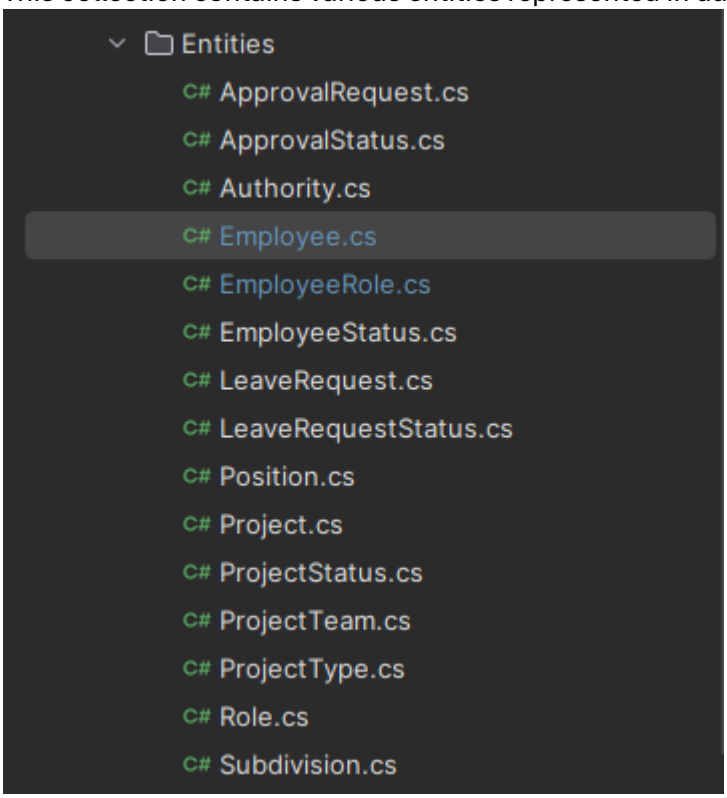
Screenshots with comments

This collection contains various entities represented in database.



The Employee class represents a model for an employee entity in a database. It includes fields for personal identification, login credentials, and associations with specific subdivisions, positions, and statuses. Additionally, the class is designed to handle relationships with other entities, for example recursive relation with “employees”, “project teams”, and etc. Also it is annotated with different attributes to enforce certain behaviors, such as deletion behavior and data validation.

```
[Table(name: "employees")]
20 usages  Vladyslav Kalynchenko  6 exposing APIs
public class Employee
{

    [Key]
    [Column(name: "id")]
    18 usages
    public int Id { get; set; }

    [Required]
    [MaxLength(100)]
    [Column(name: "full_name")]
    8 usages
    public string FullName { get; set; } = null!;

    [Required]
    [MaxLength(100)]
    [Column(name: "login")]
    4 usages
    public string Login { get; set; } = null!;

    [Required]
    [Column(name: "out_of_office_balance")]
    3 usages
    public short OutOfOfficeBalance { get; set; }

    [MaxLength(200)]
    [Column(name: "password")]
    2 usages
    public string Password { get; set; } = null!;

    [Required]
    [Column(name: "Subdivision_id")]
    [DeleteBehavior(DeleteBehavior.Restrict)]
    11 usages
    public Subdivision Subdivision { get; set; } = null!;

    [ForeignKey(nameof(Subdivision))]
    public int SubdivisionId { get; set; }

    [Required]
    [Column(name: "Position_id")]
    [DeleteBehavior(DeleteBehavior.Restrict)]
    11 usages
    public Position Position { get; set; } = null!;

    [ForeignKey(nameof(Position))]
    public int PositionId { get; set; }

    [Required]
    [Column(name: "employee_status_id")]
    [DeleteBehavior(DeleteBehavior.Restrict)]
    14 usages
    public EmployeeStatus Status { get; set; } = null!;
```

```
using ...

namespace OutOfOffice.Entities;

[Table(name: "employee_roles")]
[PrimaryKey(propertyName: "EmployeeId", params additionalPropertyNames: "RoleId")]
7 usages  Vladyslav Kalynchenko *
public class EmployeeRole

{

    [Column(name: "fk_employee_id")]
    [ForeignKey(name: "fk_employee")]
    public int EmployeeId { get; set; }

    2 usages
    public Employee Employee { get; set; } = null!;

    [Column(name: "fk_role_id")]
    [ForeignKey(name: "fk_role")]
    public int RoleId { get; set; }

    13 usages
    public Role Role { get; set; } = null!;
}
```

The EmoployeeRoles Entity represents junction table that used to represent m-tm relation in database between table ‘roles’ and ‘employees’.

