Laboratorium 1.

Platforma międzynarodowa ofert pracy.

-Możliwość dodawania/edycji/usuwania/wyświetlania ofert pracy dla różnych krajów

- Historia zmian Oferty pracy.

- Lista krajów

-Lista stanowisk

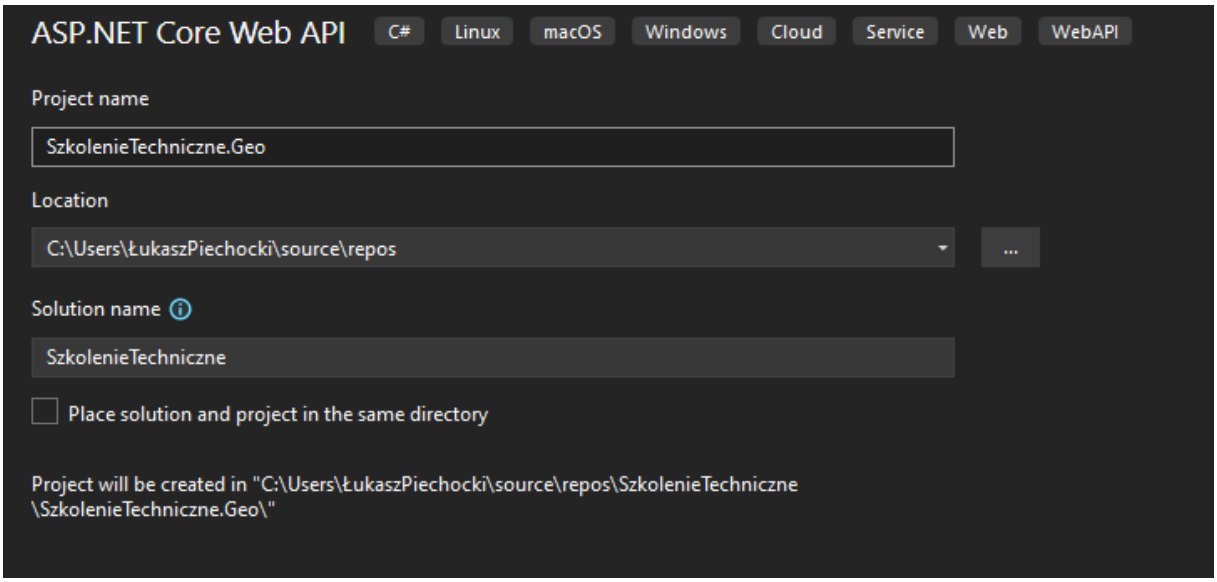
- Lista firm wraz z stanowiskami.

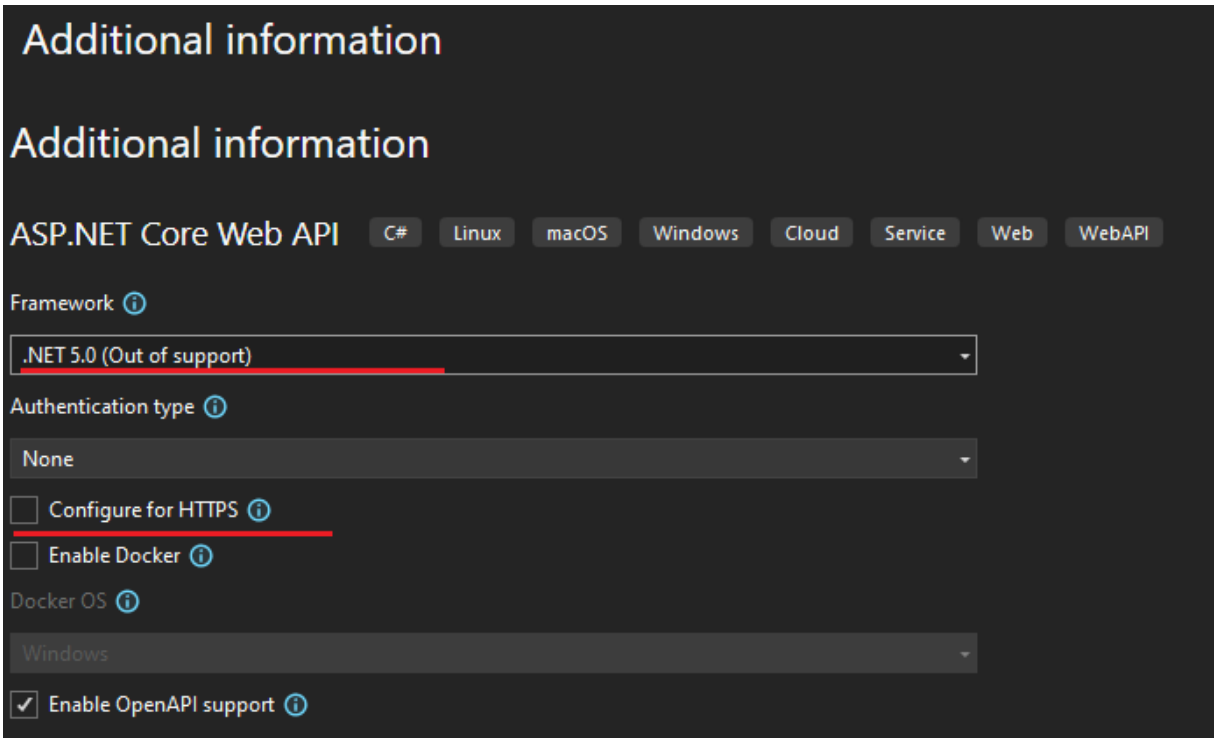
Zaczynamy:

