# Furkan Karataş

# Mobiroller .Net Core Projesi

# İçindekiler

# 1 – Giriş

## 1.1 – Problemlerin Belirlenmesi

Bu projede problem iki adet *.json dosyamız var. Bir endpoint yardımıyla bu bilgileri import etmek istiyoruz. Dosyalar türkçe ve italyanca bilgiler içermekte. Dosyalar import edildikten sonra *Request Localization* yardımıyla türkçe kullanıcılar için türkçe italyanca kullanıcılar için italyanca bilgilerin sorgulanabilmesini istiyoruz.

## 1.2 – Projenin Tanımı

Bu projede bir api projesi olacaktır. Endpointler yardımıyla üye olma, üye girişi, Json verilerini kaydetme, getirme, silme ve güncelleme gibi işlemleri barındıracaktır.

Bu bilgiler aynı zamanda Türkçe ve İtalyanca olarak iki farklı dilde veri tabanında tutulacak ve gerekli parametreler ile Türkçe veya İtalyanca veriler üzerinde işlemler gerçekleştirilecektir.

## 1.3 – Projenin Gereksinimleri

- Asp.Net Core Api (Net 5.0)
- Entity Framework Core
- MsSQL
- Fluent Validation (Nuget)(9.5.1)
- Autofac (Nuget)(6.1.0)
- Autofac.Extensions.DependencyInjection(Nuget)(7.1.0)
- Newtonsoft.Json(Nuget) (10.0.1)
- Autofac.Extras.DynamicProxy(Nuget) (6.0.0)
- Microsoft.AspNetCore.Authentication.JwtBearer(Nuget) (5.0.7)
- Microsoft.AspNetCore.Hosting.Abstractions(Nuget) (2.2.0)
- Microsoft.AspNetCore.Http(Nuget) (2.2.2)
- Microsoft.AspNetCore.Http.Abstractions(Nuget) (2.2.0)
- Microsoft.EntityFrameworkCore.SqlServer(Nuget) (3.1.11)
- Microsoft.Extensions.DependencyInjection.Abstractions(Nuget) (3.1.11)
- Microsoft.Extensions.Localization.Abstractions(Nuget) (5.0.7)
- Microsoft.IdentityModel.Tokens(Nuget) (6.8.0)
- NETStandard.Library(Nuget) (10.0.1)
- Swashbuckle.AspNetCore(Nuget) (5.6.3)(Dev)
- System.IdentityModel.Tokens.Jwt(Nuget) (6.8.0)

# 2 – Proje Katmanlarının Genel Tanımı

Bu projede toplamda 5 katmandan meydana gelecektir. Bu katmanlar aşağı kısımda genel bir tanımlamaları yapılmıştır.

## 2.1 – Core

En temel katmandır. Bu katmanda bulunan class veya interface'ler tüm yapılara uygulanabilecek bir yapıda geliştirilmelidir. Hiçbir katmandan referans almamalı sadece referans edilmelidir. Bu katmanda çoğunlukla tüm katmaların ortak olarak kullandığı yada en temel kalıtım yada extend edilecek yapılar bu katmanda yer almalıdır.

## 2.2 – Entities

Bu katmanda veri tabanında bulunan tabloları temsil etmesi için oluşturulacak properties classları , interfaceler yada data table object(dto) lar bulunabilir. Bu katmanın görevi veri tabanında bulun veri sütunları temsil etmek veri getirme yada veri gönderme işlemlerinde bu properties yardımıyla işlemlerimizi gerçekleştireceğiz.

## 2.3 – Data Access

Bu katmanda veri tabanı ile bağlantı kurulmaktadır. Entities katmanında tanımladığımız property'ler yardımıyla tanımladığımız modelleri veri tabanı ile bu katmanda ilişkilendiririz. Aynı zamanda Core katmanında tanımladığımız repository yardımıyla bu katmanda veri tabanı CRUD işlemlerini tanımladığımız ve daha fazla işlem kazandırdığımız katmandır.

## 2.4 – Business

Bu katmanda adından da anlaşıldığı üzere iş katmanıdır. Data Access katmanında tanımladığımız operasyonları kullandığımız ve bu operasyonları belli işlemlerden geçirdiğimiz katmandır. Bu işlemeler gelen verinin yada gönden verinin doğruluğunu kontrol etmek, verileri belleğe yazdırmak yada silmek gibi işlemleri bu katmanda yer almasını sağlayacağız.

## 2.5 – API

Bu katmanda verilerin endpointler yardımıyla başka servisler tarafından kullanabilmesi için kodlandığı alandır. Verileri bir endpoint vasıtasıyla alması yada gönderdiği ve bu verilerin business katmanında yer alan gerekli alanlara gönderilmesini ve tekrardan business katmanından gelen verileri servise iletmek için kullanacağız.

# 3 – Veritabanı Mimarisi

## 3.1 – Veritabanının Genel Yapısı

Veritabanı olarak MsSql kullanacağız. Toplamda 5 tablodan meydana gelecek veritabanımız bunlar Users, UserOperationClaims, OprationClaims, trDatas ve ItDatas olacaktır.

## 3.2 – Authentication için Gereksinimler

Authentication işlemi sırasında ihtiyaç duyduğumuz tablolar Users, UserOperationClaims ve OprationClaims tablolarıdır. Bu tablolardan Users tablosu Authentication işlemi gerçekleştirecek verileri tutacaktır.

| | |
|---|---|
| Id | İnt,auto increment,primary key |
| FirstName | Nvarchar(50) |
| LastName | Nvarchar(50) |
| Email | Nvarchar(50) |
| PasswordHash | Varbinary(500) |
| PasswordSalt | Varbinary(500) |
| Status | bit |

User tablosu

Bu tablolardan OprationClaims Authentication işleminin yetki seviyelerimin kayıt altında tutulan tablosudur.

| | |
|---|---|
| Id | İnt,auto increment,primary key |
| Name | Nvarchar(250) |

OprationClaims tablosu

Bu tablolardan UserOperationClaims, Users tablosunda bulunan veriler ile OperationClaims tablosunda bulunan verileri eşleştirmek için kullanılmaktadır. Aralarında Çok a Çok (n – n) ilişki vardır.

| | |
|---|---|
| Id | İnt,auto increment,primary key |
| UserId | İnt |
| OperationClaimId | İnt |

UserOperationClaims tablosu

## 3.3 – Hedef Veriler için Gereksinimler

Json olarak verilerin kayıt edildiği tablolardır. Bu tablolar trDatas Türkçe verilerin tutulduğu tablo ve ItDatas İtalyanca verilerin tutulduğu tablodur.

| | |
|---|---|
| Id | İnt ,primary key |
| dc_Zaman | Nvarchar(50) |
| dc_Kategori | Nvarchar(50) |
| dc_Olay | Nvarchar(50) |

trDatas tablosu

| | |
|---|---|
| Id | İnt ,primary key |
| dc_Oraria | Nvarchar(50) |
| dc_Categoria | Nvarchar(50) |
| dc_Evento | Nvarchar(50) |

ItDatas tablosu

# 4 – Core Katmanı

## 4.1 Entities

Bu kısımda en temel properti yada vb. Sınıfların parent interface/class'larını tutmaktayız.

```
namespace Core.Entities
{
    public interface IEntity
    {
    }
}
```

IEntity.cs

IEntity interface'i veritabanı tablolarını temsil eden sınıfların implement edeceği sınıf olacaktır.

```
namespace Core.Entities
{
    public interface IDto
    {
    }
}
```

IDto.cs

IDto interface'i veritabanı tablolarını temsil eden classların dışında gerekli verilerin alındığı yada özel sql soruguları gerektiren durumlarda kullanılan ve bu sorgulara veritabanından veri getirecek yada iletecek sınıfların implement edeceği interfacetir.

### 4.1.1 Concrete

```
namespace Core.Entities.Concrete
{
    public class User:IEntity
    {
        public int Id { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public string Email { get; set; }
        public byte[] PasswordHash { get; set; }
        public byte[] PasswordSalt { get; set; }
        public bool Status { get; set; }
    }
}
```

User.cs

Veritabanında bulunan Users tablosunu karşılayan sınıftır. Yukarıda belirtildiği gibi IEntity'den implement edilmiştir.

```
namespace Core.Entities.Concrete
{
    public class OperationClaim:IEntity
    {
        public int Id { get; set; }
        public string Name { get; set; }
    }
}
```

OperationClaim.cs

Veritabanında bulunan OperationClaim tablosunu karşılayan sınıftır. Yukarıda belirtildiği gibi IEntity'den implement edilmiştir.

```
namespace Core.Entities.Concrete
{
    public class UserOperationClaim : IEntity
    {
        public int Id { get; set; }
        public int UserId { get; set; }
        public int OperationClaimId { get; set; }
    }
}
```

UserOperationClaim.cs

Veritabanında bulunan UserOperationClaim tablosunu karşılayan sınıftır. Yukarıda belirtildiği gibi IEntity'den implement edilmiştir.

