----

```
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Text;
using DomainLayer;
using EfDataAccess.Configuration;

namespace EfDataAccess
{
    32 references
    public class AspProjContext : DbContext
    {
        10 references | 0 exceptions
        public DbSet<Teacher> Teachers { get; set; }
        3 references | 0 exceptions
        public DbSet<Standard> Standards { get; set; }
        7 references | 0 exceptions
        public DbSet<Course> Courses { get; set; }
        8 references | 0 exceptions
        public DbSet<Student> Students { get; set; }
        0 references | 0 exceptions
        public DbSet<StudentAddress> StudentAddresses { get; set; }

        0 references | 0 exceptions
        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            optionsBuilder.UseSqlServer(@"Data Source=.\SQLEXPRESS;Initial Catalog=DbAspProject;Integrated Security=True");
        }

        0 references | 0 exceptions
        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.ApplyConfiguration(new TeacherConfiguration());
            modelBuilder.ApplyConfiguration(new StandardConfiguration());
            modelBuilder.ApplyConfiguration(new CourseConfiguration());
            modelBuilder.ApplyConfiguration(new StudentConfiguration());
            modelBuilder.ApplyConfiguration(new StudentAddressConfiguration());
            modelBuilder.ApplyConfiguration(new StudentCourseConfiguration());
        }
    }
}
```

----------CONFIGURATION----------

A) course_configuration

```
using DomainLayer;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Metadata.Builders;

using System;

using System.Collections.Generic;

using System.Text;


namespace EfDataAccess.Configuration

{

  public class CourseConfiguration : IEntityTypeConfiguration<Course>

  {
```

```
    public void Configure(EntityTypeBuilder<Course> builder)
    {
        builder.Property(c => c.CourseName)
            .HasMaxLength(40)
            .IsRequired();
        builder.Property(c => c.Location)
            .HasMaxLength(50)
            .IsRequired();
        builder.Property(c => c.Description)
            .IsRequired();

        builder.Property(c => c.CreatedAt)
            .HasDefaultValueSql("GETDATE()");

        builder.HasMany(c => c.CourseStudents)
            .WithOne(cs => cs.Course)
            .HasForeignKey(cs => cs.CourseId)
            .OnDelete(DeleteBehavior.Restrict);

    }
  }
}
```

B) standard_configuration

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

```
namespace EfDataAccess.Configuration
{
   public class StandardConfiguration : IEntityTypeConfiguration<Standard>
   {
      public void Configure(EntityTypeBuilder<Standard> builder)
      {
         builder.Property(s => s.StandardName)
            .HasMaxLength(30)
            .IsRequired();
         builder.Property(s => s.Description)
            .IsRequired();

         builder.Property(s => s.CreatedAt)
            .HasDefaultValueSql("GETDATE()");

      }
   }
}
```

C) student_address_configuration

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

namespace EfDataAccess.Configuration
{
   public class StudentAddressConfiguration : IEntityTypeConfiguration<StudentAddress>
   {
```

```
    public void Configure(EntityTypeBuilder<StudentAddress> builder)
    {
        builder.Property(sa => sa.Address)
            .HasMaxLength(50)
            .IsRequired();
        builder.Property(sa => sa.City)
            .HasMaxLength(30)
            .IsRequired();

        builder.Property(sa => sa.CreatedAt)
            .HasDefaultValueSql("GETDATE()");

        builder.HasIndex(sa => sa.StudentId)
            .IsUnique();
    }
  }
}
```

D) student_configuration

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

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

```
    {
        builder.Property(std => std.StudentName)
            .HasMaxLength(30)
            .IsRequired();
        builder.Property(std => std.Natioanality)
            .HasMaxLength(40)
            .IsRequired();

        builder.Property(std => std.CreatedAt)
            .HasDefaultValueSql("GETDATE()");


        builder.HasMany(std => std.StudentCourses)
            .WithOne(sc => sc.Student)
            .HasForeignKey(sc => sc.StudentId)
            .OnDelete(DeleteBehavior.Restrict);

    }
  }
}
```

E) student_course_configuration

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

namespace EfDataAccess.Configuration
{
```

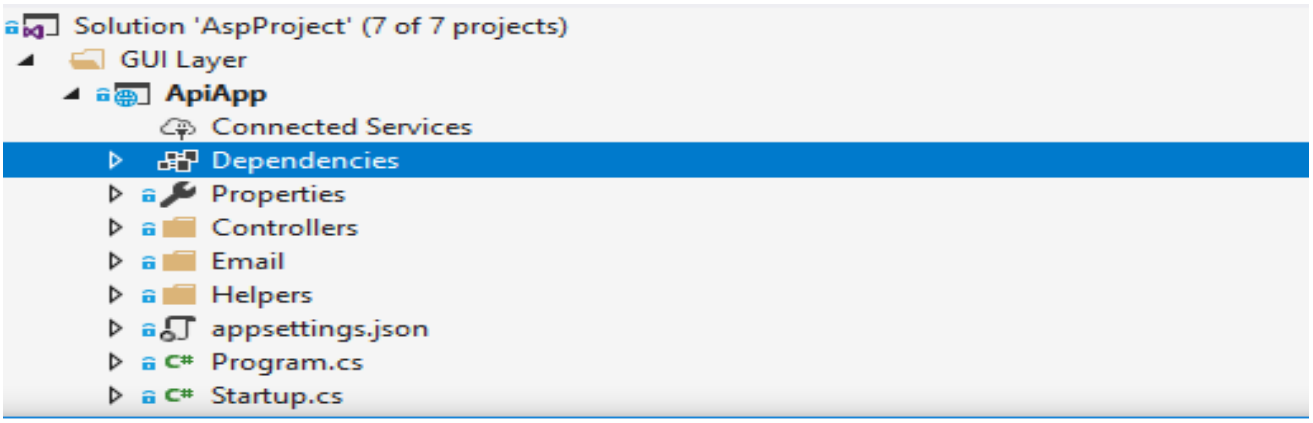
```
    public class StudentCourseConfiguration : IEntityTypeConfiguration<StudentCourse>
    {
        public void Configure(EntityTypeBuilder<StudentCourse> builder)
        {
            builder.HasKey(sc => new { sc.StudentId, sc.CourseId });
        }
    }
}
```

