# E-Ticaret Proje Dökümantasyonu- Yusuf Ziya Demirci

**Proje Özeti**

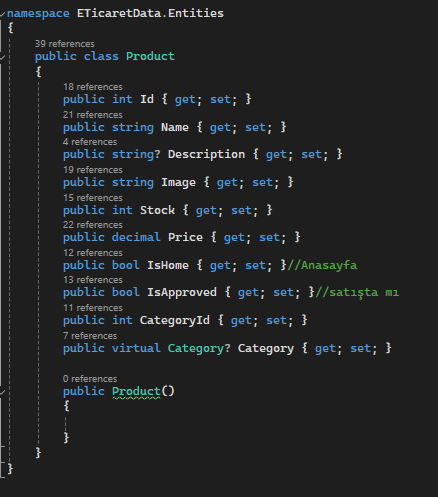
Bu proje, bir E-Ticaret platformu oluşturmak amacıyla geliştirilmiştir. Kullanıcılar ürün satın alabilir, sepet oluşturabilir ve sipariş verebilir. Projede, çeşitli iş katmanları ve veritabanı yönetimi için .NET Core ve Entity Framework Core kullanılmıştır. Aşağıda proje katmanları ve detayları açıklanmıştır.

**Teknolojiler**

- .NET Core 6  
- Entity Framework Core  
- SQL Server  
- Visual Studio 2022  
- ASP.NET Core MVC  
- JavaScript, HTML, CSS

**Projeyi ayağa kaldırma adımları için dökümasyonun sonuna gidiniz!**

**Data Katmanı ve Entity'ler**

**Data katmanı**, veritabanı ile etkileşimi sağlayan katmandır. Bu katman, **Entity Framework Core (EF Core)** kullanarak veritabanı işlemlerini gerçekleştirir ve veri ile ilgili işlemler burada yapılır. Burada veritabanı tablosuna karşılık gelen **entity** sınıfları kullanılır. **Entity**, veritabanındaki bir tabloyu temsil eden bir sınıftır. Yani, her bir **Entity** sınıfı veritabanındaki bir tablonun yapısını temsil eder. Veritabanındaki her satır, Entity sınıfının bir örneği (objesi) ile eşleşir. 

**DbContext Sınıfı**

**DbContext** sınıfı, EF Core kullanarak veritabanı ile etkileşimi yönetir. Bu sınıf sayesinde, **Entity**'lerle veritabanındaki tablolara işlem yapabiliyoruz. Aşağıdaki gibi bir DbContext sınıfı örneği olabilir:

metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**Identity ve Entity'leri**

**ASP.NET Core Identity**, kullanıcı yönetimi ve kimlik doğrulama için kullanılan bir sistemdir. **Identity**, kullanıcıları ve rollerini yönetir. Identity, veritabanındaki kullanıcılara karşılık gelen **IdentityUser** sınıfını kullanır. Ancak projelerde genellikle **AppUser** gibi kendi kullanıcı sınıfımızı oluştururuz.

**IdentityUser ve ApplUser**

**IdentityUser** sınıfı, bir kullanıcının temel bilgilerini tutar. Örneğin, kullanıcı adı, e-posta adresi gibi. Projelerde **ApplicationUser** sınıfını türeterek bu sınıfa ek alanlar (örneğin, ad, soyad, doğum tarihi) ekleyebiliriz.

metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**Validation**

**FluentValidation**, C# projelerinde veri doğrulama işlemlerini daha esnek ve okunabilir bir şekilde yapmamızı sağlayan bir kütüphanedir. Özellikle karmaşık doğrulamalar gerektiğinde tercih edilir. **Entity** sınıflarına eklediğimiz doğrulama kuralları, kullanıcıdan alınan verilerin geçerliliğini kontrol etmek için kullanılır.

**Ne İşe Yarar?**

* **Kullanıcı girişlerini doğrulamak**: Örneğin, bir kullanıcının e-posta adresinin doğru formatta olup olmadığını kontrol edebiliriz.
* **Veri doğruluğu sağlamak**: Kullanıcıdan alınan verilerin, belirlenen kurallara uygun olup olmadığını kontrol eder.

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**ViewModel**

**ViewModel**, genellikle kullanıcı arayüzünde (UI) görüntülenmesi gereken verilerin taşındığı sınıflardır. **Model** sınıflarından farklı olarak, ViewModel'ler genellikle sadece **görünüm (view)** için gerekli olan verileri içerir. Yani, veritabanından veya iş katmanından alınan veriler, kullanıcıya gösterilmeden önce genellikle ViewModel'lere dönüştürülür.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**2-Bussiness Katmanı**

Business Katmanı, yazılımın iş mantığının bulunduğu katmandır ve iş kuralları, doğrulama, iş süreçleri gibi işlemleri yönetir. Bu katman, veritabanı erişimi (Data Katmanı) ve görünüm (View) arasında bir köprü görevi görür.

