**DÜZCE ÜNİVERSİTESİ**

**TEKNOLOJİ FAKÜLTESİ BİLGİSAYAR MÜHENDİSLİĞİ**

**BULUT BİLİŞİM FİNAL ÇALIŞMASI**

**Asp.Net MVC temelli bir web uygulmasının Dockerize edilmesi.**

**172119012 – Seçkin BULGUR**

**Haziran 2021**

# **İÇİNDEKİLER DİZİNİ**

[**İÇİNDEKİLER DİZİNİ** 2](#_Toc73959833)

[**ŞEKİLLER DİZİNİ** 4](#_Toc73959834)

[4](#_Toc73959835)

[**BÖLÜM 1** 5](#_Toc73959836)

[**GİRİŞ** 5](#_Toc73959837)

[**BÖLÜM 2** 6](#_Toc73959838)

[**ASP .NET MVC TEMELLİ BİR WEB SİTESİNİN DOCKERİZE EDİLMESİ** 6](#_Toc73959839)

[**2.1 Asp.Net MVC** 6](#_Toc73959840)

[6](#_Toc73959841)

[**2.2 Docker** 9](#_Toc73959842)

[**2.3 Çalışma Planı** 11](#_Toc73959843)

[**2.4 Program Çıktıları** 12](#_Toc73959844)

[**BÖLÜM 3** 18](#_Toc73959845)

[**SONUÇ** 18](#_Toc73959846)

# **ŞEKİLLER DİZİNİ**

# 

# **BÖLÜM 1**

# **GİRİŞ**

Docker container'lar virtual machine'lere göre daha lightweight ve portable alternatifler sağlamak için kullanılan yapılardır. Sanal makinelerin aksine container'lar uygulama bazlı olup istenilen teknolojiye göre gerekli bağımlılıkların bulunduğu o uygulamaya özel environment'lar ayağa kaldırabilirler.

Docker container, uygulamanızı ihtiyaca göre farklı makinelerde horizontal bir şekilde çoğaltmanızı oldukça kolaylaştırır. Örneğin, Linux’ta geliştirmeyi tamamladık ve mac makinama gittim ve uygulamanın ordada çalışmasını istiyoruz repoyu klonladım ve “docker-compose-up” yapmam o makinada uygulamayı ve bağımlılıklarını kurup tekrar browserdan kolayca erişebilmeme olanak sağladı, bu oldukça ciddi bir kolaylık.

Container'lar aynı zamanda izolasyonu sağlarlar. Container'larda kullanılan bağımlılıklar ve ayarlar, makinenizde çalışan diğer uygulamaları etkilemez. Bu, dependecny-conflict'lerle karşılaşmamıza engel olur.

Deployment süreçleri docker ile birlikte oldukça hızlı bir hale gelir. Oluşturulan image'ler bir merkezi docker-registry'e atılır ve ihtiyaç duyulduğunda tekrar tekrar rebuild işlemi olmadan registry'den pull edilerek kullanılabilir.

Docker, uygulamaları bir container içerisinde build, deploy ve manage etmeye yarayan open-source olarak geliştirilen bir toolkit dir. Container; hem uygulama kodunu hemde gerekli bağımlılıkları içeren bir yazılım birimi olarak tanımlanabilir ve her bir container birbirinden izole bir şekilde aynı operating system'ı paylaşarak host edilirler. Tek şart bu host operating system Windows yada Linux olsun Docker runtime kurulu olması gerekmektedir.Bu sistemler üzerinde çalışabilmektedir.

# **BÖLÜM 2**

# **ASP .NET MVC TEMELLİ BİR WEB SİTESİNİN DOCKERİZE EDİLMESİ**

## **2.1 Asp.Net MVC**

## 

MVC, uygulama geliştirmede (özellikle web uygulaması geliştirmede) önemli yere sahip mimari desenlerden biridir. Günümüzde MVC denince akla Microsoft’un geliştirdiği ASP.NET MVC Framework gelmektedir, 1979 yılından beri (Microsoft 1975 yılında kurulmuştur) yazılım dünyasında yer almaktadır.

MVC, Model , View , Controller kelimelerinin baş harflerinden oluşur ve her kelime MVC’nin farklı bir katmanını ifade eder.