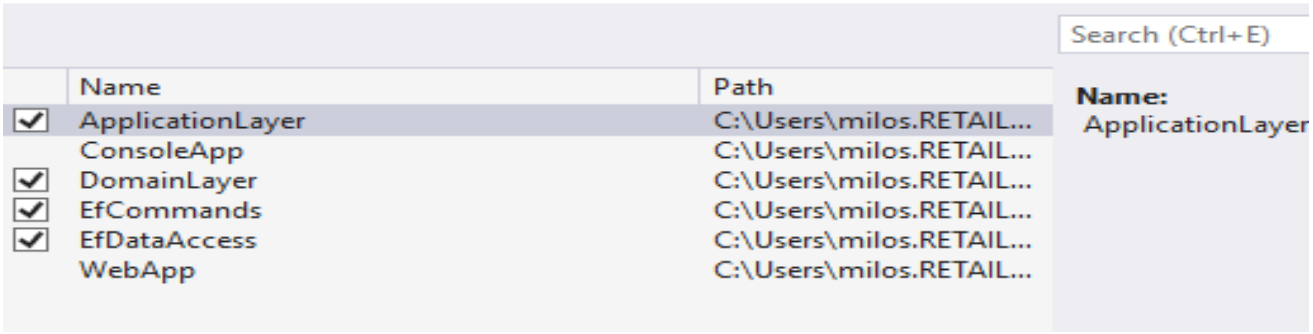
F) teacher_configuration

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

namespace EfDataAccess.Configuration
{
    public class TeacherConfiguration : IEntityTypeConfiguration<Teacher>
    {
        public void Configure(EntityTypeBuilder<Teacher> builder)
        {
            builder.Property(t => t.TFirstName)
                .HasMaxLength(30)
                .IsRequired();
            builder.Property(t => t.TLastName)
                .HasMaxLength(30)
                .IsRequired();
            builder.Property(t => t.Nationality)
                .HasMaxLength(50)
```

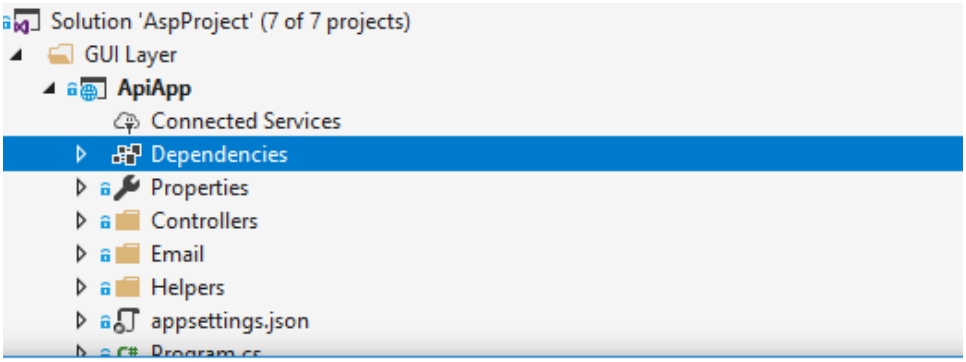
```
            .IsRequired();

        builder.Property(t => t.Description)

            .IsRequired();


        builder.Property(t => t.CreatedAt)

            .HasDefaultValueSql("GETDATE()");




    }

  }

}
```

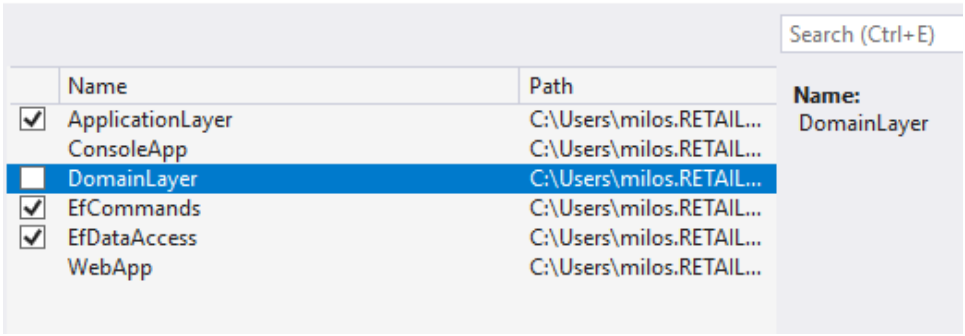
5) GUI_layer – sloj koji u sebi ima aplikacije koje imaju main metodu, tj., oni se izvrsavaju. sloj koji zavisi od svakoj gore navedenog slaja. Ako bi posmatrali arhitekturu kroz krugove ovaj sloj bi zavisio od svakog sloja koji je unutar njegovog kruga. Ali veoma je bitno da nezavisi od projekata koji se nalaze u ovom sloju(GUI_layer)





I na ovaj nacin bi mogao da se poveze





--------API_Aplication------

1) Controllers

A) Auth_controller

B)Course_controller

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using ApplicationLayer.Commands;

using ApplicationLayer.DTO;

using ApplicationLayer.Exceptions;

using ApplicationLayer.SearchQuery;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;


namespace ApiApp.Controllers

{

   [Route("api/[controller]")]

   [ApiController]

   public class CourseController : ControllerBase

   {

      private readonly IGetCoursesCommand _getCommandCourses;

      private readonly IGetCourseCommand _getCommandCourse;

      private readonly IDeleteCourseCommand _delCommandCourse;

      private readonly IEditCourseCommand _editCommandCourse;

      private readonly IAddCourseCommand _addCommandCourse;
```

```
    public CourseController(IGetCoursesCommand getCommandCourses, IGetCourseCommand getCommandCourse,
IDeleteCourseCommand delCommandCourse, IEditCourseCommand editCommandCourse, IAddCourseCommand
addCommandCourse)
    {
        _getCommandCourses = getCommandCourses;
        _getCommandCourse = getCommandCourse;
        _delCommandCourse = delCommandCourse;
        _editCommandCourse = editCommandCourse;
        _addCommandCourse = addCommandCourse;
    }

    // GET: api/Course
    [HttpGet]
    [ProducesResponseType(200)]
    public ActionResult<IEnumerable<CourseDto>> Get([FromQuery]CourseSearchQuery search)
    {
        var resultCourses = _getCommandCourses.Execute(search);
        return Ok(resultCourses); //200
    }

    // GET: api/course/5
    [HttpGet("{id}")]
    [ProducesResponseType(200)]
    [ProducesResponseType(404)]
    public ActionResult<IEnumerable<CourseDto>> Get(int id)
    {
        try
        {
            var resultCourse = _getCommandCourse.Execute(id);
            return Ok(resultCourse); //200
```

```
    }
    catch
    {
        return NotFound(); //404
    }
}

// POST: api/course
[HttpPost]
[ProducesResponseType(201)]
[ProducesResponseType(500)]
public ActionResult<IEnumerable<CreateCourseDto>> Post([FromBody] CreateCourseDto dto)
{
    try
    {
        _addCommandCourse.Execute(dto);
        return StatusCode(StatusCodes.Status201Created);
    }
    catch (EntityNotFoundException e)
    {
        return UnprocessableEntity(e.Message);
    }
    catch (Exception)
    {
        return StatusCode(StatusCodes.Status500InternalServerError);
    }
}

// PUT: api/course/5
```

```
[HttpPut("{id}")]
[ProducesResponseType(204)]
[ProducesResponseType(500)]
public ActionResult<IEnumerable<CreateCourseDto>> Put(int id, [FromBody] CreateCourseDto dto)
{
   dto.Id = id;
   try
   {
      _editCommandCourse.Execute(dto);
      return NoContent();
   }
   catch(EntityNotFoundException e)
   {
      if(e.Message == "Course doesn't exist.")
      {
         return NotFound(e.Message);
      }

      return UnprocessableEntity(e.Message);
   }
   catch(Exception)
   {
      return StatusCode(500, "error");
   }
}

// DELETE: api/course/5
[HttpDelete("{id}")]
[ProducesResponseType(204)]
```

```
        [ProducesResponseType(404)]
        [ProducesResponseType(500)]
        public ActionResult<IEnumerable<CourseDto>> Delete(int id)
        {
            try
            {
                _delCommandCourse.Execute(id);
                return Ok();
            }
            catch
            {
                return NotFound();
            }
        }
    }
}
```