Tworzymy nowy projekt

Typ projektu: **WebApi**

Nazwa projektu: SzkolenieTechniczne.Geo



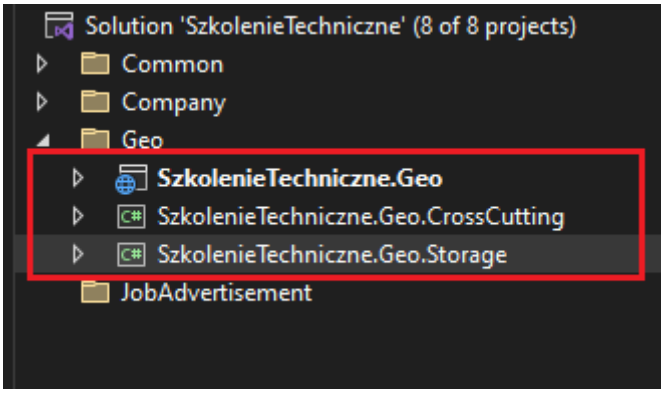


Dodanie projektów

Typ projektu: Biblioteka klas:

Nazwy: SzkolenieTechniczne.Geo.CrossCutting ,

SzkolenieTechniczne.Geo.Storage

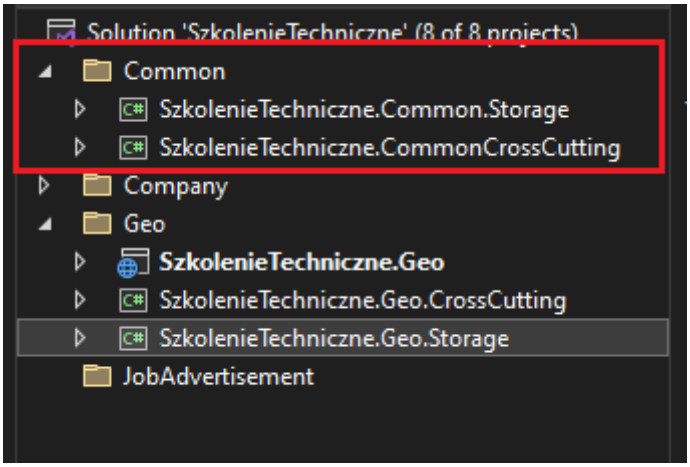


Przeniesienie ich do Katalogu: Geo.

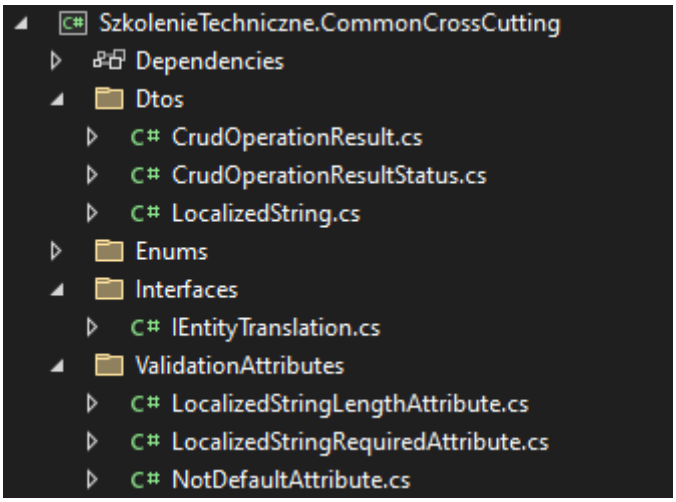
Dodanie nowego Katalogu Common.

Dodaniu w nich projektów typu biblioteka klas.

Nazwy: SzkolenieTechniczne.CommonCrossCutting, SzkolenieTechniczne.Common.Storage



Implementacja w projekcie SzkolenieTechniczne.CommonCrossCutting



Interface IEntityTranslation:

```
public interface IEntityTranslation
{
    public string LanguageCode { get; set; }
}
```

Katalog Dtos klasy:

```
public enum CrudOperationResultStatus
{
    Success,
    Failure,
    RecordNotFound
}
```

```
namespace SzkolenieTechniczne.CommonCrossCutting.Dtos
{
    public class CrudOperationResult<TDto>
    {
        public CrudOperationResultStatus Status { get; set; }

        public TDto? Result { get; set; }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SzkolenieTechniczne.CommonCrossCutting.Dtos
{
    public class LocalizedString : Dictionary<string, string>, IValidatableObject
    {
        public LocalizedString()
        {
        }

        public LocalizedString(IEnumerable<KeyValuePair<string, string>> items) : base(items)
        {
        }

        public IEnumerable<ValidationResult> Validate(ValidationContext validationContext)
        {
            if (this.Keys.Any(string.IsNullOrWhiteSpace))
            {
                yield return new ValidationResult("Language keys cannot be null nor white space");
            }
        }
    }
}
```

Katalog Enums klasy:

```
namespace SzkolenieTechniczne.CommonCrossCutting.Enums
{
    public enum LocalizedStringRequirementType
    {
        AtLeastOneLanguage,
        UISupportedLanguages
    }
}
```

Katalog ValidationAttributes klasy:

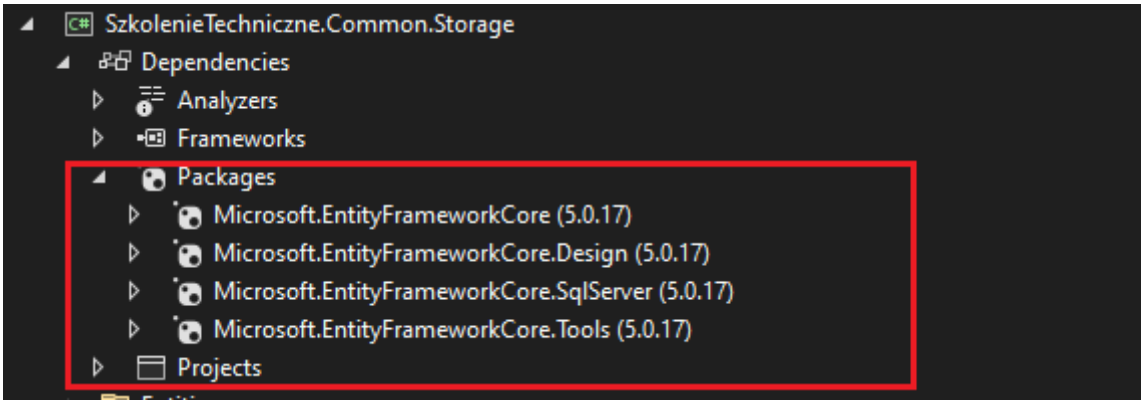
```
using System.ComponentModel.DataAnnotations;
using System.Linq;
using SzkolenieTechniczne.CommonCrossCutting.Dtos;

namespace SzkolenieTechniczne.CommonCrossCutting.ValidationAttributes
{
    public class LocalizedStringLengthAttribute : StringLengthAttribute
    {

        public LocalizedStringLengthAttribute(int maximumLength) : base(maximumLength)
        {
        }

        public override bool IsValid(object? value)
        {
            if (value == null)
            {
                return true;
            }

            if (value is LocalizedString localizedString)
            {
                return localizedString.Values.All(languageText => base.IsValid(languageText));
            }

            return false;
        }
    }
}
```

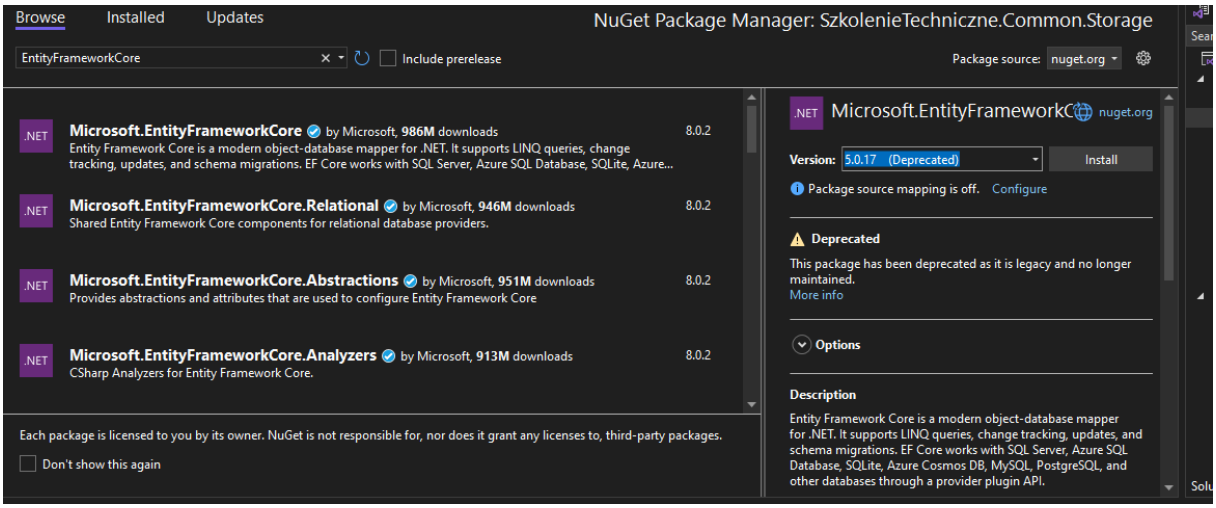
```
using System;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using SzkolenieTechniczne.CommonCrossCutting.Dtos;
using SzkolenieTechniczne.CommonCrossCutting.Enums;

namespace SzkolenieTechniczne.CommonCrossCutting.ValidationAttributes
{
    public class LocalizedStringRequiredAttribute : RequiredAttribute
    {
        private readonly string[] _uiSupportedLanguages = new string[2] { "en", "pl" };

        private readonly LocalizedStringRequirementType _requirementType;

        public LocalizedStringRequiredAttribute(LocalizedStringRequirementType requirementType = LocalizedStringRequirementType.AtLeastOneLanguage)
        {
            _requirementType = requirementType;
        }

        public override bool IsValid(object? value)
        {
            LocalizedString localizedString = value as LocalizedString;
            if (localizedString != null)
            {
                if (_requirementType == LocalizedStringRequirementType.AtLeastOneLanguage)
                {
                    if (localizedString.Values.All(new Func<string, bool>(base.IsValid)))
                    {
                        return localizedString.Values.Count > 0;
                    }

                    return false;
                }

                if (localizedString.Values.All(new Func<string, bool>(base.IsValid)))
                {
                    return _uiSupportedLanguages.All(new Func<string, bool>(localizedString.ContainsKey));
                }

                return false;
            }

            return false;
        }
    }
}
```

```
using System;
using System.ComponentModel.DataAnnotations;

namespace SzkolenieTechniczne.CommonCrossCutting.ValidationAttributes
{
    [AttributeUsage(AttributeTargets.Parameter | AttributeTargets.Property, AllowMultiple = false)]
    public class NotDefaultAttribute : ValidationAttribute
    {
        public const string DefaultErrorMessage = "The {0} field must not have the default value";
        public NotDefaultAttribute() : base(DefaultErrorMessage) { }

        protected override ValidationResult IsValid(object? value, ValidationContext validationContext)
        {
            if (value is null)
            {
                return ValidationResult.Success!;
            }

            var type = value.GetType();
            if (type.IsValueType)
            {
                var defaultValue = Activator.CreateInstance(type);
                return !value.Equals(defaultValue)
                       ? ValidationResult.Success!
                       : new ValidationResult("VALUE_IS_REQUIRED"); ;
            }

            return ValidationResult.Success!;
        }
    }
}
```