## 4.2 Data Access

Bu kısımda Data Access katmanında(Class Library) bulunan ortak yapıların olacağı konumdur.

```
using Core.Entities;
using System;
using System.Collections.Generic;
using System.Linq.Expressions;

namespace Core.DataAccess
{
    public interface IEntityRepository<T> where T : class, IEntity, new()
    {
        List<T> GetAll(Expression<Func<T, bool>> filter = null);
        T Get(Expression<Func<T, bool>> filter);
        void Add(T entity);
        void Update(T entity);
        void Delete(T entity);

    }
}
```

IEntityRepository.cs

Veritabanın için uygulacak olan CRUD operasyonları için bir klavuz niteliği taşıması için IEntityRepository interface'I oluşturulmuştur. Bu interface'I implement edeceğimiz sınıfta bu yapıların iç kısımlarını dolduracağız. Bu projede veritabanı işlemleri için Entity Framework kullacağız, ileride başka bir yapıya geçmek istediğimizde bu sınıfı implement etmemiz ve gerekli alanları doldurmamız ardından dependency injection değerini değiştirmemiz yeterli olacaktır.

### 4.2.1 Entity Framework

```
using Core.Entities;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Linq.Expressions;

namespace Core.DataAccess.EntityFramework
{
    public class EfEntityRepositoryBase<TEntity, TContext> : IEntityRepository<TEn
tity>
        where TEntity : class, IEntity, new()
        where TContext : DbContext, new()
    {
        public List<TEntity> GetAll(Expression<Func<TEntity, bool>> filter = null)
```

```
        {
            using (TContext context = new TContext())
            {
                return filter == null
                    ? context.Set<TEntity>().ToList()
                    : context.Set<TEntity>().Where(filter).ToList();
            }
        }

        public TEntity Get(Expression<Func<TEntity, bool>> filter)
        {
            using (TContext context = new TContext())
            {
                return context.Set<TEntity>().SingleOrDefault(filter);
            }
        }

        public void Add(TEntity entity)
        {
            using (TContext context = new TContext())
            {

                var addedEntity = context.Entry(entity);
                addedEntity.State = EntityState.Added;
                context.SaveChanges();
            }
        }

        public void Update(TEntity entity)
        {
            using (TContext context = new TContext())
            {
                var updatedEntity = context.Entry(entity);
                updatedEntity.State = EntityState.Modified;
                context.SaveChanges();
            }
        }

        public void Delete(TEntity entity)
        {
            using (TContext context = new TContext())
            {
                var deletedEntity = context.Entry(entity);
                deletedEntity.State = EntityState.Deleted;
                context.SaveChanges();
            }
        }
    }
}
```

EfEntityRepositoryBase.cs

Veritabanın için uygulacak olan CRUD operasyonların gerçekleştirileceği kod parçasıdır. Bu projede veritabanı işlemleri için Entity Framework kullacağız.

## 4.3 Utilities

### 4.3.1 Results

Bu kısımda basit tiplede geriye veri döndürmek yerine bize lazım olacak birkaç özellik katarak veri göndürmek için yapılar oluşturulduğu alandır.

```
namespace Core.Utilities.Results
{
    public interface IResult
    {
        bool Success { get; }
        string Message { get; }
    }
}
```

IResult.cs

Bu interface void dönen verilerin parent'I olacaktır.

```
namespace Core.Utilities.Results
{
    public interface IDataResult<T>:IResult
    {
        T Data { get; }
    }
}
```

IDataResult.cs

Bu interface void dışında bir veri dönen verilerin parent'I olacaktır. Aynı zamanda IResult interface'inin özelliklerini de taşımaktadır.

```
namespace Core.Utilities.Results
{
    public class Result:IResult
    {
        public Result(bool success, string message):this(success)
        {
            Message = message;
        }

        public Result(bool success)
        {
            Success = success;
        }

        public bool Success { get; }
        public string Message { get; }
    }
}
```

Result.cs

Result sınıfı, void dönecek verilerin temel sınıfı olacaktır. Bu alanda default constractor'lar tanımlayarak gerekli alanların doldurulması sağlanmıştır.

```
namespace Core.Utilities.Results
{
    public class DataResult<T> : Result,IDataResult<T>
    {

        public DataResult(T data,bool success, string message) : base(success, mes
sage)
```

```
        {
            Data = data;
        }

        public DataResult(T data,bool success) : base(success)
        {
            Data = data;
        }

        public T Data { get; }
    }
}
```

DataResult.cs

DataResult sınıfı, void dönmeyen verilerin temel sınıfı olacaktır. Bu alanda default constractor'lar tanımlayarak gerekli alanların doldurulması sağlanmıştır. Aynı zamanda Result sınıfınında bulunan yapılar bu sınıftada geçerlidir.

```
namespace Core.Utilities.Results
{
    public class SuccessResult:Result
    {
        public SuccessResult(string message) : base(true,message)
        {
        }

        public SuccessResult() : base(true)
        {
        }
    }
}
```

SuccessResult.cs

İşleri daha kolay hale getirmek için void dönen verilerde bulunan işlemin doğru bir şekilde işletildiğini belirten 'success' alanın direk 'true' ile işaretlenmesini sağlamaktadır ve bu doğrultuda default constractorlar oluşturulmuştur.

```
namespace Core.Utilities.Results
{
    public class ErrorResult:Result
    {
        public ErrorResult(string message) : base(false,message)
        {}
        public ErrorResult() : base(false)
        {}
    }
}
```

ErrorResult.cs

İşleri daha kolay hale getirmek için void dönen verilerde bulunan işlemlerden birinde hata olutuğunda 'success' alanın direk 'false' ile işaretlenmesini sağlamaktadır ve bu doğrultuda default constractorlar oluşturulmuştur.

```
namespace Core.Utilities.Results
{
    public class SuccessDataResult<T>:DataResult<T>
    {
        public SuccessDataResult(T data, string message) : base(data, true, messag
e)
        {
        }
```

```
        public SuccessDataResult(T data) : base(data, true)
        {
        }

        public SuccessDataResult(string message) : base(default, true,message)
        {
        }

        public SuccessDataResult(IDataResult<Entities.Concrete.User> userToCheck)
: base(default, true)
        {
        }
    }
}
```

SuccessDataResult.cs

İşleri daha kolay hale getirmek için void dönmeyen verilerde bulunan işlemin doğru bir şekilde işletildiğini belirten 'success' alanın direk 'true' ile işaretlenmesini sağlamaktadır ve bu doğrultuda default constractorlar oluşturulmuştur.

```
namespace Core.Utilities.Results
{
    public class ErrorDataResult<T>:DataResult<T>
    {
        public ErrorDataResult(T data, string message) : base(data, false, message
)
        {
        }

        public ErrorDataResult(T data) : base(data, false)
        {
        }

        public ErrorDataResult(string message) : base(default, false,message)
        {
        }

        public ErrorDataResult() : base(default, false)
        {
        }
    }
}
```

ErrorDataResult.cs

İşleri daha kolay hale getirmek için void dönmeyen verilerde bulunan işlemin hatalı bir şekilde işletildiğini belirten 'success' alanın direk 'false' ile işaretlenmesini sağlamaktadır ve bu doğrultuda default constractorlar oluşturulmuştur.