Projenizde Business Katmanı, genellikle veritabanı işlemlerini (CRUD) yöneten sınıflardan bağımsız olarak daha fazla iş mantığı içerir. Örneğin, kullanıcı kayıt işlemi, sipariş yönetimi veya fiyat hesaplamaları gibi kurallar burada bulunur.

metin, ekran görüntüsü, yazılım içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**3-Dal katmanı**

DAL katmanı, veritabanı ile etkileşimi yöneten katmandır. Bu katman, **veri ekleme**, **güncelleme**, **silme** ve **sorgulama** işlemlerini gerçekleştiren **repository** sınıflarını içerir. Projenizdeki DAL katmanında, **Product**, **Order**, **Category** gibi entity'ler için ilgili veritabanı işlemleri yapılır. Bu katman, iş mantığı ve kullanıcı arayüzünden bağımsız olarak sadece veritabanı ile etkileşir.

metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**4- UI Katmanı** (**User Interface)**

Kullanıcı ile etkileşimi sağlayan katmandır. Bu katman, **görünüm** (view) kısmını oluşturur ve kullanıcılara uygulamanın tüm işlevlerini sunar. UI katmanı, genellikle **HTML**, **CSS**, **JavaScript** gibi teknolojilerle geliştirilir ve ASP.NET Core MVC veya Razor Pages gibi framework'ler kullanılarak dinamik içerikler sunar.

**UI Katmanında Neler Bulunur?**

* **Views (Görünümler)**: Veritabanından alınan veriler ve iş mantığından gelen bilgiler kullanıcıya sunulur. Bu katmanda **ViewModel** kullanılarak veriler UI'ye taşınır.
* **Controllers (Denetleyiciler)**: Kullanıcıdan gelen istekleri işler, gerekli verileri **Business Katmanı** veya **DAL Katmanı** aracılığıyla alır ve sonuçları uygun **View**'a gönderir.
* **HTML/CSS**: UI katmanı, kullanıcının gördüğü sayfalardan oluşur. Görünüm kısmı **HTML** ile yapılandırılır, **CSS** ile tasarımı şekillendirilir.

**Örnek UI İşlemi:**

1. Kullanıcı, ürün listesini görmek için **Products a Get** isteği gönderir.
2. **Controller** bu isteği alır, **Business Katmanı** aracılığıyla ürünleri alır ve ilgili **ViewModel** ile **View**'a gönderir.
3. Kullanıcı, ürünleri görebilir ve etkileşimde bulunabilir.

**UI Katmanı**, projenizdeki **görünüm** kısmını kullanıcıya sunarken, **Controller** ve **ViewModel** ile arka plandaki iş mantığını kullanıcılara aktarır. Bu katman, genellikle **ASP.NET Core MVC** veya **Razor Pages** teknolojileriyle geliştirilir.

Aşağıda ise bir Post İsteğini görüyorsunuz.  
metin, ekran görüntüsü, yazılım içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**View** kısmında ise **@model** anatosyonu ile alıp Razor aracılığı ile Frontende **Modelin** üzerinden gösteriyoruz. metin, ekran görüntüsü, yazılım, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**Projeyi ayağa kaldırma**

Eğer projenizi zip olarak indirdiyseniz, **zip dosyasını çıkartın**.

Proje içerisinde kullanılan dış kütüphaneler için NuGet paketlerini yüklemek gerekecek. Bunun için aşağıdaki adımları izleyin:

* **Visual Studio**: Visual Studio'da proje açıldığında, NuGet paketlerinin otomatik olarak yüklendiğinden emin olun. Eğer yüklenmediyse, **Tools > NuGet Package Manager > Restore NuGet Packages** menüsünden yükleme işlemini yapabilirsiniz.

metin, ekran görüntüsü, yazı tipi, yazılım içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Yukardaki bağımlılıkları yükleyiniz.

Data katmanın içindeki ETicaretContextin içindeki Sql Connection ayarlarını kendi mssql ayarlarınıza göre düzenleyiniz. Database adını CoreData yapınız script uyumluluğu için.metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Packet Manager Konsolundan Migration yapmamız gerekiyor.  
Konsoldan “add-migration (migration ismi) “komutunu ardından “update-database” komutunu çalıştıralım.

**Ardından tablolara hazır verileri eklemek için size verdiğim scripti çalıştırabilirsiniz. (Muhtemelen dökümasyonu açtığınız aynı klasör veya platformun içindedir)**

**Admin** **Kullanıcı adı**: etodemirci **Şifre**: 1234567

**Normal Kullanıcı** (kendiniz de kayıt olabilirsiniz) **Kullanıcı adı:** busraozntrk **Şifre**: 1234567

**Yusuf Ziya Demirci**