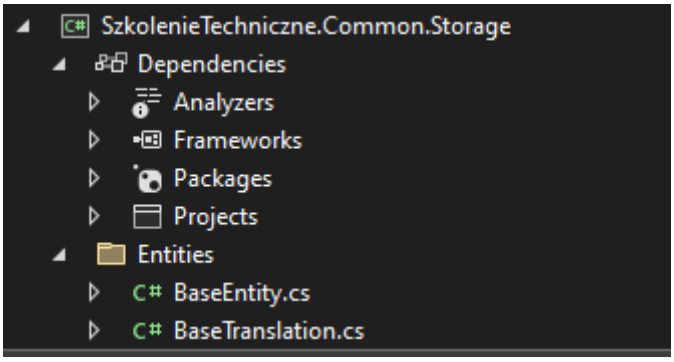
Implementacja w projekcie: SzkolenieTechniczne.Common.Storage

Dodanie pakietów nuget: w wersji 5.0.17





Dodanie katalogu Entities.

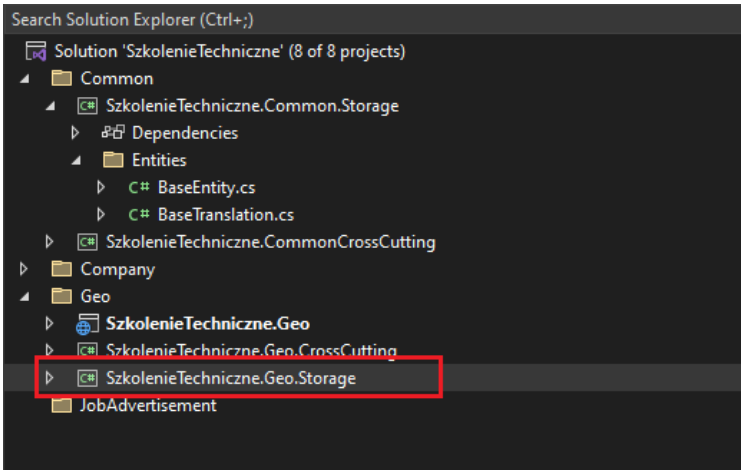


Dodanie klas BaseEntity, BaseTranslation

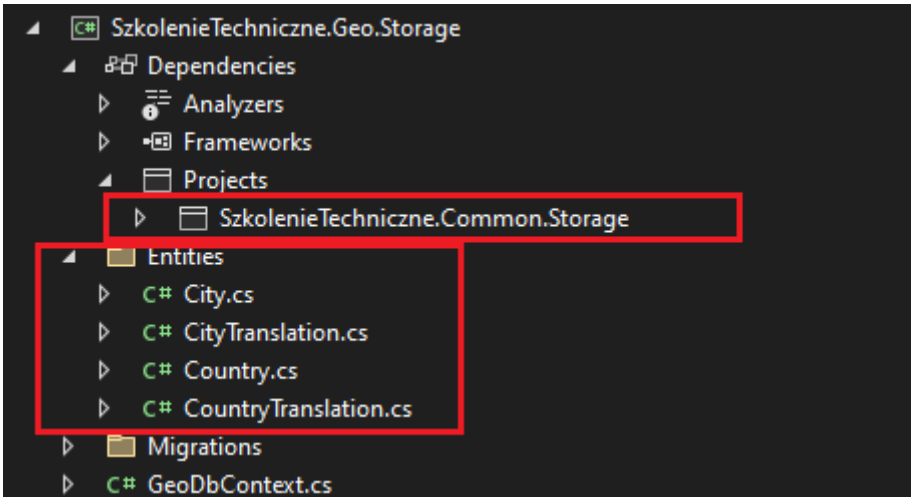
```
using System;
using System.ComponentModel.DataAnnotations;

namespace SzkolenieTechniczne.Common.Storage.Entities
{
    public abstract class  BaseEntity
    {
        [Key]
        public Guid Id { get; set; }
    }
}
```

```
using Microsoft.EntityFrameworkCore;
using System;
using System.ComponentModel.DataAnnotations;
using SzkolenieTechniczne.CommonCrossCutting.Interfaces;

namespace SzkolenieTechniczne.Common.Storage.Entities
{
    [Index(nameof(LanguageCode), IsUnique = false)]
    public class BaseTranslation : BaseEntity, IEntityTranslation
    {
        [MaxLength(16)]
        [Required]
        public string LanguageCode { get; set; } = null!;
    }
}
```

Implementacja w projekcie SzkolenieTechniczne.Geo.Storage



Dodanie referencji do projektu Common.Storage.



Dodajemy encje:

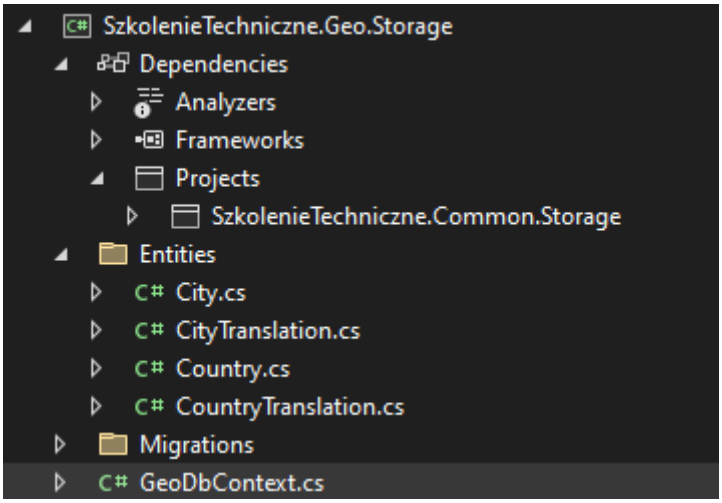
```
using Microsoft.EntityFrameworkCore;
using System.ComponentModel.DataAnnotations;
using System;
using System.ComponentModel.DataAnnotations.Schema;
using SzkolenieTechniczne.Common.Storage.Entities;

namespace SzkolenieTechniczne.Geo.Storage.Entities
{
    [Index(nameof(Name), IsUnique = false)]
    [Table("CityTranslations", Schema = "Geo")]
    public class CityTranslation : BaseTranslation
    {
        [ForeignKey("City")]
        public Guid CityId { get; set; }

        [Required]
        [MaxLength(255)]
        public string Name { get; set; }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using SzkolenieTechniczne.Common.Storage.Entities;

namespace SzkolenieTechniczne.Geo.Storage.Entities
{
    [Table("Cities", Schema = "Geo")]
    public class City : BaseEntity
    {
        [Required]
        public Guid CountryId { get; set; }

        public Country Country { get; set; }

        public ICollection<CityTranslation> Translations { get; set; }
    }
}
```

```
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using SzkolenieTechniczne.Common.Storage.Entities;

namespace SzkolenieTechniczne.Geo.Storage.Entities
{
    [Index(nameof(Name), IsUnique = false)]
    [Table("CountryTranslations", Schema = "Geo")]
    public class CountryTranslation : BaseTranslation
    {
        [ForeignKey("Country")]
        public Guid CountryId { get; set; }

        [MaxLength(64)]
        [MinLength(2)]
        [Required]
        public string Name { get; set; }
    }
}
```

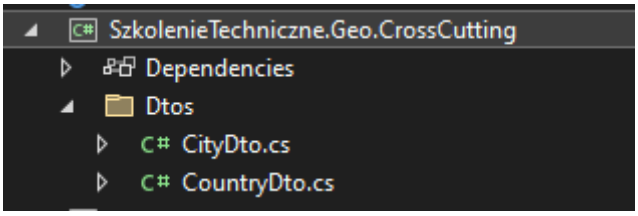
```
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using SzkolenieTechniczne.Common.Storage.Entities;

namespace SzkolenieTechniczne.Geo.Storage.Entities
{
    [Index(nameof(Alpha3Code), IsUnique = true)]
    [Table("Countries", Schema = "Geo")]
    public class Country : BaseEntity
    {
        [MaxLength(3)]
        [MinLength(2)]
        [Required]
        public string Alpha3Code { get; set; }

        public virtual ICollection<CountryTranslation> Translations { get; set; }

        public ICollection<City> Cities { get; set; } = null!;
    }
}
```

Dodanie DbContext dla Geo.



```
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using SzkolenieTechniczne.Geo.Storage.Entities;

namespace SzkolenieTechniczne.Geo.Storage
{
    public class GeoDbContext : DbContext
    {
        private IConfiguration _configuration { get; }

        public DbSet<Country> Countries { get; set; }
        public DbSet<CountryTranslation> CountryTranslations { get; set; }
        public DbSet<City> Cities { get; set; }
        public DbSet<CityTranslation> CityTranslations { get; set; }

        public GeoDbContext(IConfiguration configuration)
          : base()
        {
            _configuration = configuration;
        }

        protected override void OnConfiguring(DbContextOptionsBuilder options)
        {
            options.UseSqlServer(@"server=(localdb)\MSSQLLocalDB;database=geo-dev;trusted_connection=true;",
          //  options.UseSqlServer("",
                x => x.MigrationsHistoryTable("__EFMigrationsHistory", "Geo"));
        }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            base.OnModelCreating(modelBuilder);
        }
    }
}
```

Implementacja w projekcie SzkolenieTechniczne.Geo.CrossCutting

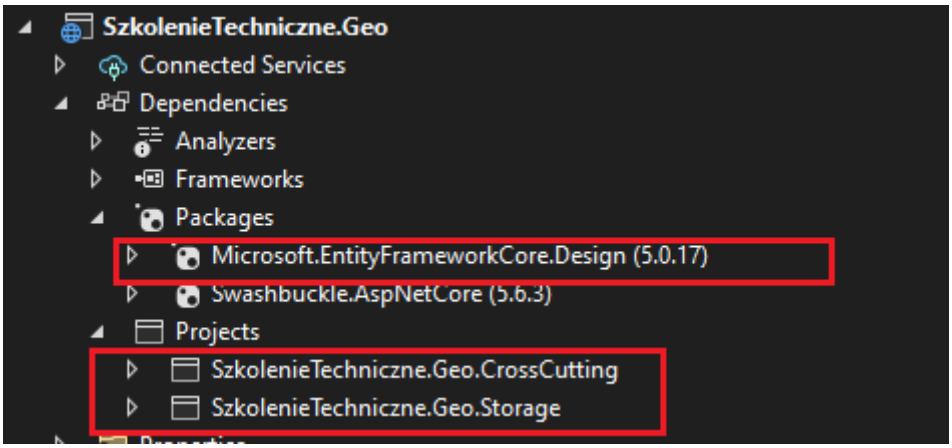


```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using SzkolenieTechniczne.CommonCrossCutting.Dtos;
using SzkolenieTechniczne.CommonCrossCutting.ValidationAttribut

namespace SzkolenieTechniczne.Geo.CrossCutting.Dtos
{
    public class CountryDto
    {
        public Guid Id { get; set; }

        [LocalizedStringRequiredAttribute]
        [LocalizedStringLengthAttribute(32)]
        public LocalizedString Name { get; set; }

        [MaxLength(3)]
        [MinLength(2)]
        [Required]
        public string Alpha3Code { get; set; }
    }
}
```