### 4.3.2 Interceptors

Interceptor'un kelime anlamı araya giren diyebiliriz. Bizim kodlarımızı daha temiz ve düzenli yazabilmemiz için interception kullanacağız. Interception bize bir yapıdan önce, sonra, başarılı veya başarısız olma durumlarında çalışacak bir mekanizma sağlayacaktır.

```
using System;
using System.Linq;
using System.Reflection;
using Castle.DynamicProxy;
```

```
namespace Core.Utilities.Interceptors
{

    public class AspectInterceptorSelector : IInterceptorSelector
    {
        public IInterceptor[] SelectInterceptors(Type type, MethodInfo method, IIn
terceptor[] interceptors)
        {
            var classAttributes = type.GetCustomAttributes<MethodInterceptionBaseA
ttribute>
                (true).ToList();
            var methodAttributes = type.GetMethod(method.Name)
                .GetCustomAttributes<MethodInterceptionBaseAttribute>(true);
            classAttributes.AddRange(methodAttributes);

            return classAttributes.OrderBy(x => x.Priority).ToArray();
        }
    }

}
```

AspectInterceptorSelector.cs

AspectInterceptorSelector method ve class'lar üzerinde bulunan attribute'leri okuyarak bir sıraya sokmayı sağlar ve bir çalışma sırasına göre sıralar bunuda Priority ile gerçekleştirir.

```
using System;
using Castle.DynamicProxy;
namespace Core.Utilities.Interceptors
{
    public abstract class MethodInterception : MethodInterceptionBaseAttribute
    {
        protected virtual void OnBefore(IInvocation invocation) { }
        protected virtual void OnAfter(IInvocation invocation) { }
        protected virtual void OnException(IInvocation invocation, System.Exceptio
n e) { }
        protected virtual void OnSuccess(IInvocation invocation) { }
        public override void Intercept(IInvocation invocation)
        {
            var isSuccess = true;
            OnBefore(invocation);
            try
            {
                invocation.Proceed();
            }
            catch (Exception e)
            {
                isSuccess = false;
                OnException(invocation, e);
                throw;
            }
            finally
            {
                if (isSuccess)
                {
                    OnSuccess(invocation);
                }
            }
            OnAfter(invocation);
        }}}
```

MethodInterception.cs

MethodInterception sayesinde, gelen endpoint isteğinin ne zaman çalışacağını belirlediğimiz bir mekanizma oluşruyor ve ayriyeten try catch yapsını tek bir noktadan tüm yapılara uygulamızada olanak sağlamaktadır.

```
using System;
using Castle.DynamicProxy;


namespace Core.Utilities.Interceptors
{

    [AttributeUsage(AttributeTargets.Class | AttributeTargets.Method, AllowMultipl
e = true, Inherited = true)]
    public abstract class MethodInterceptionBaseAttribute : Attribute, IIntercepto
r
    {
        public int Priority { get; set; }

        public virtual void Intercept(IInvocation invocation)
        {

        }
    }

}
```

MethodInterceptionBaseAttribute.cs

MethodInterceptionBaseAttribute'da Attribute ile işlerimizi kolaylaştıracağız bu sayede örn. [Validation],[CacheAdd] vb. yapılar kurarak kod karmaşıklığını önlemiz olacağız.

### 4.3.3 IoC

```
using Microsoft.Extensions.DependencyInjection;

namespace Core.Utilities.IoC
{
    public interface ICoreModule
    {
        void Load(IServiceCollection serviceCollection);
    }
}
```

ICoreModule.cs

```
using Microsoft.Extensions.DependencyInjection;
using System;

namespace Core.Utilities.IoC
{
    public static class ServiceTool
    {
        public static IServiceProvider ServiceProvider { get; private set; }

        public static IServiceCollection Create(IServiceCollection services)
        {
            ServiceProvider = services.BuildServiceProvider();
            return services;
        }
    }
}
```

ServiceTool.cs

### 4.3.4 Security

#### 4.3.4.1 Encyption

```
using System.Text;
using Microsoft.IdentityModel.Tokens;

namespace Core.Utilities.Security.Encryption
{
    public class SecurityKeyHelper
    {
        public static SecurityKey CreateSecurityKey(string securityKey)
        {
            return new SymmetricSecurityKey(Encoding.UTF8.GetBytes(securityKey));
        }
    }
}
```

SecurityKeyHelper.cs

```
using Microsoft.IdentityModel.Tokens;

namespace Core.Utilities.Security.Encryption
{
    public class SigningCredentialsHelper
    {
        public static SigningCredentials CreateSigningCredentials(SecurityKey secu
rityKey)
        {
            return new SigningCredentials(securityKey,SecurityAlgorithms.HmacSha51
2Signature);
        }
    }
}
```

SigningCredentialsHelper.cs

#### 4.3.4.2 Hashing

```
using System.Text;

namespace Core.Utilities.Security.Hashing
{
    public class HashingHelper
    {

        public static void CreatePasswordHash(string password,out byte[] passwordH
ash,out byte[] passwordSalt)
        {
            using (var hmac = new System.Security.Cryptography.HMACSHA512())
            {
                passwordSalt = hmac.Key;
                passwordHash = hmac.ComputeHash(Encoding.UTF8.GetBytes(password));
            }
        }

        public static bool VerifyPasswordHash(string password,byte[] passwordHash,
byte[] passwordSalt)
        {
            using (var hmac = new System.Security.Cryptography.HMACSHA512(password
Salt))
            {
```

```
                var computedHash = hmac.ComputeHash(Encoding.UTF8.GetBytes(passwor
d));
                for (int i = 0; i < computedHash.Length; i++)
                {
                    if (computedHash[i]!=passwordHash[i])
                    {
                        return false;
                    }
                }
            }

            return true;
        }
    }
}
```

HashingHelper.cs

HashingHelper gelen string değerin HMACSHA512 standartlarında hashlenmesini sağlamaktadır.

#### 4.3.4.3 Jwt

```
using System;

namespace Core.Utilities.Security.JWT
{
    public class AccessToken
    {
        public string Token { get; set; }
        public DateTime Expiration { get; set; }
    }
}
```

AccessToken.cs

AccessToken, token oluşturulduğunda endpointden geri gönecek verinin modelini temsil etmektedir.

```
using System.Collections.Generic;
using Core.Entities.Concrete;

namespace Core.Utilities.Security.JWT
{
    public interface ITokenHelper
    {
        AccessToken CreateToken(User user, List<OperationClaim> operationClaims);
    }
}
```

ITokenHelper.cs

ITokenHelper token oluşturma işlemi için bir klavuz oluşrulmasını sağmaktadır.

```
using Core.Entities.Concrete;
using Core.Utilities.Security.Encryption;
using Microsoft.Extensions.Configuration;
using Microsoft.IdentityModel.Tokens;
using System;
using System.Collections.Generic;
using System.IdentityModel.Tokens.Jwt;
using System.Linq;
using System.Security.Claims;
using Core.Extensions;

namespace Core.Utilities.Security.JWT
```

```
{
    public class JwtHelper : ITokenHelper
    {
        public IConfiguration Configuration { get; }
        private TokenOptions _tokenOptions;
        private DateTime _accessTokenExpiration;
        public JwtHelper(IConfiguration configuration)
        {
            Configuration = configuration;
            _tokenOptions = Configuration.GetSection("TokenOptions").Get<TokenOpti
ons>();

        }
        public AccessToken CreateToken(User user, List<OperationClaim> operationCl
aims)
        {
            _accessTokenExpiration = DateTime.Now.AddMinutes(_tokenOptions.AccessT
okenExpiration);
            var securityKey = SecurityKeyHelper.CreateSecurityKey(_tokenOptions.Se
curityKey);
            var signingCredentials = SigningCredentialsHelper.CreateSigningCredent
ials(securityKey);
            var jwt = CreateJwtSecurityToken(_tokenOptions, user, signingCredentia
ls, operationClaims);
            var jwtSecurityTokenHandler = new JwtSecurityTokenHandler();
            var token = jwtSecurityTokenHandler.WriteToken(jwt);

            return new AccessToken
            {
                Token = token,
                Expiration = _accessTokenExpiration
            };

        }

        public JwtSecurityToken CreateJwtSecurityToken(TokenOptions tokenOptions,
User user,
            SigningCredentials signingCredentials, List<OperationClaim> operationC
laims)
        {
            var jwt = new JwtSecurityToken(
                issuer: tokenOptions.Issuer,
                audience: tokenOptions.Audience,
                expires: _accessTokenExpiration,
                notBefore: DateTime.Now,
                claims: SetClaims(user, operationClaims),
                signingCredentials: signingCredentials
            );
            return jwt;
        }

        private IEnumerable<Claim> SetClaims(User user, List<OperationClaim> opera
tionClaims)
        {
            var claims = new List<Claim>();
            claims.AddNameIdentifier(user.Id.ToString());
            claims.AddEmail(user.Email);
            claims.AddName($"{user.FirstName} {user.LastName}");
            claims.AddRoles(operationClaims.Select(c => c.Name).ToArray());
```

```
            return claims;
        }
    }
}
```

JwtHelper.cs

JwtHelper ile token oluşturma işlemi gerçekleştirilmektedir.

```
namespace Core.Utilities.Security.JWT
{
    public class TokenOptions
    {
        public string Audience { get; set; }
        public string Issuer { get; set; }
        public int AccessTokenExpiration { get; set; }
        public string SecurityKey { get; set; }
    }
}
```

TokenOptions.cs

TokenOptions appsettings.json dosyasında tutulan verilerin sistem içerinde kullanabilmek için bir modeldir.