Screenshot below shows Project entity that represents ‘projects’ table in database. It has many relations with other tables, for example it has m-to-m relationship with ‘project-teams table. In code it represented by ICollection<ProjectTeam>

```
namespace OutOfOffice.Entities;

[Table( name: "projects")]
7 usages  Vladyslav Kalynchenko  1 exposing API  More...
public class Project
{
    [Key]
    [Column( name: "id")]
    2 usages
    public int Id { get; set; }

    [Required]
    [Column( name: "project_status_id")]
    7 usages
    public ProjectStatus Status { get; set; } = null!;

    [ForeignKey(nameof(Status))]
    public int ProjectStatusId { get; set; }

    [Required]
    [Column( name: "project_type_id")]
    6 usages
    public ProjectType Type { get; set; } = null!;

    [ForeignKey(nameof(Type))]
    public int ProjectTypeId { get; set; }

    [Required]
    [Column( name: "start_date")]
    2 usages
    public DateTime StartDate { get; set; }

    [Column( name: "end_date")]
    2 usages
    public DateTime? EndDate { get; set; }

    [Required]
    [Column( name: "project_manager_id")]
    6 usages
    public Employee ProjectManager { get; set; } = null!;

    [ForeignKey(nameof(ProjectManager))]
    public int ProjectManagerId { get; set; }

    [Column( name: "comment", TypeName = "text")]
    2 usages
    public string? Comment { get; set; } = null!;

    9 usages
    public ICollection<ProjectTeam> ProjectTeams { get; set; } = null!;

}
```

```
using ...

namespace OutOfOffice.Entities;

[Table( name: "project_teams")]
[PrimaryKey( propertyName: "EmployeeId",  params additionalPropertyNames: "ProjectId")]
6 usages  Vladyslav Kalynchenko
public class ProjectTeam
{
    [Column( name: "employee_id")]
    9 usages
    public Employee Employee { get; set; } = null!;

    [Column( name: "project_id"), ]
    3 usages
    public Project Project { get; set; } = null!;

    [ForeignKey( name: "Project")]
    public int ProjectId { get; set; }

    [ForeignKey( name: "Employee")]
    public int EmployeeId { get; set; }

}
```

Entity ProjectType represents ‘project_teams’ table in database that serves as junction table for m-to-m relation between ‘projects’ and ‘employees’.

On the screenshot below you can see EmployeeController that can be accessed by such path domain/list/employees. Also you can see that every method is annotated by [Authorize(Roles= "...")] which enforce every call to be authorized and checked whether or not user has got required roles to access method. Roles are securely saved in JWT token that should be included in request headers. All my controllers are made in the same way.

```
[Route("list/employees")]
[ApiController]
Vladyslav Kalynchenko  More...
public class EmployeeController : ControllerBase
{

    private readonly IEmployeeService _employeeService;

    Vladyslav Kalynchenko
    public EmployeeController(IEmployeeService employeeService)
    {
        _employeeService = employeeService;
    }

    [HttpGet]
    [Authorize(Roles = "HR, PM")]
    Vladyslav Kalynchenko
    public async Task<IActionResult> GetEmployees(
        [FromQuery] PageRequest request
        )
    {
        var pagedList = await _employeeService.GetEmployees(request);
        var response = new PageListResponse<EmployeeResponse>()
        {
            CurrentPage = pagedList.CurrentPage,
            TotalPages = pagedList.TotalPages,
            PageSize = pagedList.PageSize,
            TotalCount = pagedList.TotalCount,
            Items = pagedList.ToList()
        };

        return Ok(response);
    }

    [HttpGet("{id}")]
    [Authorize(Roles = "HR, PM")]
    Vladyslav Kalynchenko
    public async Task<IActionResult> GetEmployee(int id)
    {
        return Ok(await _employeeService.GetEmployee(id));
    }

    [HttpPost]
    [Authorize(Roles = "HR")]
    Vladyslav Kalynchenko
    public async Task<IActionResult> AddEmployee(
        [FromBody] RegisterRequest request
    )
    {
        await _employeeService.AddEmployee(request);
        return Created();
    }

    [HttpPut("{id}")]
    [Authorize(Roles = "HR")]
    Vladyslav Kalynchenko
    public async Task<IActionResult> UpdateEmployee(
```

```
using ...

namespace OutOfOffice.Utils;

22 usages  Vladyslav Kalynchenko
public class PagedList<T> : List<T>
{
    13 usages
    public int CurrentPage { get; private set; }
    8 usages
    public int TotalPages { get; private set; }
    11 usages
    public int PageSize { get; private set; }
    11 usages
    public int TotalCount { get; private set; }

    Vladyslav Kalynchenko
    public bool HasPrevious => CurrentPage > 1;
    Vladyslav Kalynchenko
    public bool HasNext => CurrentPage < TotalPages;

    5 usages  Vladyslav Kalynchenko
    public PagedList(List<T> items, int count, int pageNumber, int pageSize)
    {
        TotalCount = count;
        PageSize = pageSize;
        CurrentPage = pageNumber;
        TotalPages = (int)Math.Ceiling(count / (double)pageSize);

        AddRange(items);
    }

    4 usages  Vladyslav Kalynchenko
    public static async Task<PagedList<T>> ToPagedListAsync(IQueryable<T> source, int pageNumber, int p
    {
        var count = await source.CountAsync();
        var items = await source.Skip((pageNumber - 1) * pageSize).Take(pageSize).ToListAsync();

        return new PagedList<T>(items, count, pageNumber, pageSize);
    }
}
```

Additionally you can see I've created my own collection 'PagedList' which is used to implement pagination for data response. That helps not to send large datasets at once for faster responses and better user experience.