C) Gmail_controller

```
 using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Mail;
using System.Threading.Tasks;
using ApplicationLayer.Interfaces;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;

namespace ApiApp.Controllers
{
```

```
[Route("api/[controller]")]
[ApiController]
public class GmailController : ControllerBase
{
   private IEmailSender sender;

   public GmailController(IEmailSender sender)
   {
      this.sender = sender;
   }

   // GET: api/Mail
   [HttpGet]
   public void Get(string email)
   {

      sender.Subject = "Registration is accept";
      sender.ToEmail = email;
      sender.Body = "Midsda";
      sender.Send();

   }

   // GET: api/Mail/5
   [HttpGet("{id}")]
   public string Get(int id)
   {
      return "value";
   }
```

```
        // POST: api/Mail
        [HttpPost]
        public void Post([FromBody] string value)
        {
        }

        // PUT: api/Mail/5
        [HttpPut("{id}")]
        public void Put(int id, [FromBody] string value)
        {
        }

        // DELETE: api/ApiWithActions/5
        [HttpDelete("{id}")]
        public void Delete(int id)
        {
        }
    }
}
```

D) Student_controller

```
using System;
using Microsoft.AspNetCore.Http;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using ApplicationLayer.Commands;
using ApplicationLayer.DTO;
using ApplicationLayer.SearchQuery;
using Microsoft.AspNetCore.Mvc;
```

```
using ApplicationLayer.Exceptions;


namespace ApiApp.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class StudentsController : ControllerBase
    {
        private IGetStudentsCommand _getCommandStds;
        private IGetStudentCommand _getCommandStd;
        private IAddStudentCommand _addCommandStd;
        private IDeleteStudentCommand _delCommandStd;
        private IEditStudentCommand _editCommandStd;


        public StudentsController(IGetStudentsCommand getCommandStds, IGetStudentCommand getCommandStd,
IAddStudentCommand addCommandStd, IDeleteStudentCommand delCommandStd, IEditStudentCommand editCommandStd)
        {
            _getCommandStds = getCommandStds;
            _getCommandStd = getCommandStd;
            _addCommandStd = addCommandStd;
            _delCommandStd = delCommandStd;
            _editCommandStd = editCommandStd;
        }



        /// <summary>
        /// Returns all Students that match provided query
        /// </summary>
        /// <remarks>
        /// Sample request:
```

```
///
///    GET students
///    {
///       "id": 1,
///       "isDeleted" : false
///       "StudenName": "Nikola",
///       "StudyYear": 3,
///       "NumberIndex" : "6543324",
///       "Nationality" : "Srpsko",
///       "BirthDate" : "14121994",
///
///    }
///
/// </remarks>
/// <param name="item"></param>
/// <returns>A newly created StudentDto</returns>
/// <response code="201">Returns the Ok</response>
// GET api/students
[HttpGet]
[ProducesResponseType(200)]
public ActionResult<IEnumerable<StudentDto>> Get([FromQuery]StudentSearchQuery query)
{
   return Ok(_getCommandStds.Execute(query)); //200
}

// GET api/student/5
[HttpGet("{id}")]
[ProducesResponseType(200)]
[ProducesResponseType(404)]
```

```
public ActionResult<IEnumerable<StudentDto>> Get(int id)
{
   try
   {
      var resultStd = _getCommandStd.Execute(id);
      return Ok(resultStd);  // 200
   }
   catch
   {
      return NotFound(); // 404
   }



}

// POST api/student
[HttpPost]
[ProducesResponseType(201)]
[ProducesResponseType(500)]
public ActionResult<IEnumerable<CreateStudentDto>> Post([FromBody] CreateStudentDto dto)
{
   try
   {
      _addCommandStd.Execute(dto);
      return StatusCode(StatusCodes.Status201Created);
   }
   catch (EntityNotFoundException e)
   {
      return UnprocessableEntity(e.Message);
```

```
    }
    catch (Exception)
    {
        return StatusCode(StatusCodes.Status500InternalServerError);
    }

   // return Created("url", null);  // 201
}

// PUT api/students/5
[HttpPut("{id}")]
[ProducesResponseType(204)]
[ProducesResponseType(500)]
public ActionResult<IEnumerable<CreateStudentDto>> Put(int id, [FromBody] CreateStudentDto dto)
{
    dto.Id = id;
    try
    {
        _editCommandStd.Execute(dto);
        return NoContent();
    }
    catch (EntityNotFoundException e)
    {
        if (e.Message == "Student doesn't exist.")
        {
            return NotFound(e.Message);
        }

        return UnprocessableEntity(e.Message);
```

```
    }
    catch (Exception)
    {
        return StatusCode(500, "error");
    }
}

// DELETE api/student/5
[HttpDelete("{id}")]
[ProducesResponseType(204)]
[ProducesResponseType(404)]
[ProducesResponseType(500)]
public ActionResult<IEnumerable<StudentDto>> Delete(int id)
{
    try
    {
        _delCommandStd.Execute(id);
        return NoContent();  //204
    }
    catch(EntityNotFoundException e)
    {
        return NotFound(e.Message);  //404
    }
    catch (Exception)
    {
        return StatusCode(500, "It's not working"); //500
    }
```

```
    }
  }
}
```