### 4.4 Extensions

```
using System.Collections.Generic;
using System.IdentityModel.Tokens.Jwt;
using System.Linq;
using System.Security.Claims;

namespace Core.Extensions
{
    public static class ClaimExtensions
    {
        public static void AddEmail(this ICollection<Claim> claims, string email)
        {
            claims.Add(new Claim(JwtRegisteredClaimNames.Email, email));
        }

        public static void AddName(this ICollection<Claim> claims, string name)
        {
            claims.Add(new Claim(ClaimTypes.Name, name));
        }

        public static void AddNameIdentifier(this ICollection<Claim> claims, strin
g nameIdentifier)
        {
            claims.Add(new Claim(ClaimTypes.NameIdentifier, nameIdentifier));
        }

        public static void AddRoles(this ICollection<Claim> claims, string[] roles
)
        {
            roles.ToList().ForEach(role => claims.Add(new Claim(ClaimTypes.Role, r
ole)));
        }
    }
}
```

ClaimExtensions.cs

```
using System.Collections.Generic;
using System.Linq;
using System.Security.Claims;

namespace Core.Extensions
{
    public static class ClaimsPrincipalExtensions
    {
        public static List<string> Claims(this ClaimsPrincipal claimsPrincipal, st
ring claimType)
        {
            var result = claimsPrincipal?.FindAll(claimType)?.Select(x => x.Value)
.ToList();
            return result;
        }

        public static List<string> ClaimRoles(this ClaimsPrincipal claimsPrincipal
)
        {
            return claimsPrincipal?.Claims(ClaimTypes.Role);
        }
    }
}
```

ClaimsPrincipalExtensions.cs

```
using System.Collections.Generic;
using FluentValidation.Results;
using Newtonsoft.Json;

namespace Core.Extensions
{
    public class ErrorDetails
    {
        public string Message { get; set; }
        public int StatusCode { get; set; }


        public override string ToString()
        {
            return JsonConvert.SerializeObject(this);
        }
    }

    public class ValidationErrorDetails : ErrorDetails
    {
        public IEnumerable<ValidationFailure> Errors { get; set; }
    }
}
```

ErrorDetails.cs

```
using FluentValidation;
using FluentValidation.Results;
using Microsoft.AspNetCore.Http;
using System;
using System.Collections.Generic;
using System.Net;
using System.Threading.Tasks;

namespace Core.Extensions
{
```

```
    public class ExceptionMiddleware
    {
        private RequestDelegate _next;

        public ExceptionMiddleware(RequestDelegate next)
        {
            _next = next;
        }

        public async Task InvokeAsync(HttpContext httpContext)
        {
            try
            {
                await _next(httpContext);
            }
            catch (Exception e)
            {
                await HandleExceptionAsync(httpContext, e);
            }
        }

        private Task HandleExceptionAsync(HttpContext httpContext, Exception e)
        {
            httpContext.Response.ContentType = "application/json";
            httpContext.Response.StatusCode = (int)HttpStatusCode.InternalServerEr
ror;

            string message = "Internal Server Error";

            IEnumerable<ValidationFailure> errors;
            if (e.GetType() == typeof(ValidationException))
            {
                message = e.Message;
                errors = ((ValidationException)e).Errors;
                httpContext.Response.StatusCode = 400;

                return httpContext.Response.WriteAsync(new ValidationErrorDetails(
)
                {
                    StatusCode = 400,
                    Message = message,
                    Errors = errors
                }.ToString());
            }

            return httpContext.Response.WriteAsync(new ErrorDetails
            {
                StatusCode = httpContext.Response.StatusCode,
                Message = message
            }.ToString());
        }
    }
}
```

ExceptionMiddleware.cs

```
using Microsoft.AspNetCore.Builder;

namespace Core.Extensions
{
    public static class ExceptionMiddlewareExtensions
    {
        public static void ConfigureCustomExceptionMiddleware(this IApplicationBui
lder app)
        {
            app.UseMiddleware<ExceptionMiddleware>();
        }
    }
}
```

ExceptionMiddlewareExtensions.cs

```
using Core.Utilities.IoC;
using Microsoft.Extensions.DependencyInjection;

namespace Core.Extensions
{
    public static class ServiceCollectionExtensions
    {
        public static IServiceCollection AddDependencyResolvers(this IServiceColle
ction serviceCollection,ICoreModule[] modules)
        {
            foreach (var module in modules)
            {
                module.Load(serviceCollection);
            }

            return ServiceTool.Create(serviceCollection);
        }
    }
}
```

ServiceCollectionExtensions.cs

## 4.5 Cross Cutting Concerns

### 4.5.1 Caching

```
namespace Core.CrossCuttingConcerns.Caching
{
    public interface ICacheManager
    {
        object Get(string key);
        T Get<T>(string key);
        void Add(string key, object value, int duration);
        bool IsAdd(string key);
        void Remove(string key);
        void RemoveByPattern(string pattern);
    }
}
```

ICacheManager.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text.RegularExpressions;
using Core.Utilities.IoC;
using Microsoft.Extensions.Caching.Memory;
```

```
using Microsoft.Extensions.DependencyInjection;

namespace Core.CrossCuttingConcerns.Caching.Microsoft
{
    public class MemoryCacheManager:ICacheManager
    {
        private IMemoryCache _memoryCache;

        public MemoryCacheManager()
        {
            _memoryCache = ServiceTool.ServiceProvider.GetService<IMemoryCache>();
        }

        public object Get(string key)
        {
            return _memoryCache.Get(key);
        }

        public T Get<T>(string key)
        {
            return _memoryCache.Get<T>(key);
        }

        public void Add(string key, object value, int duration)
        {
            _memoryCache.Set(key, value, TimeSpan.FromMinutes(duration));
        }

        public bool IsAdd(string key)
        {
            return _memoryCache.TryGetValue(key,out _);
        }

        public void Remove(string key)
        {
            _memoryCache.Remove(key);
        }

        public void RemoveByPattern(string pattern)
        {
            var cacheEntriesCollectionDefinition = typeof(MemoryCache).GetProperty
("EntriesCollection", System.Reflection.BindingFlags.NonPublic | System.Reflection
.BindingFlags.Instance);
            var cacheEntriesCollection = cacheEntriesCollectionDefinition.GetValue
(_memoryCache) as dynamic;
            List<ICacheEntry> cacheCollectionValues = new List<ICacheEntry>();

            foreach (var cacheItem in cacheEntriesCollection)
            {
                ICacheEntry cacheItemValue = cacheItem.GetType().GetProperty("Valu
e").GetValue(cacheItem, null);
                cacheCollectionValues.Add(cacheItemValue);
            }

            var regex = new Regex(pattern, RegexOptions.Singleline | RegexOptions.
Compiled | RegexOptions.IgnoreCase);
            var keysToRemove = cacheCollectionValues.Where(d => regex.IsMatch(d.Ke
y.ToString())).Select(d => d.Key).ToList();
```

```
            foreach (var key in keysToRemove)
            {
                _memoryCache.Remove(key);
            }
        }
    }
}
```

MemoryCacheManager.cs

### 4.5.2 Validation

```
using FluentValidation;

namespace Core.CrossCuttingConcerns.Validation
{
    public static class ValidationTool
    {

        public static void Validate(IValidator validator,object entity)
        {
            var context = new ValidationContext<object>(entity);
            var result = validator.Validate(context);
            if (!result.IsValid)
            {
                throw new ValidationException(result.Errors);
            }

        }
    }
}
```

ValidationTool.cs

## 4.6 – Aspects

### 4.6.1 – Autofac – Caching