EmployeeService implements IEmployeeService, ensuring adherence to the Interface Segregation Principle. This service is utilized in EmployeeController, which segregates the view from business logic. All of my services are made in the same way.

```
public class EmployeeService(ApplicationContext _context, IAuthService authService, IMapper _mapper) : IEmployeeService
{

    private static readonly string[] AllowedSortColumns = { "id", "fullName", "outOfOfficeBalance", "subdivision.name", "position.name", "status.status" };
    private static readonly string[] AllowedSortDirections = { "asc", "desc" };


    0+1 usages   Vladyslav Kalynchenko
    public async Task<PagedList<EmployeeResponse>> GetEmployees(PageRequest request)
    {

        IQueryable<Employee> query = _context.Employees // DbSet<Employee>
        .Include(navigationPropertyPath: e => e.Subdivision) // IIncludableQueryable<Employee,Subdivision>
        .Include(navigationPropertyPath: e => e.Position) // IIncludableQueryable<Employee,Position>
        .Include(navigationPropertyPath: e => e.Status) // IIncludableQueryable<Employee,EmployeeStatus>
        .Where(e => e.Status.Status != EmployeeStatusType.Deleted) // IQueryable<Employee>
        .Include(navigationPropertyPath: e => e.Roles) // IIncludableQueryable<Employee,ICollection<...>>
            .ThenInclude(er => er.Role) // IIncludableQueryable<Employee,Role>
        .AsQueryable();
        if (!AllowedSortDirections.Contains(request.SortDirection) ||
            !AllowedSortColumns.Contains(request.SortBy))
        {
            throw new BadRequestParameters(message: "Bad sorting parameters");
        }

        string sorting = $"{request.SortBy} {request.SortDirection}";
        query = query.OrderBy(sorting);

        var pagedEmployees = PagedList<Employee>.ToPagedListAsync(query, pageNumber: request.Page, request.PageSize);
        var employeeResponses = pagedEmployees.Select(e => _mapper.Map<EmployeeResponse>(e)).ToList();

        return new PagedList<EmployeeResponse>(employeeResponses, pagedEmployees.TotalCount, pageNumber: pagedEmployees.CurrentPage, pagedEmployees.PageSize);
    }

    0+1 usages   Vladyslav Kalynchenko
    public async Task<EmployeeResponse> GetEmployee(int id)
    {
        var emp = GetEmployeeById(id);
        return _mapper.Map<EmployeeResponse>(emp);
    }

    0+1 usages   Vladyslav Kalynchenko
    public async Task<EmployeeResponse> AddEmployee(RegisterRequest request)
    {
        return await authService.RegisterUser(request);
    }

    0+1 usages   Vladyslav Kalynchenko
    public async Task<EmployeeResponse> UpdateEmployee(int id, UpdateEmployeeRequest request)
    {
        var emp = GetEmployeeById(id);

        if (request.FullName != null) emp.FullName = request.FullName;
        if (request.OutOfOfficeBalance != null) emp.OutOfOfficeBalance = request.OutOfOfficeBalance.Value;
        if (request.SubdivisionId != null)
        {
```

```
                    .FirstOrDefaultAsync(p => p.Id == projectId) // Task<Project?>
                ?? throw new NotFoundException(message: $"Project not found with id {projectId}", statusCode: 404);
        if(project.ProjectTeams.Any(pt => pt.Employee.Id == id))
        {
            throw new BadRequestParameters(message: "Employee already in project");
        }
        project.ProjectTeams.Add(item: new ProjectTeam
        {
            Employee = employee,
            Project = project
        });
        await _context.SaveChangesAsync();
    }

    3 usages   Vladyslav Kalynchenko
    private async Task<Employee> GetEmployeeById(int id)
    {
        return await _context.Employees // DbSet<Employee>
                    .Include(navigationPropertyPath: e => e.Subdivision) // IIncludableQueryable<Employee,Subdivision>
                    .Include(navigationPropertyPath: e => e.Position) // IIncludableQueryable<Employee,Position>
                    .Include(navigationPropertyPath: e => e.Status) // IIncludableQueryable<Employee,EmployeeStatus>
                    .Where(e => e.Status.Status != EmployeeStatusType.Deleted) // IQueryable<Employee>
                    .Include(navigationPropertyPath: e => e.Roles) // IIncludableQueryable<Employee,ICollection<...>>
                    .ThenInclude(er => er.Role) // IIncludableQueryable<Employee,Role>
                    .FirstOrDefaultAsync(e => e.Id == id) // Task<Employee?>
                ?? throw new NotFoundException(message: $"Employee not found with id {id}", statusCode: 404);
    }
}

4 usages   1 inheritor   Vladyslav Kalynchenko
public interface IEmployeeService
{
    1 usage   1 implementation   Vladyslav Kalynchenko
    Task<PagedList<EmployeeResponse>> GetEmployees(PageRequest request);

    1 usage   1 implementation   Vladyslav Kalynchenko
    Task<EmployeeResponse> GetEmployee(int id);

    1 usage   1 implementation   Vladyslav Kalynchenko
    Task<EmployeeResponse> AddEmployee(RegisterRequest request);

    1 usage   1 implementation   Vladyslav Kalynchenko
    Task<EmployeeResponse> UpdateEmployee(int id, UpdateEmployeeRequest request);

    1 usage   1 implementation   Vladyslav Kalynchenko
    Task DeactivateEmployee(int id);

    1 usage   1 implementation   Vladyslav Kalynchenko
    Task AddEmployeeToProject(int id, int projectId);
}
```