E) Teacher_controller

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using ApplicationLayer.Commands;
using ApplicationLayer.DTO;
using ApplicationLayer.Exceptions;
using ApplicationLayer.SearchQuery;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;

namespace ApiApp.Controllers
{
  [Route("api/[controller]")]
  [ApiController]
  public class TeacherController : ControllerBase
  {
    private IGetTeachersCommand _getCommandTeachers;
    private IGetTeacherCommand _getCommandTeacher;
    private IDeleteTeacherCommand _delCommandTeacher;
    private IAddTeacherCommand _addCommandTeacher;
    private IEditTeacherCommand _editCommandTeacher;

    public TeacherController(IGetTeachersCommand getCommandTeachers, IGetTeacherCommand getCommandTeacher,
IDeleteTeacherCommand delCommandTeacher, IAddTeacherCommand addCommandTeacher, IEditTeacherCommand
editCommandTeacher)
```

```
{
    _getCommandTeachers = getCommandTeachers;
    _getCommandTeacher = getCommandTeacher;
    _delCommandTeacher = delCommandTeacher;
    _addCommandTeacher = addCommandTeacher;
    _editCommandTeacher = editCommandTeacher;
}

// GET: api/Teacher
[HttpGet]
[ProducesResponseType(200)]
public ActionResult<IEnumerable<TeacherDto>> Get([FromQuery]TeacherSearchQuery search)
{
    var resultTeachers = _getCommandTeachers.Execute(search);
    return Ok(resultTeachers); //200
}

// GET: api/Teacher/5
[HttpGet("{id}")]
[ProducesResponseType(200)]
[ProducesResponseType(404)]
public ActionResult<IEnumerable<TeacherDto>> Get(int id)
{
    try
    {
        var resultTeacher = _getCommandTeacher.Execute(id);
        return Ok(resultTeacher);  //200
    }
    catch
```

```
    {
        return NotFound(); //404
    }
}

// POST: api/Teacher
[HttpPost]
[ProducesResponseType(201)]
[ProducesResponseType(500)]
public ActionResult<IEnumerable<CreateTeacherDto>> Post([FromBody] CreateTeacherDto dto)
{
    try
    {
        _addCommandTeacher.Execute(dto);
        return StatusCode(StatusCodes.Status201Created);
    }
    catch (EntityNotFoundException e)
    {
        return UnprocessableEntity(e.Message);
    }
    catch (Exception)
    {
        return StatusCode(StatusCodes.Status500InternalServerError);
    }
}

// PUT: api/Teacher/5
[HttpPut("{id}")]
[ProducesResponseType(204)]
```

```
[ProducesResponseType(500)]
public ActionResult<IEnumerable<CreateTeacherDto>> Put(int id, [FromBody] CreateTeacherDto dto)
{
   dto.Id = id;
   try
   {
      _editCommandTeacher.Execute(dto);
      return NoContent();
   }
   catch (EntityNotFoundException e)
   {
      if (e.Message == "Teacher doesn't exist.")
      {
         return NotFound(e.Message);
      }

      return UnprocessableEntity(e.Message);
   }
   catch (Exception)
   {
      return StatusCode(500, "error");
   }
}

// DELETE: api/ApiWithActions/5
[HttpDelete("{id}")]
[ProducesResponseType(204)]
[ProducesResponseType(404)]
[ProducesResponseType(500)]
```

```
        public ActionResult<IEnumerable<TeacherDto>> Delete(int id)
        {
            try
            {
                _delCommandTeacher.Execute(id);
                return Ok(); //200
            }
            catch
            {
                return NotFound(); //404
            }
        }
    }
}
```

2) EMAIL

```
using ApplicationLayer.Interfaces;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Mail;
using System.Threading.Tasks;

namespace ApiApp.Email
{
    public class SmtpEmailSender : IEmailSender
    {
        private string host;
        private int port;
```

```
private string from;
private string password;

public SmtpEmailSender(string host, int port, string from, string password)
{
    this.host = host;
    this.port = port;
    this.from = from;
    this.password = password;
}

public string ToEmail { get; set; }
public string Body { get; set; }
public string Subject { get; set; }

public void Send()
{
    var smtp = new SmtpClient
    {
        Host = host,
        Port = port,
        EnableSsl = true,
        DeliveryMethod = SmtpDeliveryMethod.Network,
        UseDefaultCredentials = false,
        Credentials = new NetworkCredential(from, password)
    };

    using (var message = new MailMessage(from, /*"uspesnoposlato@gmail.com"*/ ToEmail)
    {
```