```
using System.Linq;
using Castle.DynamicProxy;
using Core.CrossCuttingConcerns.Caching;
using Core.Utilities.Interceptors;
using Core.Utilities.IoC;
using Microsoft.Extensions.DependencyInjection;

namespace Core.Aspects.Autofac.Caching
{
    public class CacheAspect : MethodInterception
    {
        private int _duration;
        private ICacheManager _cacheManager;

        public CacheAspect(int duration = 60)
        {
            _duration = duration;
            _cacheManager = ServiceTool.ServiceProvider.GetService<ICacheManager>(
);
        }

        public override void Intercept(IInvocation invocation)
        {
```

```
            var methodName = string.Format($"{invocation.Method.ReflectedType.Full
Name}.{invocation.Method.Name}");
            var arguments = invocation.Arguments.ToList();
            var key = $"{methodName}({string.Join(",", arguments.Select(x => x?.To
String() ?? "<Null>"))})";
            if (_cacheManager.IsAdd(key))
            {
                invocation.ReturnValue = _cacheManager.Get(key);
                return;
            }
            invocation.Proceed();
            _cacheManager.Add(key, invocation.ReturnValue, _duration);
        }
    }
}
```

CacheAspect.cs

```
using Castle.DynamicProxy;
using Core.CrossCuttingConcerns.Caching;
using Core.Utilities.Interceptors;
using Core.Utilities.IoC;
using Microsoft.Extensions.DependencyInjection;

namespace Core.Aspects.Autofac.Caching
{
    public class CacheRemoveAspect : MethodInterception
    {
        private string _pattern;
        private ICacheManager _cacheManager;

        public CacheRemoveAspect(string pattern)
        {
            _pattern = pattern;
            _cacheManager = ServiceTool.ServiceProvider.GetService<ICacheManager>(
);
        }

        protected override void OnSuccess(IInvocation invocation)
        {
            _cacheManager.RemoveByPattern(_pattern);
        }
    }
}
```

CacheRemoveAspect.cs

### 4.6.2– Aspects – Validation

```
using System;
using System.Linq;
using Castle.DynamicProxy;
using Core.CrossCuttingConcerns.Validation;
using Core.Utilities.Interceptors;
using FluentValidation;

namespace Core.Aspects.Autofac.Validation
{
    public class ValidationAspect : MethodInterception
    {
        private Type _validatorType;
        public ValidationAspect(Type validatorType)
```

```
        {
            if (!typeof(IValidator).IsAssignableFrom(validatorType))
            {
                throw new System.Exception("Bu bir doğrulama sınıfı değil");
            }

            _validatorType = validatorType;
        }
        protected override void OnBefore(IInvocation invocation)
        {
            var validator = (IValidator)Activator.CreateInstance(_validatorType);
            var entityType = _validatorType.BaseType.GetGenericArguments()[0];
            var entities = invocation.Arguments.Where(t => t.GetType() == entityTy
pe);
            foreach (var entity in entities)
            {
                ValidationTool.Validate(validator, entity);
            }
        }
    }
}
```

ValidationAspect.cs

### 4.7 Dependence Resolvers

```
using System.Diagnostics;
using Core.CrossCuttingConcerns.Caching;
using Core.CrossCuttingConcerns.Caching.Microsoft;
using Core.Utilities.IoC;
using Microsoft.AspNetCore.Http;
using Microsoft.Extensions.DependencyInjection;

namespace Core.DependencyResolvers
{
    public class CoreModule : ICoreModule
    {
        public void Load(IServiceCollection serviceCollection)
        {
            serviceCollection.AddMemoryCache();
            serviceCollection.AddSingleton<IHttpContextAccessor, HttpContextAccess
or>();
            serviceCollection.AddSingleton<ICacheManager, MemoryCacheManager>();
            serviceCollection.AddSingleton<Stopwatch>();
        }
    }
}
```

CoreModule.cs

# 5 – Entities

## 5.1 – Concrete

```
namespace Entities.Concrete
{
    public class DataModel
    {
        public ItData ItData { get; set; }
        public TrData TrData { get; set; }
    }
}
```

DataModel.cs

```
using Core.Entities;

namespace Entities.Concrete
{
    public class ItData: IEntity
    {
        public int Id { get; set; }
        public string dc_Orario { get; set; }
        public string dc_Categoria { get; set; }
        public string dc_Evento { get; set; }
    }
}
```

ItData.cs

```
using Microsoft.AspNetCore.Http;

namespace Entities.Concrete
{
    public class JsonData
    {
        public IFormFile TrDatas { get; set; }
        public IFormFile ItDatas { get; set; }
    }
}
```

JsonData.cs

```
using Core.Entities;

namespace Entities.Concrete
{
    public class TrData : IEntity
    {
        public int Id { get; set; }
        public string dc_Zaman { get; set; }
        public string dc_Kategori { get; set; }
        public string dc_Olay { get; set; }
    }
}
```

TrData.cs

## 5.2 – Dtos

```
using Core.Entities;

namespace Entities.DTOs
{
```

```
    public class UserForLoginDto : IDto
    {
        public string Email { get; set; }
        public string Password { get; set; }
    }
}
```

UserForLoginDto.cs

```
using Core.Entities;

namespace Entities.DTOs
{
    public class UserForRegisterDto : IDto
    {
        public string Email { get; set; }
        public string Password { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }
    }
}
```

UserForRegisterDto.cs

# 6 – Data Access Katmanı

## 6.1 – Abstract

```
using System.Collections.Generic;
using Core.DataAccess;
using Core.Entities.Concrete;

namespace DataAccess.Abstract
{
    public interface IUserDal : IEntityRepository<User>
    {
        List<OperationClaim> GetClaims(User user);
    }
}
```

IUserDal.cs

Userlar için yapılacak işlemlerin klavuzunu oluşturduğumuz  interfacetir. IEntityRepository sayesinde CRUD operasyonlarıdan bu yapıya dahil oldu.

```
using Core.DataAccess;
using Entities.Concrete;

namespace DataAccess.Abstract
{
    public interface ITrDataDal : IEntityRepository<TrData>
    {

    }
}
```

ITrDataDal.cs

Türkçe veriler için yapılacak işlemlerin klavuzunu oluşturduğumuz  interfacetir. IEntityRepository sayesinde CRUD operasyonlarıdan bu yapıya dahil oldu.

```
using Core.DataAccess;
using Entities.Concrete;

namespace DataAccess.Abstract
{
    public interface IItDataDal : IEntityRepository<ItData>
    {

    }
}
```

IItDataDal.cs

İtelyanca veriler için yapılacak işlemlerin klavuzunu oluşturduğumuz  interfacetir. IEntityRepository sayesinde CRUD operasyonlarıdan bu yapıya dahil oldu.

### 6.1 – Concrete

```
using Core.Entities.Concrete;
using Entities.Concrete;
using Microsoft.EntityFrameworkCore;

namespace DataAccess.Concrete.EntityFramework
{
    public class DataContext : DbContext
    {
        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuild
er)
        {

            optionsBuilder.UseSqlServer(@"Data Source=(localdb)\MSSQLLocalDB;Initi
al Catalog=databaseName;Integrated Security=True;Connect Timeout=30;Encrypt=False;
TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False
");

        }

        public DbSet<OperationClaim> OperationClaims { get; set; }
        public DbSet<User> Users { get; set; }
        public DbSet<UserOperationClaim> UserOperationClaims { get; set; }
        public DbSet<ItData> ItDatas { get; set; }
        public DbSet<TrData> TrDatas { get; set; }
    }
}
```

DataContext.cs

Veritabanı connection string'inin tutulduğu ve model ve veritananı tablolarının eşleştirildiği alandır.

```
using Core.DataAccess.EntityFramework;
using DataAccess.Abstract;
using Entities.Concrete;

namespace DataAccess.Concrete.EntityFramework
{
    public class EfItDataDal: EfEntityRepositoryBase<ItData, DataContext>, IItData
Dal
    {

    }
}
```

EfItDataDal.cs

Oluruduğumuz klavuzlar sayesinde bu operasyonları doldurduğumuz alandır ama biz zaten EfEntityReposiyoryBase sayesinde CRUD operasyonlarını otomatik olarak yazmış oluyoruz. Extra bir klavuz eklersek bunu burada doldurmamız yeterli.

```
using Core.DataAccess.EntityFramework;
using DataAccess.Abstract;
using Entities.Concrete;
namespace DataAccess.Concrete.EntityFramework
{
    public class EfTrDataDal: EfEntityRepositoryBase<TrData, DataContext>, ITrData
Dal{}}
```

EfTrDataDal.cs

Oluruduğumuz klavuzlar sayesinde bu operasyonları doldurduğumuz alandır ama biz zaten EfEntityReposiyoryBase sayesinde CRUD operasyonlarını otomatik olarak yazmış oluyoruz. Extra bir klavuz eklersek bunu burada doldurmamız yeterli.