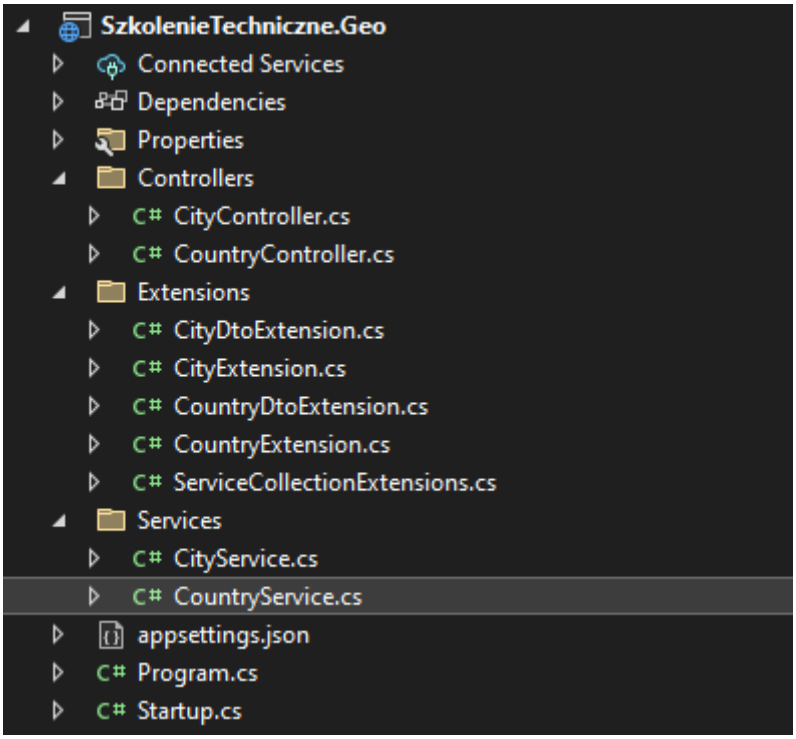
Klasa `CityDto` do własnej implementacji.

Implementacja w projekcie **SzkolenieTechniczne.Geo**

Dodanie pakietu nuget oraz referencji do projektów:



Struktura katalogów:

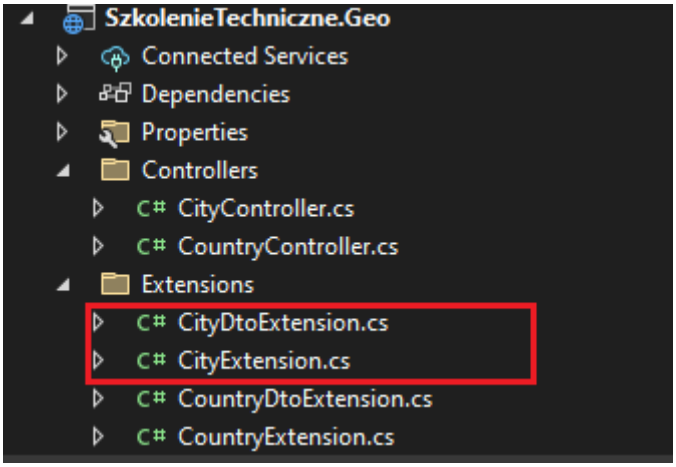


Implementacja w katalogu Extensions:

```
using System.Collections.Generic;
using System.Linq;
using SzkolenieTechniczne.CommonCrossCutting.Dtos;
using SzkolenieTechniczne.Geo.CrossCutting.Dtos;
using SzkolenieTechniczne.Geo.Storage.Entities;

namespace SzkolenieTechniczne.Geo.Extensions
{
    public static class CountryExtension
    {
        public static CountryDto ToDto(this Country entity)
        {
            var result = new CountryDto
            {
                Id = entity.Id,
                Name = new LocalizedString(entity.Translations.Select(t => new KeyValuePair<string, string>(t.LanguageCode, t.Name))),
                Alpha3Code= entity.Alpha3Code
            };
            return result;
        }
    }
}
```

```
using SzkolenieTechniczne.Geo.CrossCutting.Dtos;
using SzkolenieTechniczne.Geo.Storage.Entities;

namespace SzkolenieTechniczne.Geo.Extensions
{
    public static class CountryDtoExtension
    {
        public static Country ToEntity(this CountryDto dto)
        {
            return new Country
            {
                Id = dto.Id,
                Alpha3Code = dto.Alpha3Code
            };
        }
    }
}
```

Sami implementujemy analogicznie dla kasy City.