```
            Subject = Subject,
            Body = Body
        })
        {
            smtp.Send(message);
        }
    }
  }
}
```

3) HELPERS

4) DEPENDECE_CONTENER (START_APP)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using ApiApp.Email;
using ApplicationLayer.Commands;
using ApplicationLayer.Interfaces;
using EfCommands;
using EfDataAccess;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.HttpsPolicy;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Options;
```

```
using Swashbuckle.AspNetCore.Swagger;

namespace ApiApp
{
   public class Startup
   {
      public Startup(IConfiguration configuration)
      {
         Configuration = configuration;
      }

      public IConfiguration Configuration { get; }

      // This method gets called by the runtime. Use this method to add services to the container.
      public void ConfigureServices(IServiceCollection services)
      {
         services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_2);
         services.AddDbContext<AspProjContext>();
         services.AddTransient<IGetCoursesCommand, EfGetCoursesCommand>();
         services.AddTransient<IGetCourseCommand, EfGetCourseCommand>();
         services.AddTransient<IAddCourseCommand, EfAddCourseCommand>();
         services.AddTransient<IEditCourseCommand, EfEditCourseCommand>();
         services.AddTransient<IDeleteCourseCommand, EfDeleteCourseCommand>();
         services.AddTransient<IGetStudentsCommand, EfGetStudentsCommand>();
         services.AddTransient<IGetStudentCommand, EfGetStudentCommand>();
         services.AddTransient<IAddStudentCommand, EfAddStudentCommand>();
         services.AddTransient<IEditStudentCommand, EfEditStudentCommand>();
         services.AddTransient<IDeleteStudentCommand, EfDeleteStudentCommand>();
         services.AddTransient<IGetTeachersCommand, EfGetTeachersCommand>();
```

```
            services.AddTransient<IGetTeacherCommand, EfGetTeacherCommand>();
            services.AddTransient<IDeleteTeacherCommand, EfDeleteTeacherCommand>();
            services.AddTransient<IAddTeacherCommand, EfAddTeacherCommand>();
            services.AddTransient<IEditTeacherCommand, EfEditTeacherCommand>();



            var section = Configuration.GetSection("Email");

            var sender = new SmtpEmailSender(section["host"], Int32.Parse(section["port"]), section["fromaddress"],
section["password"]);

            services.AddSingleton<IEmailSender>(sender);

            services.AddSwaggerGen(c =>
            {
                c.SwaggerDoc("v1", new Info { Title = "AspProject", Version = "v1" });

            });



        }

        // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
        public void Configure(IApplicationBuilder app, IHostingEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }
            else
```

```
        {
            // The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.
            app.UseHsts();
        }

        app.UseHttpsRedirection();
        app.UseMvc();

        app.UseSwagger();

        // Enable middleware to serve swagger-ui (HTML, JS, CSS, etc.),
        // specifying the Swagger JSON endpoint.
        app.UseSwaggerUI(c =>
        {
            c.SwaggerEndpoint("/swagger/v1/swagger.json", "My API V1");
            //c.RoutePrefix = string.Empty;
        });
    }
  }
}
```

---------WEB APLICATION-----

1)CONTROLLER

A) Student_controller

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

```
using ApplicationLayer.Commands;
using ApplicationLayer.DTO;
using ApplicationLayer.Exceptions;
using ApplicationLayer.SearchQuery;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;

namespace WebApp.Controllers
{
   public class StudentsController : Controller
   {
      private readonly IAddStudentCommand _addCommandStd;
      private readonly IGetStudentsCommand _getCommandStds;
      private readonly IGetStudentCommand _getCommandStd;
      private readonly IEditStudentCommand _editCommandStd;
      private readonly IDeleteStudentCommand _delCommandStd;

      public StudentsController(IAddStudentCommand addCommandStd, IGetStudentsCommand getCommandStds,
IGetStudentCommand getCommandStd, IEditStudentCommand editCommandStd, IDeleteStudentCommand delCommandStd)
      {
         _addCommandStd = addCommandStd;
         _getCommandStds = getCommandStds;
         _getCommandStd = getCommandStd;
         _editCommandStd = editCommandStd;
         _delCommandStd = delCommandStd;
      }

      // GET: Students
      public ActionResult Index(StudentSearchQuery searchQuery)
      {
```

```
    var getStds = _getCommandStds.Execute(searchQuery);
    return View(getStds);
}

// GET: Students/Details/5
public ActionResult Details(int id)
{
    try
    {
        var getStd = _getCommandStd.Execute(id);
        return View(getStd);
    }
    catch(Exception)
    {
        return View();
    }



}

// GET: Students/Create
public ActionResult Create()
{
    return View();
}

// POST: Students/Create
[HttpPost]
```

```
[ValidateAntiForgeryToken]
public ActionResult Create(CreateStudentDto dto)
{
   if (!ModelState.IsValid)
   {
      return View(dto);
   }
   try
   {
      // TODO: Add insert logic here
      _addCommandStd.Execute(dto);
      return RedirectToAction(nameof(Index));
   }
   catch(Exception)
   {
      TempData["error"] = "error.";
   }

   return View();
}

// GET: Students/Edit/5
public ActionResult Edit(int id)
{
   try
   {

      var editStd = _getCommandStd.Execute(id);
      return View(editStd);
```

```
    }
    catch(Exception)
    {
        return RedirectToAction("index");
    }



}

// POST: Students/Edit/5
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Edit(int id,[FromBody] CreateStudentDto dto)
{
    if (!ModelState.IsValid)
    {
        return View(dto);
    }
    try
    {
        // TODO: Add update logic here
        _editCommandStd.Execute(dto);
        return RedirectToAction(nameof(Index));
    }
    catch(EntityNotFoundException)
    {
        return RedirectToAction(nameof(Index));
    }
}
```

```
// GET: Students/Delete/5
public ActionResult Delete(int id)
{
    if (!ModelState.IsValid)
    {
        return View(id);
    }
    try
    {

        _delCommandStd.Execute(id);
        return RedirectToAction(nameof(Index));
    }
    catch (Exception)
    {
        TempData["error"] = "error.";
    }

    return View();
}

// POST: Students/Delete/5
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Delete(int id, IFormCollection collection)
{
    try
    {
```

```
            // TODO: Add delete logic here


            return RedirectToAction(nameof(Index));
        }
        catch
        {
            return View();
        }
    }
  }
}
```