```
using Core.DataAccess.EntityFramework;
using Core.Entities.Concrete;
using DataAccess.Abstract;
using System.Collections.Generic;
using System.Linq;

namespace DataAccess.Concrete.EntityFramework
{
    public class EfUserDal : EfEntityRepositoryBase<User, DataContext>, IUserDal
    {
        public List<OperationClaim> GetClaims(User user)
        {
            using (var context = new DataContext())
            {
                var result = from operationClaim in context.OperationClaims
                             join userOperationClaim in context.UserOperationClaim
s
                                 on operationClaim.Id equals userOperationClaim.Op
erationClaimId
                             where userOperationClaim.UserId == user.Id
                             select new OperationClaim { Id = operationClaim.Id, N
ame = operationClaim.Name };
                return result.ToList();

            }
        }
    }
}
```

EfUserDal.cs

Oluşturduğumuz klavuzlar sayesinde bu operasyonları doldurduğumuz alandır ama biz zaten EfEntityReposiyoryBase sayesinde CRUD operasyonlarını otomatik olarak yazmış oluyoruz. Extra olarak GetClaims methodunu yazdığımız için bu method'u bu alanda içerini dolduryoruz.

# 7 – Business Katmanı

## 7.1 – Abstract

Bu alanda yapacağımız işlemdirin klavuzunu oluşturacağız çoğunlukla her model için ayarı ayrı yapılar oluşturulur. Bunları illaki veritabanı işlemleri içermesi gerekmez.

```
using Core.Entities.Concrete;
using Core.Utilities.Results;
using Core.Utilities.Security.JWT;
using Entities.DTOs;

namespace Business.Abstract
{
    public interface IAuthService
    {

        IDataResult<User> Register(UserForRegisterDto userForRegisterDto, string p
assword);
        IDataResult<User> Login(UserForLoginDto userForLoginDto);
        IResult UserExists(string email);
        IDataResult<AccessToken> CreateAccessToken(User user);
    }
}
```

IAuthService.cs

```
using System.Collections.Generic;
using Core.Utilities.Results;
using Entities.Concrete;

namespace Business.Abstract
{
    public interface IItDataService
    {
        IDataResult<List<ItData>> GetAll();
        IDataResult<ItData> GetbyId(int id);
        IResult Add(ItData data);
        IResult AddAll(List<ItData> datas);
        IResult Delete(ItData data);
        IResult Update(ItData data);
    }
}
```

IItDataService.cs

```
using System.Collections.Generic;
using System.Threading.Tasks;
using Core.Utilities.Results;
using Entities.Concrete;
using Microsoft.AspNetCore.Http;

namespace Business.Abstract
{
    public interface IJsonService
    {
        Task<IDataResult<List<TrData>>> SaveToDatabaseTrDatas(IFormFile jsonData);
        Task<IDataResult<List<ItData>>> SaveToDatabaseItDatas(IFormFile jsonData);
    }
}
```

IJsonService.cs

```
using System.Collections.Generic;
using Core.Utilities.Results;
using Entities.Concrete;

namespace Business.Abstract
{
    public interface ITrDataService
    {
        IDataResult<List<TrData>> GetAll();
        IDataResult<TrData> GetbyId(int id);
        IResult Add(TrData data);
        IResult AddAll(List<TrData> datas);
        IResult Delete(TrData data);
        IResult Update(TrData data);
    }
}
```

ITrDataService.cs

```
using Core.Entities.Concrete;
using System.Collections.Generic;

namespace Business.Abstract
{
    public interface IUserService
    {
        List<OperationClaim> GetClaims(User user);
        void Add(User user);
        User GetByMail(string email);
    }
}
```

IUserService.cs

### 7.2 – Concrete

Klavuzların içerilerin doldurulduldu alandır.

```
using Business.Abstract;
using Business.Constants;
using Core.Entities.Concrete;
using Core.Utilities.Results;
using Core.Utilities.Security.Hashing;
using Core.Utilities.Security.JWT;
using Entities.DTOs;

namespace Business.Concrete
{
    public class AuthManager : IAuthService
    {
        private IUserService _userService;
        private ITokenHelper _tokenHelper;

        public AuthManager(IUserService userService, ITokenHelper tokenHelper)
        {
            _userService = userService;
            _tokenHelper = tokenHelper;
        }

        public IDataResult<User> Register(UserForRegisterDto userForRegisterDto, s
tring password)
        {
```

```
            byte[] passwordHash, passwordSalt;
            HashingHelper.CreatePasswordHash(password, out passwordHash, out passw
ordSalt);
            var user = new User
            {
                Email = userForRegisterDto.Email,
                FirstName = userForRegisterDto.FirstName,
                LastName = userForRegisterDto.LastName,
                PasswordHash = passwordHash,
                PasswordSalt = passwordSalt,
                Status = true
            };
            _userService.Add(user);
            return new SuccessDataResult<User>(user, Messages.UserRegistered);
        }

        public IDataResult<User> Login(UserForLoginDto userForLoginDto)
        {
            var userToCheck = _userService.GetByMail(userForLoginDto.Email);
            if (userToCheck == null)
            {
                return new ErrorDataResult<User>(Messages.UserNotFound);
            }

            if (!HashingHelper.VerifyPasswordHash(userForLoginDto.Password, userTo
Check.PasswordHash, userToCheck.PasswordSalt))
            {
                return new ErrorDataResult<User>(Messages.PasswordError);
            }

            return new SuccessDataResult<User>(userToCheck, Messages.SuccessfulLog
in);
        }

        public IResult UserExists(string email)
        {
            if (_userService.GetByMail(email) != null)
            {
                return new ErrorResult(Messages.UserAlreadyExists);
            }
            return new SuccessResult();
        }

        public IDataResult<AccessToken> CreateAccessToken(User user)
        {
            var claims = _userService.GetClaims(user);
            var accessToken = _tokenHelper.CreateToken(user, claims);
            return new SuccessDataResult<AccessToken>(accessToken, Messages.Access
TokenCreated);
        }
    }
}
```

AuthManager.cs

```
using Business.Abstract;
using Business.ValidationRules.FluentValidation;
using Core.Aspects.Autofac.Caching;
using Core.Aspects.Autofac.Validation;
using Core.Utilities.Results;
using DataAccess.Abstract;
```

```
using Entities.Concrete;
using System.Collections.Generic;
using System.Linq;
using Business.BusinessAspects.Autofac;

namespace Business.Concrete
{
    [SecuredOperation("user")]
    public class ItDataManger : IItDataService
    {
        private IItDataDal _itDataDal;

        public ItDataManger(IItDataDal itDataDal)
        {
            _itDataDal = itDataDal;
        }

        [CacheAspect]
        public IDataResult<List<ItData>> GetAll()
        {
            return new SuccessDataResult<List<ItData>>(_itDataDal.GetAll());
        }

        [CacheAspect]
        public IDataResult<ItData> GetbyId(int id)
        {
            return new SuccessDataResult<ItData>(_itDataDal.Get(x => x.Id == id));
        }

        [CacheRemoveAspect("IItDataService.Get")]
        [ValidationAspect(typeof(ItDataModelAddValidator))]
        public IResult Add(ItData data)
        {
            var lastData = _itDataDal.GetAll().OrderByDescending(x => x.Id).FirstO
rDefault();
            data.Id = lastData.Id + 1;
            _itDataDal.Add(data);

            return new SuccessResult();
        }

        [CacheRemoveAspect("IItDataService.Get")]
        public IResult AddAll(List<ItData> datas)
        {
            foreach (var data in datas)
            {
                _itDataDal.Add(data);
            }

            return new SuccessResult();
        }

        [CacheRemoveAspect("IItDataService.Get")]
        [ValidationAspect(typeof(ItDataModelDeleteValidator))]
        public IResult Delete(ItData data)
        {
            _itDataDal.Delete(data);

            return new SuccessResult();
```

```
        }

        [CacheRemoveAspect("IProductService.Get")]
        [ValidationAspect(typeof(ItDataModelUpdateValidator))]
        public IResult Update(ItData data)
        {
            _itDataDal.Update(data);

            return new SuccessResult();
        }
    }
}
```

ItDataManger.cs

```
using Business.Abstract;
using Business.ValidationRules.FluentValidation;
using Core.Aspects.Autofac.Validation;
using Core.Utilities.Results;
using Entities.Concrete;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using Newtonsoft.Json;
using System.Collections.Generic;
using System.IO;
using System.Threading.Tasks;
using Business.BusinessAspects.Autofac;

namespace Business.Concrete
{
    [SecuredOperation("user")]
    [ValidationAspect(typeof(JsonValidator))]
    public class JsonManager : IJsonService
    {
        private readonly IHostingEnvironment _hostingEnvironment;

        public JsonManager(IHostingEnvironment hostingEnvironment)
        {
            _hostingEnvironment = hostingEnvironment;
        }


        public async Task<IDataResult<List<TrData>>> SaveToDatabaseTrDatas(IFormFi
le jsonData)
        {
            dynamic o;
            var serializer = new Newtonsoft.Json.JsonSerializer();
            string dir = _hostingEnvironment.WebRootPath;
            string fileName = "tr";
            string extension = Path.GetExtension(jsonData.FileName);
            fileName = fileName + extension;
            string savePath = Path.Combine(dir, fileName);