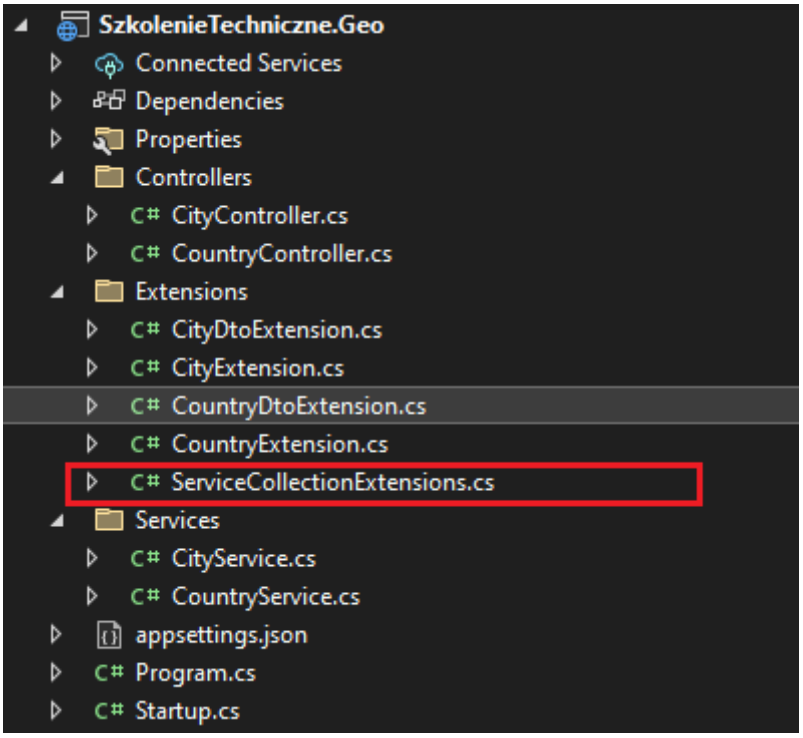


Implementacja w katalogu Services:

```
using System;
using SzkolenieTechniczne.CommonCrossCutting.Dtos;
using SzkolenieTechniczne.Geo.CrossCutting.Dtos;
using SzkolenieTechniczne.Geo.Storage.Entities;
using SzkolenieTechniczne.Geo.Storage;
using Microsoft.EntityFrameworkCore;
using System.Linq;
using SzkolenieTechniczne.Geo.Extensions;

namespace SzkolenieTechniczne.Geo.Services
{
    public class CountryService
    {
        private GeoDbContext _geoDbContext;

        public CountryService(GeoDbContext geoDbContext)
        {
            _geoDbContext = geoDbContext;
        }

        public async Task<CountryDto> GetById(Guid id)
        {
            var country = await _geoDbContext
                .Set<Country>()
                .Include(x=>x.Translations)
                .AsNoTracking()
                .Where(e => e.Id!.Equals(id))
                .SingleOrDefaultAsync();

            return country.ToDto();
        }

        public async Task<IEnumerable<CountryDto>> Get()
        {
            var cities = await _geoDbContext
                .Set<Country>()
                .Include(x => x.Translations)
                .AsNoTracking()
                .Select(e => e.ToDto())
                .ToListAsync();

            return cities;
        }

        public async Task<CrudOperationResult<CountryDto>> Create(CountryDto dto)
        {
            var entity = dto.ToEntity();

            _geoDbContext
                .Set<Country>()
                .Add(entity);

            await _geoDbContext.SaveChangesAsync();

            var newDto = await GetById(entity.Id);

            return new CrudOperationResult<CountryDto>
            {
                Result = newDto,
                Status = CrudOperationResultStatus.Success
            };
        }
```

Sami implementujemy analogicznie dla kasy CityService.

Dodajemy klasę ServiceCollectionExtensions w katalogu Extensions

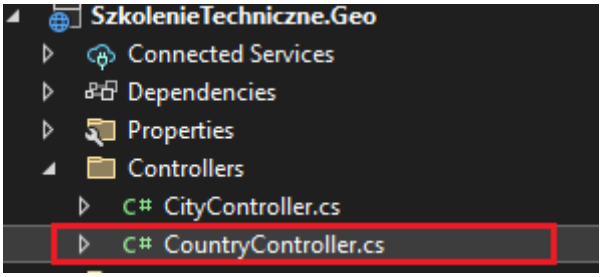



```
public static class ServiceCollectionExtensions
{
    public static IServiceCollection AddGeoServices(this IServiceCollection serviceCollection)
    {
        serviceCollection.AddTransient<CountryService>();
        serviceCollection.AddTransient<CityService>();
        serviceCollection.AddDbContext<GeoDbContext, GeoDbContext>();
        return serviceCollection;
    }
}
```

Dodanie wywołania w klasie startup:

```
namespace SzkolenieTechniczne.Geo
{
    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.AddGeoServices();
            services.AddControllers();
            services.AddSwaggerGen(c =>
            {
                c.SwaggerDoc("v1", new OpenApiInfo { Title = "SzkolenieTechniczne.Geo", Version = "v1" });
            });
        }

        // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
                app.UseSwagger();
                app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.json", "SzkolenieTechniczne.Geo v1"));
            }

            app.UseRouting();

            app.UseAuthorization();

            app.UseEndpoints(endpoints =>
            {
                endpoints.MapControllers();
            });
        }
    }
}
```

Dodajemy nowy Controller o nazwie CountryController



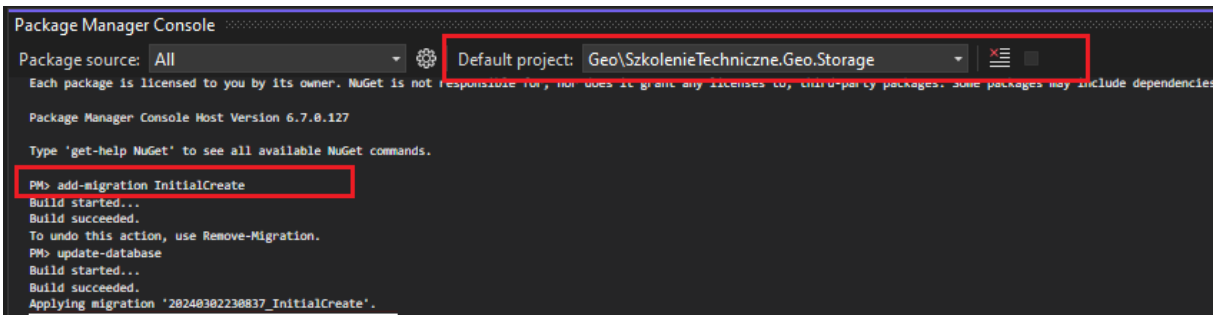
```
[Route("/geo")]
public class CountryController : ControllerBase
{
    private readonly CountryService _countryService;

    public CountryController(CountryService countryService)
    {
        _countryService = countryService;
    }

    /// <summary>
    /// Gets list of countries
    /// </summary>
    /// <returns></returns>
    [HttpGet("countries")]
    public async Task<IEnumerable<CountryDto>> Read() => await _countryService.Get();

    /// <summary>
    /// Gets country information by identifier
    /// </summary>
    /// <param name="id">Identifier of the country</param>
    /// <returns></returns>
    [HttpGet("countries/{id}")]
    public async Task<IActionResult> ReadById(Guid id)
    {
        var cityDto = await _countryService.GetById(id);

        if (cityDto == null)
        {
            return NotFound();
        }

        return Ok(cityDto);
    }

    /// <summary>
    /// Creates a new contry. The identifier of the record will be automatically generated.
    /// </summary>
    /// <param name="dto">Data transfer object describing city</param>
    /// <returns></returns>
    [HttpPost("country")]
    public async Task<IActionResult> Create([FromBody] CountryDto dto)
    {
        if (!ModelState.IsValid)
        {
            return BadRequest(ModelState);
        }

        var operationResult = await _countryService.Create(dto);

        return Ok(operationResult.Result);
    }
}
```