B) Course_controller

2)VIEWS

- COURSES – Find.cshtml

```
@{
  ViewData["Title"] = "Find";
  Layout = "~/Views/Shared/_Layout.cshtml";
}

<!-- Page Container -->
<div class="w3-content w3-margin-top" style="max-width:1400px;">

  <!-- The Grid -->
  <div class="w3-row-padding">

    <!-- Left Column -->
    <div class="w3-third">
```

```
<div class="w3-white w3-text-grey w3-card-4">
  <div class="w3-display-container">
    @*<img src="/w3images/avatar_hat.jpg" style="width:100%" alt="Avatar">*@
    <div class="w3-display-bottomleft w3-container w3-text-black">
      <h2>Jane Doe</h2>
    </div>
  </div>
  <div class="w3-container">
    <p><i class="fa fa-briefcase fa-fw w3-margin-right w3-large w3-text-teal"></i>Designer</p>
    <p><i class="fa fa-home fa-fw w3-margin-right w3-large w3-text-teal"></i>London, UK</p>
    <p><i class="fa fa-envelope fa-fw w3-margin-right w3-large w3-text-teal"></i>ex@mail.com</p>
    <p><i class="fa fa-phone fa-fw w3-margin-right w3-large w3-text-teal"></i>1224435534</p>
    <hr>

    <p class="w3-large"><b><i class="fa fa-asterisk fa-fw w3-margin-right w3-text-teal"></i>Skills</b></p>
    <p>Adobe Photoshop</p>
    <div class="w3-light-grey w3-round-xlarge w3-small">
      <div class="w3-container w3-center w3-round-xlarge w3-teal" style="width:90%">90%</div>
    </div>
    <p>Photography</p>
    <div class="w3-light-grey w3-round-xlarge w3-small">
      <div class="w3-container w3-center w3-round-xlarge w3-teal" style="width:80%">
        <div class="w3-center w3-text-white">80%</div>
      </div>
    </div>
    <p>Illustrator</p>
    <div class="w3-light-grey w3-round-xlarge w3-small">
      <div class="w3-container w3-center w3-round-xlarge w3-teal" style="width:75%">75%</div>
```

```
            </div>
            <p>Media</p>
            <div class="w3-light-grey w3-round-xlarge w3-small">
              <div class="w3-container w3-center w3-round-xlarge w3-teal" style="width:50%">50%</div>
            </div>
            <br>

            <p class="w3-large w3-text-theme"><b><i class="fa fa-globe fa-fw w3-margin-right w3-text-teal"></i>Languages</b></p>
            <p>English</p>
            <div class="w3-light-grey w3-round-xlarge">
              <div class="w3-round-xlarge w3-teal" style="height:24px;width:100%"></div>
            </div>
            <p>Spanish</p>
            <div class="w3-light-grey w3-round-xlarge">
              <div class="w3-round-xlarge w3-teal" style="height:24px;width:55%"></div>
            </div>
            <p>German</p>
            <div class="w3-light-grey w3-round-xlarge">
              <div class="w3-round-xlarge w3-teal" style="height:24px;width:25%"></div>
            </div>
            <br>
          </div>
        </div><br>

        <!-- End Left Column -->
      </div>

      <!-- Right Column -->
      <div class="w3-twothird">
```

```
<div class="w3-container w3-card w3-white w3-margin-bottom">

  <h2 class="w3-text-grey w3-padding-16"><i class="fa fa-suitcase fa-fw w3-margin-right w3-xxlarge w3-text-teal"></i>Work Experience</h2>

  <div class="w3-container">

    <h5 class="w3-opacity"><b>Front End Developer / w3schools.com</b></h5>

    <h6 class="w3-text-teal"><i class="fa fa-calendar fa-fw w3-margin-right"></i>Jan 2015 - <span class="w3-tag w3-teal w3-round">Current</span></h6>

    <p>Lorem ipsum dolor sit amet. Praesentium magnam consectetur vel in deserunt aspernatur est reprehenderit sunt hic. Nulla tempora soluta ea et odio, unde doloremque repellendus iure, iste.</p>

    <hr>

  </div>

  <div class="w3-container">

    <h5 class="w3-opacity"><b>Web Developer / something.com</b></h5>

    <h6 class="w3-text-teal"><i class="fa fa-calendar fa-fw w3-margin-right"></i>Mar 2012 - Dec 2014</h6>

    <p>Consectetur adipisicing elit. Praesentium magnam consectetur vel in deserunt aspernatur est reprehenderit sunt hic. Nulla tempora soluta ea et odio, unde doloremque repellendus iure, iste.</p>

    <hr>

  </div>

  <div class="w3-container">

    <h5 class="w3-opacity"><b>Graphic Designer / designsomething.com</b></h5>

    <h6 class="w3-text-teal"><i class="fa fa-calendar fa-fw w3-margin-right"></i>Jun 2010 - Mar 2012</h6>

    <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. </p><br>

  </div>

</div>


<div class="w3-container w3-card w3-white">

  <h2 class="w3-text-grey w3-padding-16"><i class="fa fa-certificate fa-fw w3-margin-right w3-xxlarge w3-text-teal"></i>Education</h2>

  <div class="w3-container">

    <h5 class="w3-opacity"><b>W3Schools.com</b></h5>

    <h6 class="w3-text-teal"><i class="fa fa-calendar fa-fw w3-margin-right"></i>Forever</h6>
```

```
            <p>Web Development! All I need to know in one place</p>
            <hr>
          </div>
          <div class="w3-container">
            <h5 class="w3-opacity"><b>London Business School</b></h5>
            <h6 class="w3-text-teal"><i class="fa fa-calendar fa-fw w3-margin-right"></i>2013 - 2015</h6>
            <p>Master Degree</p>
            <hr>
          </div>
          <div class="w3-container">
            <h5 class="w3-opacity"><b>School of Coding</b></h5>
            <h6 class="w3-text-teal"><i class="fa fa-calendar fa-fw w3-margin-right"></i>2010 - 2013</h6>
            <p>Bachelor Degree</p><br>
          </div>
        </div>

        <!-- End Right Column -->
      </div>

      <!-- End Grid -->
    </div>

    <!-- End Page Container -->
</div>
```

- Students

--- Create-----

```
@model ApplicationLayer.DTO.StudentDto

@{
```

```
    ViewData["Title"] = "Create";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h1>Create</h1>

<h4>StudentDto</h4>

@if(TempData["error"] != null)
{
    <p class="text-danger">@TempData["error"]</p>
}

<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Create" method="post">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="StudentName" class="control-label"></label>
                <input asp-for="StudentName" class="form-control" />
                <span asp-validation-for="StudentName" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="StudyYear" class="control-label"></label>
                <input asp-for="StudyYear" class="form-control" />
                <span asp-validation-for="StudyYear" class="text-danger"></span>
            </div>
            <div class="form-group">
```

```
            <label asp-for="NumberIndex" class="control-label"></label>
            <input asp-for="NumberIndex" class="form-control" />
            <span asp-validation-for="NumberIndex" class="text-danger"></span>
        </div>
        <div class="form-group">
            <input type="submit" value="Create" class="btn btn-primary" />
        </div>
    </form>
  </div>
</div>

<div>
  <a asp-action="Index">Back to List</a>
</div>
```

-----Delete------

```
@{
  ViewData["Title"] = "Delete";
  Layout = "~/Views/Shared/_Layout.cshtml";
}

<h1>Deleted</h1>
```

