**UserManagement.API**

Bu katman, uygulamanın dış dünyaya açılan kapısıdır. HTTP isteklerini alır ve yanıtlar. Genellikle API uç noktalarının tanımlandığı katmandır.

* **Controllers**:
  + API uç noktalarını tanımlar. İsteklerin nasıl işleneceğini belirler ve ilgili servisleri çağırarak iş mantığını uygular. Örneğin, UserController kullanıcılara ait işlemleri yönetir.
* **Filters**:
  + Hata yönetimi, yetkilendirme, ve validasyon işlemleri için kullanılan özel sınıflardır. Örneğin, ExceptionFilter hataları yakalar ve uygun HTTP yanıtlarını döner.
* **Middleware**:
  + HTTP isteklerini ve yanıtlarını işleyen ara katman yazılımlarıdır. Örneğin, AuthenticationMiddleware kullanıcı doğrulamasını yapabilir.
* **appsettings.json**:
  + Uygulama yapılandırma dosyasıdır. Veritabanı bağlantı dizeleri, üçüncü parti servis ayarları gibi yapılandırma ayarlarını içerir.
* **Program.cs**:
  + Uygulamanın başlangıç noktasıdır. Uygulama başlatılırken yapılacak işlemleri ve servisleri burada tanımlar.

**UserManagement.Core**

Uygulamanın iş mantığını ve temel yapı taşlarını içerir. Çekirdek/Uygulama katmanı olarak bilinir.

* **DTOs** (Data Transfer Objects):
  + Veri aktarım objeleridir. Genellikle istemci ve sunucu arasında veri taşımak için kullanılır. Örneğin, UserDTO kullanıcı bilgilerini taşımak için kullanılır.
* **Entities**:
  + Veritabanı tablolarını temsil eden sınıflardır. Örneğin, User sınıfı kullanıcı tablosunu temsil eder.
* **Interfaces**:
  + Arayüzlerdir. Uygulamanın bağımlılıklarını yönetmek ve bağımlılık enjeksiyonunu kolaylaştırmak için kullanılır. Örneğin, IUserService kullanıcı servisinin arayüzüdür.
* **Services**:
  + İş mantığını içeren sınıflardır. Bu sınıflar genellikle arayüzlerle birlikte tanımlanır ve iş mantığını uygular. Örneğin, UserService kullanıcı işlemlerini yönetir.
* **Mappings**:
  + Nesne dönüştürme işlemlerini yöneten sınıflardır. Örneğin, AutoMapper kullanarak DTO'ları entity'lere dönüştürmek için kullanılır.

**UserManagement.Repository**

Veritabanı erişim katmanıdır. Veritabanı işlemlerini ve veri erişim mantığını içerir.

* **Data**:
  + Veritabanı bağlamını içerir. Örneğin, UserManagementDbContext EF Core kullanarak veritabanı ile iletişimi sağlar.
* **Configurations**:
  + Entity yapılandırmalarını içerir. Veritabanı tablolarının nasıl oluşturulacağını belirler. Örneğin, UserConfiguration kullanıcı tablosunun yapılandırmasını içerir.
* **Migrations**:
  + Veritabanı göçleri (migrations) bu klasörde yer alır. Veritabanı şemasının versiyonlanmasını ve güncellenmesini sağlar.
* **Repositories**:
  + Veri erişim sınıflarıdır. Veritabanı işlemlerini gerçekleştiren sınıflardır. Örneğin, UserRepository kullanıcı verileriyle ilgili işlemleri yapar.
* **UnitOfWork**:
  + Birim çalışma (Unit of Work) deseninin implementasyonlarını içerir. Veritabanı işlemlerinin bir arada yönetilmesini sağlar. Örneğin, UnitOfWork sınıfı tüm repository'leri bir arada yönetir.
* **DependencyInjection.cs**:
  + Bağımlılık enjeksiyonu yapılandırma sınıfıdır. Tüm bağımlılıkları burada tanımlar ve uygulamaya ekler.

**UserManagement.Service**

Uygulamanın iş mantığını ve servislerini içerir. Bu katman iş mantığını soyutlar ve API katmanı ile Repository katmanı arasında köprü görevi görür.

* **Authorization**:
  + Yetkilendirme işlemlerini içerir. Kullanıcıların erişim yetkilerini yönetir.
* **Exceptions**:
  + İstisnaları yönetir. Uygulama boyunca oluşan hataları ele alır ve yönetir.
* **Helpers**:
  + Yardımcı sınıflardır. Genellikle sık kullanılan yardımcı metotları içerir.
* **Services**:
  + İş mantığı servislerini içerir. Bu servisler genellikle UserManagement.Core katmanındaki arayüzlerin implementasyonlarıdır.
* **Messaging**:
  + RabbitMQ ve MassTransit ile ilgili yapılandırmaları ve işleyicileri içerir. Mesaj göndericiler ve tüketiciler burada tanımlanır.
* **MappingProfiles**:
  + AutoMapper profillerini içerir. Nesne dönüştürme işlemlerini yönetir.
* **DependencyInjection.cs**:
  + Bağımlılık enjeksiyonu yapılandırma sınıfıdır. Tüm bağımlılıkları burada tanımlar ve uygulamaya ekler.

**UserManagement.WorkerService**

Worker servis katmanıdır. RabbitMQ'dan gelen mesajları alır ve işler.

* **Program.cs**:
  + Worker servis giriş noktasıdır. Uygulamanın başlatılması ve servislerin konfigürasyonu burada yapılır.
* **Worker.cs**:
  + İşleyici sınıfıdır. RabbitMQ'dan gelen mesajları alır ve iş mantığını uygular. Örneğin, gelen kullanıcı bilgilerini konsola yazdırır.
* **MessagingConfiguration**:
  + RabbitMQ ve MassTransit yapılandırmalarını içerir. Mesajlaşma altyapısının konfigürasyonlarını burada yapar.