Sami implementujemy analogicznie dla kasy CityController.

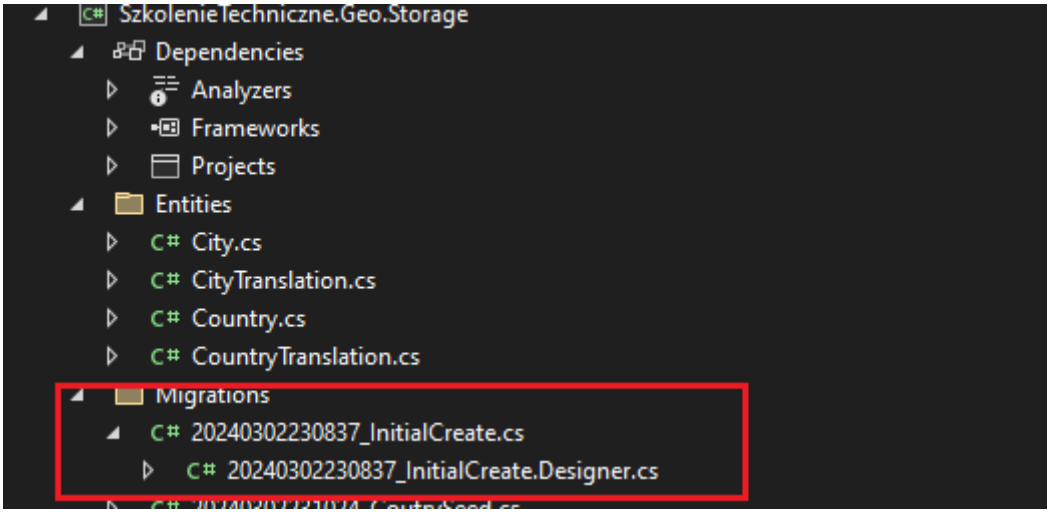
Przechodzimy teraz do konsoli **Package Manager Console** i tworzymy nową migrację

Ustawiamy Default project

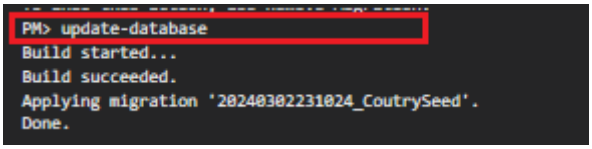
Oraz wykonujemy komendę: `add-migration InitialCreate`



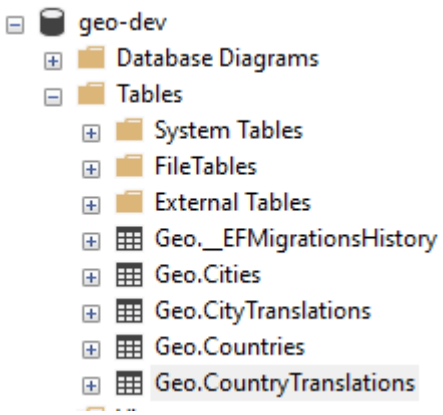
Dodał nam się katalog Migrations:



Teraz tworzymy nasza nową bazę danych poleceniem `update-database`



Jak zalogujemy się do naszego serwera bazodanowego , powinno nam utworzyć baze danych:



Generujemy listę krajów za pomocą migracji:

Dodajmy nową migrację: `Add-migration CountrySeed`

```
PM> Add-migration CoutrySeed
Build started...
Build succeeded.
To undo this action, use Remove-Migration.
```

Wejdźmy w katalog migrations i dodajmy implementację migracji

```
namespace SzkolenieTechniczne.Geo.Storage.Migrations
{
    public partial class CoutrySeed : Migration
    {
        protected override void Up(MigrationBuilder migrationBuilder)
        {

            InsertCountrySector(migrationBuilder, "PL", "Polska", "Poland");
            InsertCountrySector(migrationBuilder, "RU", "Rosja", "Russia");
            InsertCountrySector(migrationBuilder, "IT", "Włochy", "Italy");
            InsertCountrySector(migrationBuilder, "SK", "Słowacja", "Slovakia");
        }

        protected override void Down(MigrationBuilder migrationBuilder)
        {
        }

        private void InsertCountrySector(MigrationBuilder migrationBuilder, string alpha3Code, string namePl, string nameEn)
        {
            Guid countryId = Guid.NewGuid();
            migrationBuilder.InsertData("Countries", schema: "Geo", columns: new[] { "Id", "Alpha3Code"},
                values: new object[] { countryId, alpha3Code });

            var columns = new[] { "Id", "CountryId", "LanguageCode", "Name" };
            migrationBuilder.InsertData("CountryTranslations", schema: "Geo", columns: columns, values: new object[] { Guid.NewGuid(), countryId, "pl", namePl });
            migrationBuilder.InsertData("CountryTranslations", schema: "Geo", columns: columns, values: new object[] { Guid.NewGuid(), countryId, "en", nameEn });
        }
    }
}
```