Those two code snippets represent three middleware's, 'ExceptionHandler' is used for uniform way of sending back to used exceptions. Other two is a common way in ASP.NET to implement authorization and authentication using JWT token.

```
public class ExceptionHandler
{
    private readonly RequestDelegate _next;

    Vladyslav Kalynchenko
    public ExceptionHandler(RequestDelegate next)
    {
        _next = next;
    }

    Vladyslav Kalynchenko
    public async Task InvokeAsync(HttpContext context)
    {
        try
        {
            await _next(context);
        }
        catch (NotFoundException ex)
        {
            await ReturnResponse(context, ex.StatusCode, ex.Message);
        }
        catch (UserExitsException ex)
        {
            await ReturnResponse(context, ex.StatusCode, ex.Message);
        }
        catch (Exception ex)
        {
            await ReturnResponse(context, statusCode: 500, ex.Message);
        }
    }

    3 usages  Vladyslav Kalynchenko
    private async Task ReturnResponse(HttpContext context, int statusCode, string message)
    {
        context.Response.StatusCode = statusCode;
        context.Response.ContentType = "application/json";
        string jsonResponse = JsonSerializer.Serialize(new
        {
            Message = message,
            Time = DateTime.Now
        });


        await context.Response.WriteAsync(jsonResponse);
    }
}
```

```
builder.Services.AddAuthentication(configureOptions: options =>
    {
        options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
        options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
    }
).AddJwtBearer(opt JwtBearerOptions =>
{
    opt.TokenValidationParameters = new TokenValidationParameters
    {
        ValidateIssuer = true,
        ValidateAudience = true,
        ValidateLifetime = true,
        ValidateIssuerSigningKey = true,

        ValidIssuer = builder.Configuration["JWT:Issuer"],
        ValidAudience = builder.Configuration["JWT:Audience"],
        IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(builder.Configuration["JWT:Key"]!)),

        ClockSkew = TimeSpan.Zero
    };
}).AddJwtBearer(authenticationScheme: "IgnoreTokenExpirationScheme", configureOptions: opt JwtBearerOptions =>
{
    opt.TokenValidationParameters = new TokenValidationParameters
    {
        ValidateIssuer = true,
        ValidateAudience = true,
        ValidateLifetime = true,
        ValidateIssuerSigningKey = true,
        ValidIssuer = builder.Configuration["JWT:RefIssuer"],
        ValidAudience = builder.Configuration["JWT:RefAudience"],
        IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(builder.Configuration["JWT:RefKey"]!))
    };
});
```

In the code below is used in AuthService in order to generate JWT accses token and JWT refresh token.

```
2 usages  Vladyslav Kalynchenko *
public string GenerateJwtToken(Employee user)
{

    var roles string[] = user.Roles.Select(re EmployeeRole => re.Role.RoleName).ToArray();

    var claims = new List<Claim>
    {
        new Claim(type: "name", value: user.FullName),
        new Claim(type: "id", value: $"{user.Id}"),
    };

    claims.AddRange(collection: roles.Select(role string => new Claim(type: "role", value: role)));

    Claim[] userClaims = claims.ToArray();


    SymmetricSecurityKey key = new(Encoding.UTF8.GetBytes(_configuration["JWT:Key"]!));
    SigningCredentials credentials = new(key, algorithm: SecurityAlgorithms.HmacSha256);

    JwtSecurityToken token = new(
        issuer: _configuration["JWT:Issuer"],
        audience: _configuration["JWT:Audience"],
        userClaims,
        expires: DateTime.Now.AddMinutes(1),
        signingCredentials: credentials
    );

    return new JwtSecurityTokenHandler().WriteToken(token);
}

2 usages  Vladyslav Kalynchenko
public string GenerateRefreshToken(Employee user)
{
    Claim[] userClaims = new[]
    {
        new Claim(type: "name", value: user.FullName),
    };

    SymmetricSecurityKey key = new(Encoding.UTF8.GetBytes(_configuration["JWT:RefKey"]!));
    SigningCredentials credentials = new(key, algorithm: SecurityAlgorithms.HmacSha256);

    JwtSecurityToken token = new(
        issuer: _configuration["JWT:RefIssuer"],
        audience: _configuration["JWT:RefAudience"],
        userClaims,
        expires: DateTime.Now.AddDays(3),
        signingCredentials: credentials
    );

    return new JwtSecurityTokenHandler().WriteToken(token);
}
```