**Model**

MVC dünyasında model uygulama verisinin veya durumunun saklandığı yerdir, genellikle veritabanı veya xml/json dosyası formatındadır.  
Model, veri katmanını (database, xml, json dosyası, vb.) uygulamadan izole eder, böylece diğer katmanlarda veri katmanının neresi olduğunun bilinmesine gerek kalmaz.  
Model katmanı sıklıkla Entity Framework, Nhibernate, LLBLGen, vb. gibi araçlar kullanılarak oluşturulur.

**View**

View, istemcinin gördüğü arayüzü içeren katmandır, genellikle Model katmanındaki verinin kullanılması ile oluşturulur. View katmanının Model ve Controller katmanlarından ayrılması ile arayüz değişikliklerinin uygulamanın diğer katmanlarını değiştirmeye gerek kalmadan yapılabilmesi sağlanmıştır.

View katmanında HTML5 ve CSS3 gibi son versiyon teknolojiler kullanmak mümkündür. HTML5 ve CSS3 ile masaüstü ve mobil tarayıcılarda çalışabilen uygulamalar geliştirmek çok kolaylaşmıştır. Hatta Windows Store uygulamaları geliştirmek için HTML5 ve CSS3 teknolojilerinden yararlanılabilir.

**Controller**

Controller, istemciden gelen isteği işlemek, Model ve View katmanları arasında köprü olmak gibi görevleri yerine getirir. Controller içerisinde bir veya daha fazla Action olabilir, genellikle her Action bir web sayfası üretmek için kullanılır.

**Routing**

Routing, istemci’nin uygulamaya yaptığı isteği uygun Controller ve Action’a yönlendiren yapıdır. İstemci, isteği uygulamanın belli bir adresine gönderir, routing mekanizması sayesinde ilgili adres için en uygun Controller ve içerisindeki Action tespit edilir ve çalıştırılır.

**Neden ASP.NET MVC**

MVC ile istemci’nin isteğine karşılık üretilen çıktı üzerinde çok büyük kontrol imkanı vardır. Bu sayede her alanına müdahele edebildiğimiz ve isteğe en uygun çıktının üretilebilmesi sağlanmıştır.

MVC ile tekrar kullanılabilir (reusable) kod üretmek mümkündür. MVC’nin katmanları birbirinden ayrıldığı için her bir katmanın başka projelerde kullanılabilmesi sağlanmıştır.

MVC ile istemci’nin istek göndereceği adresler üzerinde çok büyük kontrol imkanı vardır. Bu sayede adres’in içerik ile tam bir ilişki içerisinde olması sağlanmıştır. Arama motorları için adres-içerik ilişkisi önemli olduğu için uygulamanın bulunabilirliğine katkısı yüksektir.

MVC ile test edilebilir uygulamalar geliştirme çok kolaylaşmıştır. Katmanların birbirinden ayrı olması ve Test Driven Development yapmayı kolaylaştırıcı mimarisi sayesinde test edilebilir uygulama geliştirilebilmesi sağlanmıştır.

**Sonuç**

MVC uzun yıllardır bir çok framework’te ve programlama dilinde kullanılmış (Java, PHP, vb.) ve olgunlaşmış bir desendir. ASP.NET MVC sayesinde .Net framework dilleri ile MVC pattern kullanılarak hızlı çalışan, test edilebilir, tekrar kullanılabilir parçaları olabilen web uygulamaları geliştirilebilmesi sağlanmıştır.

## **2.2 Docker**

Sistem imajı kısaca bir işletim sisteminin klonu demektir. Kullandığınız işletim sisteminin o anki bir yedeğini içerisinde ki tüm veriler ile birlikte almak istiyorsanız bu işleme sistem imajı almak deniyor.

Docker Linux türevinden işletim sistemleri üzerinde çalışır. Bahsetmek istediğimiz imaj ise, Linux türevi işletim sistemlerinden türetilmiş, boyut olarak küçültülmüş ve **Docker** ortamında çalışmaya uygun işletim sistemi klonudur.