            List<TrData> datas = new List<TrData>();


            using (var stream = System.IO.File.Create(savePath))
            {
                await jsonData.CopyToAsync(stream);
            }
```

```
            using (FileStream s = System.IO.File.Open(savePath, FileMode.Open))
            using (StreamReader sr = new StreamReader(s))
            using (JsonReader reader = new JsonTextReader(sr))
            {
                while (reader.Read())
                {
                    if (reader.TokenType == JsonToken.StartObject)
                    {
                        TrData data = new TrData();
                        o = serializer.Deserialize(reader);
                        data.Id = o["ID"];
                        data.dc_Kategori = o["dc_Kategori"];
                        data.dc_Zaman = o["dc_Zaman"];
                        data.dc_Olay = o["dc_Olay"];
                        datas.Add(data);
                    }
                }
            }

            return new SuccessDataResult<List<TrData>>(datas);

        }

        public async Task<IDataResult<List<ItData>>> SaveToDatabaseItDatas(IFormFi
le jsonData)
        {
            dynamic o;
            var serializer = new Newtonsoft.Json.JsonSerializer();
            string dir = _hostingEnvironment.WebRootPath;
            string fileName = "it";
            string extension = Path.GetExtension(jsonData.FileName);
            fileName = fileName + extension;
            string savePath = Path.Combine(dir, fileName);


            List<ItData> datas = new List<ItData>();

            using (var stream = System.IO.File.Create(savePath))
            {
                await jsonData.CopyToAsync(stream);
            }
            using (FileStream s = System.IO.File.Open(savePath, FileMode.Open))
            using (StreamReader sr = new StreamReader(s))
            using (JsonReader reader = new JsonTextReader(sr))
            {
                while (reader.Read())
                {
                    if (reader.TokenType == JsonToken.StartObject)
                    {
                        ItData data = new ItData();
                        o = serializer.Deserialize(reader);
                        data.Id = o["ID"];
                        data.dc_Categoria = o["dc_Categoria"];
                        data.dc_Evento = o["dc_Evento"];
                        data.dc_Orario = o["dc_Orario"];
                        datas.Add(data);
                    }
                }
```

```
            }

            return new SuccessDataResult<List<ItData>>(datas);
        }
    }
}
```

JsonManager.cs

```
using Business.Abstract;
using Business.ValidationRules.FluentValidation;
using Core.Aspects.Autofac.Caching;
using Core.Aspects.Autofac.Validation;
using Core.Utilities.Results;
using DataAccess.Abstract;
using Entities.Concrete;
using System.Collections.Generic;
using System.Linq;
using Business.BusinessAspects.Autofac;

namespace Business.Concrete
{
    [SecuredOperation("user")]
    public class TrDataManger : ITrDataService
    {
        private ITrDataDal _trDataDal;

        public TrDataManger(ITrDataDal trDataDal)
        {
            _trDataDal = trDataDal;
        }

        [CacheAspect]
        public IDataResult<List<TrData>> GetAll()
        {
            return new SuccessDataResult<List<TrData>>(_trDataDal.GetAll());
        }

        [CacheAspect]
        public IDataResult<TrData> GetbyId(int id)
        {
            return new SuccessDataResult<TrData>(_trDataDal.Get(x => x.Id == id));
        }

        [CacheRemoveAspect("ITrDataService.Get")]
        [ValidationAspect(typeof(TrDataModelAddValidator))]
        public IResult Add(TrData data)
        {
            var lastData = _trDataDal.GetAll().OrderByDescending(x => x.Id).FirstO
rDefault();
            data.Id = lastData.Id + 1;
            _trDataDal.Add(data);

            return new SuccessResult();
        }

        [CacheRemoveAspect("ITrDataService.Get")]
        public IResult AddAll(List<TrData> datas)
        {
            foreach (var data in datas)
            {
```

```
                _trDataDal.Add(data);
            }

            return new SuccessResult();
        }

        [CacheRemoveAspect("ITrDataService.Get")]
        [ValidationAspect(typeof(TrDataModelDeleteValidator))]
        public IResult Delete(TrData data)
        {
            _trDataDal.Delete(data);

            return new SuccessResult();
        }

        [CacheRemoveAspect("ITrDataService.Get")]
        [ValidationAspect(typeof(TrDataModelUpdateValidator))]
        public IResult Update(TrData data)
        {
            _trDataDal.Update(data);

            return new SuccessResult();
        }
    }
}
```

TrDataManger.cs

```
using Business.Abstract;
using Core.Entities.Concrete;
using DataAccess.Abstract;
using System.Collections.Generic;

namespace Business.Concrete
{
    public class UserManager : IUserService
    {
        IUserDal _userDal;

        public UserManager(IUserDal userDal)
        {
            _userDal = userDal;
        }

        public List<OperationClaim> GetClaims(User user)
        {
            return new List<OperationClaim>(_userDal.GetClaims(user));
        }

        public void Add(User user)
        {
            _userDal.Add(user);
        }

        public User GetByMail(string email)
        {
            return _userDal.Get(u => u.Email == email);
        }
    }
}
```

UserManager.cs

### 7.3 – Business Aspects

```
using Business.Constants;
using Castle.DynamicProxy;
using Core.Extensions;
using Core.Utilities.Interceptors;
using Core.Utilities.IoC;
using Microsoft.AspNetCore.Http;
using Microsoft.Extensions.DependencyInjection;
using System;

namespace Business.BusinessAspects.Autofac
{
    public class SecuredOperation : MethodInterception
    {
        private string[] _roles;
        private IHttpContextAccessor _httpContextAccessor;

        public SecuredOperation(string roles)
        {
            _roles = roles.Split(',');
            _httpContextAccessor = ServiceTool.ServiceProvider.GetService<IHttpCon
textAccessor>();

        }

        protected override void OnBefore(IInvocation invocation)
        {
            var roleClaims = _httpContextAccessor.HttpContext.User.ClaimRoles();
            foreach (var role in _roles)
            {
                if (roleClaims.Contains(role))
                {
                    return;
                }
            }
            throw new Exception(Messages.AuthorizationDenied);
        }
    }
}
```

SecuredOperation.cs

### 7.4 – Validation Rules

```
using Entities.Concrete;
using FluentValidation;

namespace Business.ValidationRules.FluentValidation
{
    public class ItDataModelAddValidator : AbstractValidator<ItData>
    {
        public ItDataModelAddValidator()
        {
            RuleFor(p => p.dc_Categoria).NotEmpty();
            RuleFor(p => p.dc_Evento).NotEmpty();
            RuleFor(p => p.dc_Orario).NotEmpty();
        }
    }
}
```

ItDataModelAddValidator.cs

```
using Entities.Concrete;
using FluentValidation;

namespace Business.ValidationRules.FluentValidation
{
    public class ItDataModelDeleteValidator : AbstractValidator<ItData>
    {
        public ItDataModelDeleteValidator()
        {
            RuleFor(p => p.Id).NotEmpty();
        }
    }
}
```

ItDataModelDeleteValidator.cs

```
using Entities.Concrete;
using FluentValidation;

namespace Business.ValidationRules.FluentValidation
{
    public class ItDataModelUpdateValidator : AbstractValidator<ItData>
    {
        public ItDataModelUpdateValidator()
        {
            RuleFor(p => p.Id).NotEmpty();
            RuleFor(p => p.dc_Categoria).NotEmpty();
            RuleFor(p => p.dc_Orario).NotEmpty();
            RuleFor(p => p.dc_Evento).NotEmpty();