------ Details------

```
@model ApplicationLayer.DTO.StudentDto

@{
```

```
    ViewData["Title"] = "Details";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h1>Details</h1>

@if (Model == null)
{
    <p>Object is not exist</p>
}
else
{
<div>
    <h4>StudentDto</h4>
    <hr />
    <dl class="row">
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Id)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Id)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.StudentName)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.StudentName)
        </dd>
        <dt class = "col-sm-2">
```

```
        @Html.DisplayNameFor(model => model.StudyYear)
    </dt>
    <dd class = "col-sm-10">
        @Html.DisplayFor(model => model.StudyYear)
    </dd>
    <dt class = "col-sm-2">
        @Html.DisplayNameFor(model => model.NumberIndex)
    </dt>
    <dd class = "col-sm-10">
        @Html.DisplayFor(model => model.NumberIndex)
    </dd>
  </dl>
</div>
}
<div>
  @Html.ActionLink("Edit", "Edit", new { /* id = Model.PrimaryKey */ }) |
  <a asp-action="Index">Back to List</a>
</div>
```

----- Edit ------

```
@model ApplicationLayer.DTO.StudentDto

@{
  ViewData["Title"] = "Edit";
  Layout = "~/Views/Shared/_Layout.cshtml";
}

<h1>Edit</h1>

<h4>StudentDto</h4>
```

```
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Edit" method="get">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="StudentName" class="control-label"></label>
                <input asp-for="StudentName" class="form-control" />
                <span asp-validation-for="StudentName" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="StudyYear" class="control-label"></label>
                <input asp-for="StudyYear" class="form-control" />
                <span asp-validation-for="StudyYear" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="NumberIndex" class="control-label"></label>
                <input asp-for="NumberIndex" class="form-control" />
                <span asp-validation-for="NumberIndex" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Save" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>

<div>
    <a asp-action="Index">Back to List</a>
```

```
</div>
```

---- Index------

```
@model IEnumerable<ApplicationLayer.DTO.StudentDto>


@{
    ViewData["Title"] = "Index";
    Layout = "~/Views/Shared/_Layout.cshtml";
}


<h1>Index</h1>


<p>
    <a asp-action="Create">Create New</a>
</p>
<table class="table">
    <thead>
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.Id)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.StudentName)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.StudyYear)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.NumberIndex)
            </th>
```

```
        <th></th>
      </tr>
    </thead>
    <tbody>
@foreach (var item in Model) {
      <tr>
        <td>
          @Html.DisplayFor(modelItem => item.Id)
        </td>
        <td>
          @Html.DisplayFor(modelItem => item.StudentName)
        </td>
        <td>
          @Html.DisplayFor(modelItem => item.StudyYear)
        </td>
        <td>
          @Html.DisplayFor(modelItem => item.NumberIndex)
        </td>
        <td>
          @Html.ActionLink("Edit", "Edit", new { /* id=item.PrimaryKey */ }) |
          @Html.ActionLink("Details", "Details", new { /* id=item.PrimaryKey */ }) |
          @Html.ActionLink("Delete", "Delete", new { /* id=item.PrimaryKey */ })
        </td>
      </tr>
}
    </tbody>
</table>
```

----- STARTUP -----

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using ApplicationLayer.Commands;
using EfCommands;
using EfDataAccess;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.HttpsPolicy;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;

namespace WebApp
{
   public class Startup
   {
      public Startup(IConfiguration configuration)
      {
         Configuration = configuration;
      }

      public IConfiguration Configuration { get; }

      // This method gets called by the runtime. Use this method to add services to the container.
```

```
public void ConfigureServices(IServiceCollection services)
{
    services.Configure<CookiePolicyOptions>(options =>
    {
        // This lambda determines whether user consent for non-essential cookies is needed for a given request.
        options.CheckConsentNeeded = context => true;
        options.MinimumSameSitePolicy = SameSiteMode.None;
    });


    services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_2);
    services.AddDbContext<AspProjContext>();
    services.AddTransient<IAddStudentCommand, EfAddStudentCommand>();
    services.AddTransient<IGetStudentsCommand, EfGetStudentsCommand>();
    services.AddTransient<IGetStudentCommand, EfGetStudentCommand>();
    services.AddTransient<IEditStudentCommand, EfEditStudentCommand>();
    services.AddTransient<IDeleteStudentCommand, EfDeleteStudentCommand>();

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    else
    {
```

```
            app.UseExceptionHandler("/Home/Error");
            // The default HSTS value is 30 days. You may want to change this for production scenarios, see
https://aka.ms/aspnetcore-hsts.
            app.UseHsts();
        }

        app.UseHttpsRedirection();
        app.UseStaticFiles();
        app.UseCookiePolicy();

        app.UseMvc(routes =>
        {
            routes.MapRoute(
                name: "default",
                template: "{controller=Home}/{action=Index}/{id?}");
        });
    }
  }
}
```

FUNKCIONALNOST PROJEKTA SE ZASNIVA NA CRUID PRINCIPIMA (SELECT * 2, INSERT, UPDATE, DELETE)

IMPLEMENTACIJA OVIH PRINCIPA JE IMPLEMENTIRANA U -----EFCOMMAND ----

---SELECT --- SVIH

```
using ApplicationLayer.Commands;
using ApplicationLayer.DTO;
using ApplicationLayer.SearchQuery;
```

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

namespace EfCommands
{
   public class EfGetStudentsCommand : BaseEfCommand, IGetStudentsCommand
   {

      public EfGetStudentsCommand(AspProjContext context) : base(context)
      {

      }

      public IEnumerable<StudentDto> Execute(StudentSearchQuery request)
      {
         var getStudents = _context.Students.AsQueryable();

         if(request.Keyword != null)
         {
            getStudents = getStudents.Where(std => std.StudentName
            .ToLower()
            .Contains(request.Keyword.ToLower()));
         }

         if (request.OnlyActive.HasValue)
         {
```

```
            getStudents = getStudents.Where(std => std.IsDeleted != request.OnlyActive);
        }

        return getStudents.Select(std => new StudentDto
        {
            Id = std.Id,
            StudentName = std.StudentName,
            StudyYear = std.StudyYear,
            NumberIndex = std.NumberIndex
        });
    }
  }
}
```