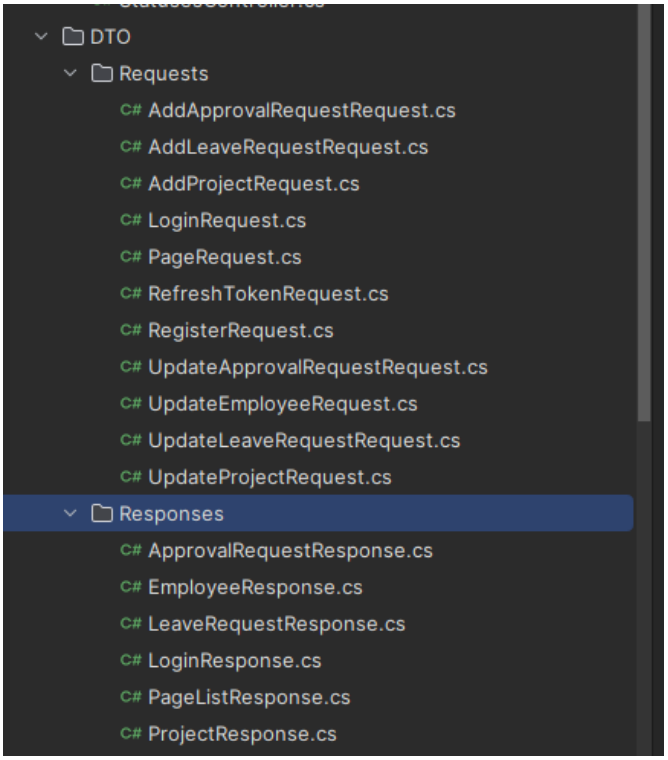
This code below is showing helping class that maps Entities to DTO objects. It is automatically injected in different services with 'IMapper' interface.

```
Vladyslav Kalynchenko
public class AutoMapper : Profile
{
    Vladyslav Kalynchenko
    public AutoMapper()
    {
        CreateMap<Employee, EmployeeResponse>()
            .ForMember( destinationMember: dest EmployeeResponse => dest.Position, memberOptions: opt IMemberConfigurationExpression<Employee,EmployeeResponse,...> => opt.MapFrom( mapExpression: src Employee => src.Position.Name))
            .ForMember( destinationMember: dest EmployeeResponse => dest.Subdivision, memberOptions: opt IMemberConfigurationExpression<Employee,EmployeeResponse,...> => opt.MapFrom( mapExpression: src Employee => src.Subdivision.Name))
            .ForMember( destinationMember: dest EmployeeResponse => dest.Status, memberOptions: opt IMemberConfigurationExpression<Employee,EmployeeResponse,...> => opt.MapFrom( mapExpression: src Employee => src.Status.Status))
            .ForMember( destinationMember: dest EmployeeResponse => dest.PartnerId, memberOptions: opt IMemberConfigurationExpression<Employee,EmployeeResponse,...> => opt.MapFrom( mapExpression: src Employee => src.PeoplePartnerId))
            .ForMember( destinationMember: dest EmployeeResponse => dest.Roles, memberOptions: opt IMemberConfigurationExpression<Employee,EmployeeResponse,...> => opt.MapFrom( mapExpression: src Employee => src.Roles.Select(er EmployeeRole => er.Role.RoleName).ToList()));

        CreateMap<ApprovalRequest, ApprovalRequestResponse>()
            .ForMember( destinationMember: dest ApprovalRequestResponse => dest.Status, memberOptions: opt IMemberConfigurationExpression<ApprovalRequest,ApprovalRequestResponse,...> => opt.MapFrom( mapExpression: src ApprovalRequest => src.ApprovalStatus.Status))
            .ForMember( destinationMember: dest ApprovalRequestResponse => dest.Employee, memberOptions: opt IMemberConfigurationExpression<ApprovalRequest,ApprovalRequestResponse,...> => opt.MapFrom( mapExpression: src ApprovalRequest => src.Employee));

        CreateMap<LeaveRequest, LeaveRequestResponse>()
            .ForMember( destinationMember: dest LeaveRequestResponse => dest.Status, memberOptions: opt IMemberConfigurationExpression<LeaveRequest,LeaveRequestResponse,...> => opt.MapFrom( mapExpression: src LeaveRequest => src.Status.Status))
            .ForMember( destinationMember: dest LeaveRequestResponse => dest.Employee, memberOptions: opt IMemberConfigurationExpression<LeaveRequest,LeaveRequestResponse,...> => opt.MapFrom( mapExpression: src LeaveRequest => src.Employee));

        CreateMap<Project, ProjectResponse>()
            .ForMember( destinationMember: dest ProjectResponse => dest.ProjectManager, memberOptions: opt IMemberConfigurationExpression<Project,ProjectResponse,...> => opt.MapFrom( mapExpression: src Project => src.ProjectManager.FullName))
            .ForMember( destinationMember: dest ProjectResponse => dest.ProjectStatus, memberOptions: opt IMemberConfigurationExpression<Project,ProjectResponse,...> => opt.MapFrom( mapExpression: src Project => src.Status.Status))
            .ForMember( destinationMember: dest ProjectResponse => dest.ProjectType, memberOptions: opt IMemberConfigurationExpression<Project,ProjectResponse,...> => opt.MapFrom( mapExpression: src Project => src.Type.Type))
            .ForMember( destinationMember: dest ProjectResponse => dest.Employees, memberOptions: opt IMemberConfigurationExpression<Project,ProjectResponse,...> => opt.MapFrom( mapExpression: src Project => src.ProjectTeams.Select(pt ProjectTeam => pt.Employee.FullName).ToList()));

    }
}
```

Code below is showing all services are inserted by DI container. AddScoped means that one service will be created and inserted in every call where it needed.

```
builder.Services.AddScoped<IAuthService, AuthService>();

builder.Services.AddScoped<IEmployeeService, EmployeeService>();
builder.Services.AddScoped<IAprovalRequestService, ApprovalRequestService>();
builder.Services.AddScoped<ILeaveRequestService, LeaveRequestService>();
builder.Services.AddScoped<IProjectService, ProjectService>();
builder.Services.AddScoped<IStatusService, StatusService>();
```

Screenshot below are showing different DTO that are used in my application.



