School Management with Clean Architecture

1. This project is divided into five layers
   * Data
   * Infrastructure
   * Services
   * Core
   * API
2. Each layer contains
   * Data
     + Router
     + Command
     + Entities
     + Enums
   * Infrastructure
     + Data
       - Configurations
       - DB Context
     + Generic Repository
     + Interface
     + Migrations
     + Persistence
       - Unit Of Work
     + Repository
   * Services
     + Interface
     + Services
   * Core
     + Base Response
       - Generic Base Response
       - Generic Base Response Handler
     + Behavior
       - Validation Behavior
     + Features
       - Commands
       - Queries
     + Localization
     + Mapping
     + Middlewares
     + Pagination
   * API
     + Controllers
     + Appsettings
     + Program
3. Packages nugget
   * Data
     + Microsoft.AspNetCore.Identity.EntityFrameworkCore
   * Infrastructure
     + Microsoft.AspNetCore.Authentication.JwtBearer
     + Microsoft.AspNetCore.Identity.EntityFrameworkCore
     + Microsoft.EntityFrameworkCore
     + Microsoft.EntityFrameworkCore.Design
     + Microsoft.EntityFrameworkCore.Relational
     + Microsoft.EntityFrameworkCore.SqlServer
     + Microsoft.EntityFrameworkCore.Tools
   * Core
     + AutoMapper.Extensions.Microsoft.DependencyInjection
     + FluentValidation.AspNetCore
     + FluentValidation.DependencyInjectionExtensions
     + MediatR
   * API
     + Microsoft.EntityFrameworkCore
     + Microsoft.EntityFrameworkCore.Design
     + Microsoft.EntityFrameworkCore.Relational
4. Commands Contains:
   * Handlers
   * Models
   * Validations
5. Queries Contains:
   * Handlers
   * Models
   * Results
6. Use Lib
   * Configuration of Mediator

## services.AddMediatR(cfg => cfg.RegisterServicesFromAssemblies(Assembly.GetExecutingAssembly()));

* + Configuration of Automapper

## services.AddAutoMapper(Assembly.GetExecutingAssembly());

* + Add Fluent Validation

## services.AddValidatorsFromAssembly(Assembly.GetExecutingAssembly());

1. Generic Repository

public interface IGenericRepositoryAsync<T> where T : class

{

    IDbContextTransaction BeginTransaction();

    void Commit();

    void RollBack();

    IQueryable<T> GetTableNoTracking();

    IQueryable<T> GetTableAsTracking();

    Task<T> GetByIdAsync(int id);

    Task<T> AddAsync(T entity);

    Task AddRangeAsync(ICollection<T> entities);

    Task UpdateAsync(T entity);

    Task UpdateRangeAsync(ICollection<T> entities);

    Task DeleteAsync(T entity);

    Task DeleteRangeAsync(ICollection<T> entities);

    Task SaveChangesAsync();

}

public class GenericRepositoryAsync<T> : IGenericRepositoryAsync<T>   
where T : class

{

    #region Vars / Props

    protected readonly ApplicationDBContext \_dbContext;

    #endregion

    #region Constructor(s)

    public GenericRepositoryAsync(ApplicationDBContext dbContext)

    {

        \_dbContext = dbContext;

    }

    #endregion

    #region Methods

    #endregion

    #region Actions

    public virtual async Task<T> GetByIdAsync(int id)

    {

        return await \_dbContext.Set<T>().FindAsync(id);

    }

1. When Create Endpoint Item
   * Request
     + Name
     + Address
     + Phone
     + DID
   * Response
     + Added Successfully
   * Validations
     + Name is Required
     + Address id Required
   * Conditions
     + Check The Item Already Exit or Not.
     + Check If Item Equal Null
     + Add Item