        }
    }
}
```

ItDataModelUpdateValidator.cs

```
using Entities.Concrete;
using FluentValidation;
using System.IO;

namespace Business.ValidationRules.FluentValidation
{
    public class JsonValidator : AbstractValidator<JsonData>
    {
        public JsonValidator()
        {
            RuleFor(p => p.ItDatas.Length).GreaterThan(0);
            RuleFor(p => Path.GetExtension(p.ItDatas.FileName) == ".json");
            RuleFor(p => p.TrDatas.Length).GreaterThan(0);
            RuleFor(p => Path.GetExtension(p.TrDatas.FileName) == ".json");
        }


    }
}
```

JsonValidator.cs

```
using Entities.Concrete;
using FluentValidation;

namespace Business.ValidationRules.FluentValidation
{
    public class TrDataModelAddValidator : AbstractValidator<TrData>
    {
        public TrDataModelAddValidator()
        {
            RuleFor(p => p.dc_Kategori).NotEmpty();
            RuleFor(p => p.dc_Olay).NotEmpty();
            RuleFor(p => p.dc_Zaman).NotEmpty();
        }
    }
}
```

TrDataModelAddValidator.cs

```
using Entities.Concrete;
using FluentValidation;

namespace Business.ValidationRules.FluentValidation
{
    public class TrDataModelDeleteValidator : AbstractValidator<TrData>
    {
        public TrDataModelDeleteValidator()
        {
            RuleFor(p => p.Id).NotEmpty();

        }
    }
}
```

TrDataModelDeleteValidator.cs

```
using Entities.Concrete;
using FluentValidation;

namespace Business.ValidationRules.FluentValidation
{
    public class TrDataModelUpdateValidator : AbstractValidator<TrData>
    {
        public TrDataModelUpdateValidator()
        {
            RuleFor(p => p.Id).NotEmpty();
            RuleFor(p => p.dc_Kategori).NotEmpty();
            RuleFor(p => p.dc_Olay).NotEmpty();
            RuleFor(p => p.dc_Zaman).NotEmpty();
        }
    }
}
```

TrDataModelUpdateValidator.cs

### 7.5 – Constants

```
namespace Business.Constants
{
    public static class Messages
    {
        public static string ProductAdded = "Ürün Eklendi";
        public static string ProductNameInvalid = "Ürün Adı Geçersiz";

        public static string MaintenanceTime = "Bakım yapılıyor";
        public static string ProductListed = "Ürünler Listelendi";
        public static string ProductCountCategoryError = "Eklemek istediğiniz kate
goriye daha fazla ürün eklenemez";

        public static string ProductNameAlreadyExists =
            "Bu isimde başka bir ürün var lütfen farklı bir isimde ürün ekleyiniz.
";

        public static string CheckIfCategoryLimitExceded = "Kategori limiti aşıldı
ğı için yeni ürün eklemiyor";

        public static string AuthorizationDenied = "AuthorizationDenied";
        public static string UserRegistered = "Kayıt başarılı";
        public static string UserNotFound = "Kayıt başarılı";
        public static string SuccessfulLogin = "Giriş Başarılı";
        public static string PasswordError = "Parola Hatası";
        public static string UserAlreadyExists = "Bu kullanıcı zaten var";
        public static string AccessTokenCreated = "AccessTokenCreated";
    }
}
```

Messages.cs

# 8 – API Katmanı

## 8.1 – Genel Ayarlamalar

```
using Autofac;
using Autofac.Extensions.DependencyInjection;
using Business.DependencyResolvers.Autofac;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Hosting;

namespace WebAPI
{
    public class Program
    {
        public static void Main(string[] args)
        {
            CreateHostBuilder(args).Build().Run();
        }

        public static IHostBuilder CreateHostBuilder(string[] args) =>
            Host.CreateDefaultBuilder(args)
                .UseServiceProviderFactory(new AutofacServiceProviderFactory())
                .ConfigureContainer<ContainerBuilder>(builder =>
                {
                    builder.RegisterModule(new AutofacBusinessModule());
                })
                .ConfigureWebHostDefaults(webBuilder =>
                {
                    webBuilder.UseStartup<Startup>();
                });
    }
}
```

Program.cs

```
using Core.DependencyResolvers;
using Core.Extensions;
using Core.Utilities.IoC;
using Core.Utilities.Security.Encryption;
using Core.Utilities.Security.JWT;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Localization;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Options;
using Microsoft.IdentityModel.Tokens;
using Microsoft.OpenApi.Models;
using System.Collections.Generic;
using System.Globalization;

namespace WebAPI
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
```

```
        }

        public IConfiguration Configuration { get; }

        public void ConfigureServices(IServiceCollection services)
        {
            services.AddControllers();
            services.AddLocalization(option => option.ResourcesPath = "Resources")
;
            services.Configure<RequestLocalizationOptions>(
                options =>
                {
                    var supportedCultures = new List<CultureInfo>
                    {
                        new CultureInfo("tr-TR"),
                        new CultureInfo("it-IT")


                    };
                    options.DefaultRequestCulture = new RequestCulture(culture: "t
r-TR", uiCulture: "tr-TR");
                    options.SupportedCultures = supportedCultures;
                    options.SupportedUICultures = supportedCultures;
                }
            );
            services.AddSwaggerGen(c =>
            {
                c.SwaggerDoc("v1", new OpenApiInfo { Title = "WebAPI", Version = "
v1" });
            });

            services.AddCors();
            var tokenOptions = Configuration.GetSection("TokenOptions").Get<TokenO
ptions>();
            services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
                .AddJwtBearer(options =>
                {
                    options.TokenValidationParameters = new TokenValidationParamet
ers
                    {
                        ValidateIssuer = true,
                        ValidateAudience = true,
                        ValidateLifetime = true,
                        ValidIssuer = tokenOptions.Issuer,
                        ValidAudience = tokenOptions.Audience,
                        ValidateIssuerSigningKey = true,
                        IssuerSigningKey = SecurityKeyHelper.CreateSecurityKey(tok
enOptions.SecurityKey)
                    };
                });

            services.AddDependencyResolvers(new ICoreModule[]
            {
                new CoreModule()
            });
        }

        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
```

```
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
                app.UseSwagger();
                app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.json"
, "WebAPI v1"));
            }


            app.UseStaticFiles();

            app.ConfigureCustomExceptionMiddleware();

            app.UseCors(builder => builder.WithOrigins("http://localhost:5000").Al
lowAnyHeader());

            app.UseHttpsRedirection();

            var locOptions = app.ApplicationServices.GetService<IOptions<RequestLo
calizationOptions>>();
            app.UseRequestLocalization(locOptions.Value);

            app.UseRouting();

            app.UseAuthentication();

            app.UseAuthorization();

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

Startup.cs

```
"TokenOptions": {
    "Audience": "furkan@furkan.com",
    "Issuer": "furkan@furkan.com",
    "AccessTokenExpiration": 10,
    "SecurityKey": "mysupersecretkeymysupersecretkey"
  },
```

appsettings.json

### 8.2 – Controllers

```
using Business.Abstract;
using Entities.DTOs;
using Microsoft.AspNetCore.Mvc;

namespace WebAPI.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class AuthController : Controller
    {
        private IAuthService _authService;
```

```
        public AuthController(IAuthService authService)
        {
            _authService = authService;

        }

        [HttpPost("login")]
        public ActionResult Login(UserForLoginDto userForLoginDto)
        {
            var userToLogin = _authService.Login(userForLoginDto);
            if (!userToLogin.Success)
            {
                return BadRequest(userToLogin.Message);
            }

            var result = _authService.CreateAccessToken(userToLogin.Data);
            if (result.Success)
            {
                return Ok(result);
            }

            return BadRequest(result.Message);
        }

        [HttpPost("register")]
        public ActionResult Register(UserForRegisterDto userForRegisterDto)
        {
            var userExists = _authService.UserExists(userForRegisterDto.Email);
            if (!userExists.Success)
            {
                return BadRequest(userExists.Message);
            }

            var registerResult = _authService.Register(userForRegisterDto, userFor
RegisterDto.Password);
            var result = _authService.CreateAccessToken(registerResult.Data);
            if (result.Success)
            {
                return Ok(result.Data);
            }

            return BadRequest(result.Message);
        }
    }
}
```

AuthController.cs