The ‘ApplicationContext’ class serves as data access layer of an application configured to interact with a database using the EF.

```
11 usages  Vladyslav Kalynchenko
public class ApplicationContext : DbContext
{
    16 usages
    public DbSet<Employee> Employees { get; set; }
    3 usages
    public DbSet<Subdivision> Subdivisions { get; set; }
    3 usages
    public DbSet<Position> Positions { get; set; }
    3 usages
    public DbSet<Role> Roles { get; set; }
    public DbSet<EmployeeRole> EmployeeRoles { get; set; }
    4 usages
    public DbSet<EmployeeStatus> EmployeeStatuses { get; set; }

    4 usages
    public DbSet<Project> Projects { get; set; }
    public DbSet<ProjectTeam> ProjectTeams { get; set; }
    4 usages
    public DbSet<ProjectStatus> ProjectStatuses { get; set; }
    3 usages
    public DbSet<ProjectType> ProjectTypes { get; set; }

    8 usages
    public DbSet<LeaveRequest> LeaveRequests { get; set; }
    4 usages
    public DbSet<LeaveRequestStatus> LeaveRequestStatuses { get; set; }
    5 usages
    public DbSet<ApprovalRequest> ApprovalRequests { get; set; }
    6 usages
    public DbSet<ApprovalStatus> ApprovalStatuses { get; set; }

    Vladyslav Kalynchenko
    public ApplicationContext()
    {
    }

    Vladyslav Kalynchenko
    public ApplicationContext(DbContextOptions<ApplicationContext> options) : base(options)
    {
    }

    Vladyslav Kalynchenko
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Employee>() // EntityTypeBuilder<Employee>
            .HasIndex(u Employee => u.Login)
            .IsUnique();
    }
}
```