---- SELECT --- JEDNOG

```
using ApplicationLayer.Commands;
using ApplicationLayer.DTO;
using ApplicationLayer.Exceptions;
using EfDataAccess;
using System;
using System.Collections.Generic;
using System.Text;

namespace EfCommands
{
  public class EfGetStudentCommand : BaseEfCommand, IGetStudentCommand
  {
    public EfGetStudentCommand(AspProjContext context) : base(context)
    {
```

```
    }
    public StudentDto Execute(int request)
    {
        var getStd = _context.Students.Find(request);

        if (getStd == null)
            throw new EntityNotFoundException();

        return new StudentDto
        {
            Id = getStd.Id,
            StudentName = getStd.StudentName,
            StudyYear = getStd.StudyYear,
            NumberIndex = getStd.NumberIndex
        };
    }
  }
}
```

----- INSERT ---

```
using ApplicationLayer.Commands;
using ApplicationLayer.DTO;
using ApplicationLayer.Exceptions;
using EfDataAccess;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Runtime.Serialization;
using DomainLayer;
```

```
namespace EfCommands
{
   public class EfAddStudentCommand : BaseEfCommand, IAddStudentCommand
   {
      public EfAddStudentCommand (AspProjContext context) : base(context)
      {

      }
      public void Execute(CreateStudentDto request)
      {
         if (_context.Students.Any(std => std.StudentName == request.StudentName && std.NumberIndex ==
request.NumberIndex && std.StudyYear > 12))
         {
            throw new EntityNotFoundException();
         }

         if (!_context.Standards.Any(s => s.Id == request.StandardId))
         {
            throw new EntityNotFoundException("Standard");
            // Message -> Student doesn't exist
         }

         _context.Students.Add(new Student
         {
            //Id = request.Id,
            StudentName = request.StudentName,
            StudyYear = request.StudyYear,
            NumberIndex = request.NumberIndex,
            NumberPhone = request.NumberPhone,
```

```
            Natioanality = request.Natioanality,
            BirthDate = request.BirthDate,
            CreatedAt = DateTime.Now,
            StandardId = request.StandardId,
        });

        _context.SaveChanges();
    }
  }
}
```

------ UPDATE ----

```
using ApplicationLayer.Commands;
using ApplicationLayer.DTO;
using ApplicationLayer.Exceptions;
using EfDataAccess;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace EfCommands
{
  public class EfEditStudentCommand : BaseEfCommand, IEditStudentCommand
  {
    public EfEditStudentCommand(AspProjContext context) : base(context)
    {

    }

    public void Execute(CreateStudentDto request)
```

```
    {
        var editStd = _context.Students.Find(request.Id);


        if (editStd == null)
            throw new EntityNotFoundException();


        if(editStd.StudentName != request.StudentName)
        {
            if(_context.Students.Any(s => s.StudentName == request.StudentName/* && s.NumberIndex ==
request.NumberIndex*/))
            {
                throw new EntityNotFoundException();
            }


            editStd.StudentName = request.StudentName;
            editStd.NumberIndex = request.NumberIndex;
            editStd.StudyYear = request.StudyYear;
            editStd.NumberPhone = request.NumberPhone;
            editStd.Natioanality = request.Natioanality;
            editStd.BirthDate = request.BirthDate;
            editStd.StandardId = request.StandardId;




            _context.SaveChanges();
        }
    }
  }
}
```

---- DELETE ---

```
using ApplicationLayer.Commands;
```

```
using ApplicationLayer.Exceptions;
using EfDataAccess;
using System;
using System.Collections.Generic;
using System.Text;

namespace EfCommands
{
   public class EfDeleteStudentCommand : BaseEfCommand, IDeleteStudentCommand
   {
      public EfDeleteStudentCommand(AspProjContext context) : base(context)
      {

      }
      public void Execute(int request)
      {
         var delStd = _context.Students.Find(request);

         if (delStd == null)
            throw new EntityNotFoundException("Student");

         _context.Students.Remove(delStd);

         _context.SaveChanges();
      }
   }
}
```

BAZA PODATAKA – DIZAJN --

---BASE_ENTITY---

```
using System;

using System.Collections.Generic;

using System.Text;


namespace DomainLayer

{

   public abstract class BaseEntity

   {

      public int Id { get; set; }

      public DateTime CreatedAt { get; set; }

      public DateTime? ModifiedAt { get; set; }

      public bool IsDeleted { get; set; }

   }

}
```

--- COURSE---

```
using System;

using System.Collections.Generic;

using System.Text;


namespace DomainLayer

{

   public class Course : BaseEntity
```

```
    {
        public string CourseName { get; set; }
        public string Description { get; set; }
        public string Location { get; set; }



        public int TeacherId { get; set; } // foriegn key


        public Teacher Teacher { get; set; }  // ref
        public ICollection<StudentCourse> CourseStudents { get; set; }
    }
}
```

----STANDARD--

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

namespace DomainLayer
{
    public class Standard : BaseEntity
    {
        public string StandardName { get; set; }
        public string Description { get; set; }


        public ICollection<Teacher> Teachers { get; set; }
        public ICollection<Student> Students { get; set; }
    }-
}
```

-----STUDENT-----

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

namespace DomainLayer
{
   public class Student : BaseEntity
   {
      public string StudentName { get; set; }
      public int StudyYear { get; set; }
      public int NumberIndex { get; set; }
      public int NumberPhone { get; set; }
      public string Natioanality { get; set; }
      public int BirthDate { get; set; }

      public int StandardId { get; set; }  // forign key

      public Standard StandardStudent { get; set; }  // ref
      public StudentAddress StudentAddress { get; set; }
      public ICollection<StudentCourse> StudentCourses { get; set; }
   }
}
```

-----STUDENT_ADDRESS---

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

namespace DomainLayer
```

```
{
   public class StudentAddress : BaseEntity
   {
      public string Address { get; set; }
      public string City { get; set; }
      public bool State { get; set; }

      public int StudentId { get; set; }  // for key , must be unique
                                 // One-to-One relationships with Student Entity

      public Student Student { get; set; }  // ref
   }
}
```

---STUDENT_COURSE--

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

namespace DomainLayer
{
   public class StudentCourse
   {
      public int StudentId { get; set; }
      public int CourseId { get; set; }

      public Student Student { get; set; }
      public Course Course { get; set; }
   }
}
```

---TEACHER--

```
using System;

using System.Collections.Generic;

using System.Text;


namespace DomainLayer

{

   public class Teacher : BaseEntity

   {

      public string TFirstName { get; set; }

      public string TLastName { get; set; }

      public string Description { get; set; }

      public string Nationality { get; set; }


      public int StandardId { get; set; }


      public Standard StandardTeacher { get; set; }

      public ICollection<Course> Courses { get; set; }


   }

}
```