SQL Queries

```sql
-- Create approval_statuses table with constraints
CREATE TABLE approval_statuses (
                                id integer GENERATED BY DEFAULT AS IDENTITY,
                                status character varying(50) NOT NULL,
                                CONSTRAINT "PK_approval_statuses" PRIMARY KEY (id)
);

-- Insert various statuses into the approval_statuses table
INSERT INTO approval_statuses (status)
VALUES ('Approved'), ('New'), ('Canceled'), ('Rejected');

-- Create employee_statuses table with constraints
CREATE TABLE employee_statuses (
                                id integer GENERATED BY DEFAULT AS IDENTITY,
                                status character varying(50) NOT NULL,
                                CONSTRAINT "PK_employee_statuses" PRIMARY KEY (id)
);

-- Insert various statuses into the employee_statuses table
INSERT INTO employee_statuses (status)
VALUES ('Active'), ('Inactive'), ('On Vacation'), ('Deleted');

-- Create leave_request_statuses table with constraints
CREATE TABLE leave_request_statuses (
                                     id integer GENERATED BY DEFAULT AS IDENTITY,
                                     status character varying(50) NOT NULL,
                                     CONSTRAINT "PK_leave_request_statuses" PRIMARY KEY (id)
);

-- Insert various statuses into the leave_request_statuses table
INSERT INTO leave_request_statuses (status)
VALUES ('Approved'), ('Canceled'), ('Created'), ('Rejected'), ('Submitted');

-- Create positions table with constraints
CREATE TABLE positions (
                        id integer GENERATED BY DEFAULT AS IDENTITY,
                        position_name character varying(50) NOT NULL,
                        CONSTRAINT "PK_positions" PRIMARY KEY (id)
);

-- Insert various position names into the positions table
INSERT INTO positions (position_name)
VALUES ('Junior'), ('Middle'), ('Senior'), ('Lead');

-- Create project_statuses table with constraints
CREATE TABLE project_statuses (
                               id integer GENERATED BY DEFAULT AS IDENTITY,
                               status character varying(50) NOT NULL,
                               CONSTRAINT "PK_project_statuses" PRIMARY KEY (id)
);

-- Insert various statuses into the project_statuses table
INSERT INTO project_statuses (status)
VALUES ('Active'), ('Canceled'), ('Inactive'), ('New');
```

```sql
-- Create project_types table with constraints
CREATE TABLE project_types (
                                id integer GENERATED BY DEFAULT AS IDENTITY,
                                type character varying(50) NOT NULL,
                                CONSTRAINT "PK_project_types" PRIMARY KEY (id)
);

-- Insert various project types into the project_types table
INSERT INTO project_types (type)
VALUES ('Application'), ('Service'), ('Website');

-- Create roles table with constraints
CREATE TABLE roles (
                        id integer GENERATED BY DEFAULT AS IDENTITY,
                        role_name character varying(50) NOT NULL,
                        CONSTRAINT "PK_roles" PRIMARY KEY (id)
);

-- Insert various roles into the roles table
INSERT INTO roles (role_name)
VALUES ('Admin'), ('HR'), ('EMP'), ('PM');

-- Create subdivisions table with constraints
CREATE TABLE subdivisions (
                                id integer GENERATED BY DEFAULT AS IDENTITY,
                                subdivision_name character varying(50) NOT NULL,
                                CONSTRAINT "PK_subdivisions" PRIMARY KEY (id)
);

-- Insert various subdivision names into the subdivisions table
INSERT INTO subdivisions (subdivision_name)
VALUES ('HR'), ('IT'), ('PR'), ('Sales'), ('Security'), ('Support');

-- Create employees table with constraints and foreign keys
CREATE TABLE employees (
                            id integer GENERATED BY DEFAULT AS IDENTITY,
                            full_name character varying(100) NOT NULL,
                            login character varying(100) NOT NULL,
                            out_of_office_balance smallint NOT NULL,
                            password character varying(200) NOT NULL,
                            "SubdivisionId" integer NOT NULL,
                            "PositionId" integer NOT NULL,
                            "StatusId" integer NOT NULL,
                            "PeoplePartnerId" integer,
                            CONSTRAINT "PK_employees" PRIMARY KEY (id),
                            CONSTRAINT "FK_employees_employee_statuses_StatusId" FOREIGN KEY ("StatusId") REFERENCES employee_statuses (id) ON DELETE RESTRICT,
                            CONSTRAINT "FK_employees_employees_PeoplePartnerId" FOREIGN KEY ("PeoplePartnerId") REFERENCES employees (id) ON DELETE RESTRICT,
                            CONSTRAINT "FK_employees_positions_PositionId" FOREIGN KEY ("PositionId") REFERENCES positions (id) ON DELETE RESTRICT,
                            CONSTRAINT "FK_employees_subdivisions_SubdivisionId" FOREIGN KEY ("SubdivisionId") REFERENCES subdivisions (id) ON DELETE RESTRICT
);
```

```sql
-- Create employee_roles table with constraints and foreign keys
CREATE TABLE employee_roles (
                                fk_employee_id integer NOT NULL,
                                fk_role_id integer NOT NULL,
                                CONSTRAINT "PK_employee_roles" PRIMARY KEY (fk_employee_id, fk_role_id),
                                CONSTRAINT "FK_employee_roles_employees_fk_employee_id" FOREIGN KEY (fk_employee_id) REFERENCES employees (id) ON DELETE CASCADE,
                                CONSTRAINT "FK_employee_roles_roles_fk_role_id" FOREIGN KEY (fk_role_id) REFERENCES roles (id) ON DELETE CASCADE
);

-- Create leave_requests table with constraints and foreign keys
CREATE TABLE leave_requests (
                                id integer GENERATED BY DEFAULT AS IDENTITY,
                                absence_reason character varying(255) NOT NULL,
                                start_date timestamp with time zone NOT NULL,
                                end_date timestamp with time zone NOT NULL,
                                comment text,
                                "LeaveRequestStatusId" integer NOT NULL,
                                "EmployeeId" integer NOT NULL,
                                CONSTRAINT "PK_leave_requests" PRIMARY KEY (id),
                                CONSTRAINT "FK_leave_requests_employees_EmployeeId" FOREIGN KEY ("EmployeeId") REFERENCES employees (id) ON DELETE CASCADE,
                                CONSTRAINT "FK_leave_requests_leave_request_statuses_LeaveRequestStatusId" FOREIGN KEY ("LeaveRequestStatusId") REFERENCES leave_request_statuses (id) ON DELETE CASCADE
);

-- Create projects table with constraints and foreign keys
CREATE TABLE projects (
                          id integer GENERATED BY DEFAULT AS IDENTITY,
                          "ProjectStatusId" integer NOT NULL,
                          "ProjectTypeId" integer NOT NULL,
                          start_date timestamp with time zone NOT NULL,
                          end_date timestamp with time zone,
                          "ProjectManagerId" integer NOT NULL,
                          comment text,
                          CONSTRAINT "PK_projects" PRIMARY KEY (id),
                          CONSTRAINT "FK_projects_employees_ProjectManagerId" FOREIGN KEY ("ProjectManagerId") REFERENCES employees (id) ON DELETE CASCADE,
                          CONSTRAINT "FK_projects_project_statuses_ProjectStatusId" FOREIGN KEY ("ProjectStatusId") REFERENCES project_statuses (id) ON DELETE CASCADE,
                          CONSTRAINT "FK_projects_project_types_ProjectTypeId" FOREIGN KEY ("ProjectTypeId") REFERENCES project_types (id) ON DELETE CASCADE
);

-- Create approval_requests table with constraints and foreign keys
CREATE TABLE approval_requests (
                                   id integer GENERATED BY DEFAULT AS IDENTITY,
                                   comment text,
                                   "ApprovalStatusId" integer NOT NULL,
                                   "LeaveRequestId" integer NOT NULL,
                                   "EmployeeId" integer NOT NULL,
                                   CONSTRAINT "PK_approval_requests" PRIMARY KEY (id),
                                   CONSTRAINT "FK_approval_requests_approval_statuses_ApprovalStatusId" FOREIGN KEY ("ApprovalStatusId") REFERENCES approval_statuses (id) ON DELETE CASCADE,
                                   CONSTRAINT "FK_approval_requests_employees_EmployeeId" FOREIGN KEY ("EmployeeId") REFERENCES employees (id) ON DELETE CASCADE,
                                   CONSTRAINT "FK_approval_requests_leave_requests_LeaveRequestId" FOREIGN KEY ("LeaveRequestId") REFERENCES leave_requests (id) ON DELETE CASCADE
);
```

```sql
-- Create project_teams table with constraints and foreign keys
CREATE TABLE project_teams (
                                "ProjectId" integer NOT NULL,
                                "EmployeeId" integer NOT NULL,
                                CONSTRAINT "PK_project_teams" PRIMARY KEY ("EmployeeId", "ProjectId"),
                                CONSTRAINT "FK_project_teams_employees_EmployeeId" FOREIGN KEY ("EmployeeId") REFERENCES employees (id) ON DELETE CASCADE,
                                CONSTRAINT "FK_project_teams_projects_ProjectId" FOREIGN KEY ("ProjectId") REFERENCES projects (id) ON DELETE CASCADE
);

-- Create various indexes for the tables
CREATE INDEX "IX_approval_requests_ApprovalStatusId" ON approval_requests ("ApprovalStatusId");
CREATE INDEX "IX_approval_requests_EmployeeId" ON approval_requests ("EmployeeId");
CREATE INDEX "IX_approval_requests_LeaveRequestId" ON approval_requests ("LeaveRequestId");
CREATE INDEX "IX_employee_roles_fk_role_id" ON employee_roles (fk_role_id);
CREATE UNIQUE INDEX "IX_employees_login" ON employees (login);
CREATE INDEX "IX_employees_PeoplePartnerId" ON employees ("PeoplePartnerId");
CREATE INDEX "IX_employees_PositionId" ON employees ("PositionId");
CREATE INDEX "IX_employees_StatusId" ON employees ("StatusId");
CREATE INDEX "IX_employees_SubdivisionId" ON employees ("SubdivisionId");
CREATE INDEX "IX_leave_requests_EmployeeId" ON leave_requests ("EmployeeId");
CREATE INDEX "IX_leave_requests_LeaveRequestStatusId" ON leave_requests ("LeaveRequestStatusId");
CREATE INDEX "IX_project_teams_ProjectId" ON project_teams ("ProjectId");
CREATE INDEX "IX_projects_ProjectManagerId" ON projects ("ProjectManagerId");
CREATE INDEX "IX_projects_ProjectStatusId" ON projects ("ProjectStatusId");
CREATE INDEX "IX_projects_ProjectTypeId" ON projects ("ProjectTypeId");
```

```sql
-- Adjust sequence values for various tables
SELECT setval(
        pg_get_serial_sequence('approval_statuses', 'id'),
        GREATEST(
            (SELECT MAX(id) FROM approval_statuses) + 1,
            nextval(pg_get_serial_sequence('approval_statuses', 'id'))),
        false);
SELECT setval(
        pg_get_serial_sequence('employee_statuses', 'id'),
        GREATEST(
            (SELECT MAX(id) FROM employee_statuses) + 1,
            nextval(pg_get_serial_sequence('employee_statuses', 'id'))),
        false);
SELECT setval(
        pg_get_serial_sequence('leave_request_statuses', 'id'),
        GREATEST(
            (SELECT MAX(id) FROM leave_request_statuses) + 1,
            nextval(pg_get_serial_sequence('leave_request_statuses', 'id'))),
        false);
SELECT setval(
        pg_get_serial_sequence('positions', 'id'),
        GREATEST(
            (SELECT MAX(id) FROM positions) + 1,
            nextval(pg_get_serial_sequence('positions', 'id'))),
        false);
SELECT setval(
        pg_get_serial_sequence('project_statuses', 'id'),
        GREATEST(
            (SELECT MAX(id) FROM project_statuses) + 1,
            nextval(pg_get_serial_sequence('project_statuses', 'id'))),
        false);
SELECT setval(
        pg_get_serial_sequence('project_types', 'id'),
        GREATEST(
            (SELECT MAX(id) FROM project_types) + 1,
            nextval(pg_get_serial_sequence('project_types', 'id'))),
        false);
SELECT setval(
        pg_get_serial_sequence('roles', 'id'),
        GREATEST(
            (SELECT MAX(id) FROM roles) + 1,
            nextval(pg_get_serial_sequence('roles', 'id'))),
        false);
SELECT setval(
        pg_get_serial_sequence('subdivisions', 'id'),
        GREATEST(
            (SELECT MAX(id) FROM subdivisions) + 1,
            nextval(pg_get_serial_sequence('subdivisions', 'id'))),
        false);
```