**VM Virtual Machine (Sanal Makine)**

Sanal Makine, mevcut işletim sisteminizden başka bir işletim sistemini çalıştırmanıza olanak tanıyan sistemdir. Sanal Makine işletim sisteminiz içerisinde bilgisayarınızda ki diğer uygulamaları çalıştırır gibi bir işletim sistemini çalıştırır. Burada ana bilgisayar işletim sisteminin içerisinde pencerede çalışan konak işletim sistemi yalnızca kendisi için mevcut olan depolama alanıyla sınırlıdır.

**VirtualBox**

Windows, macOS ve Linux üzerinde çalışan açık kaynak kodlu bir sanal makine uygulamasıdır. Ticari bir sürümü yoktur ve tamamen ücretsizdir.

**VMware Player**

Windows ve Linux için kullanılan ve iyi bilinen bir Sanal Makine yazılımıdır . Ticari sürüm olan VMware Workstation’ın ücretsiz karşılığıdır.

**Hypervisor**

Hypervisor çoklu işletim sistemlerinin aynı donanım üzerinde çalışmasını sağlayan bir kod parçasıdır. Bu kod parçası direk donanım üzerinde çalışarak fiziksel sunucumuz üzerinde birden fazla guest operating system yani konuk işletim sistemleri oluşturmamıza olanak sağlamaktadır.

Yeni bir web yığını üzerine kurulmuş yeni bir çerçeve olmasına rağmen, ASP.NET ile yüksek derecede konsept uyumluluğuna sahiptir. ASP.NET Core uygulamaları, aynı makinede çalışan farklı uygulamaların ASP.NET Core'un farklı sürümlerini hedefleyebildiği yan yana sürüm oluşturmayı destekler. Bu, ASP.NET'in önceki sürümlerinde mümkün değildir.

**Docker Compose**

Docker Compose, çok kaplamalı Docker uygulamalarını tanımlamak ve çalıştırmak için bir araçtır . Compose ile uygulamanızın hizmetlerini yapılandırmak için bir YAML dosyası kullanırsınız. Ardından, tek bir komutla, tüm hizmetleri yapılandırmanızdan oluşturur ve başlatırsınız. Docker Compose, çoklu mikro servisler , veritabanları ve benzeri bağımlılıklardan oluşan ve yapıların çalıştırılması için kullanılır . “docker-compose.yml”, gerekli olan tüm servisleri tek bir yerden konfigüre etmemize ve hepsini tek bir komut ile oluşturup, çalıştırmamızı sağlar.

**Docker Machine**

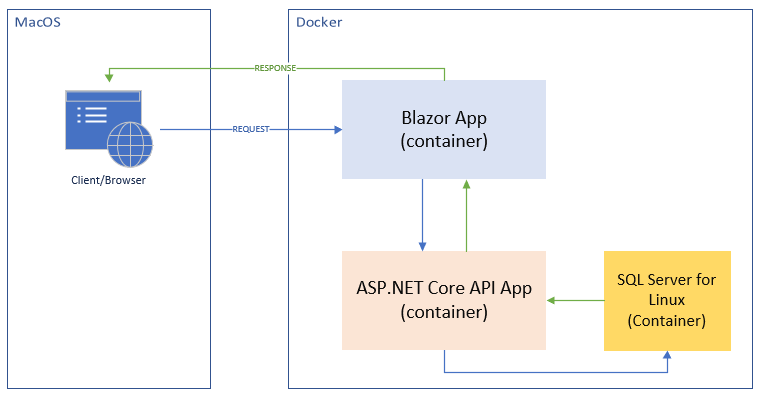
Docker Machİne, sanal ana bilgisayarlara Docker Engine’i yüklemenizi ve ana bilgisayarları “docker-machine” komutlarıyla yönetmenizi sağlayan bir araçtır. Docker, içerisinde birden fazla Docker Engine motoru yönetilebilir. Docker Machine, Docker Engine’i uzaktaki yer alan makinelerine yüklemenize ve kendi bilgisayarınızdan, uzaktaki yer alan Docker Engine motorunu yönetilmesini sağlar.

**Docker Swarm**

Docker Swarm, Docker platformu için konteyner orkestrason aracıdır. Veritabanı, uygulama sunucuları, web sunucuları gibi bileşenlerden oluşan büyük kapsama sahip uygulamalarınızı Docker Swarm ile yönetebilir, yük altında kolaylıkla ölçekleme yapılabilmektedir.

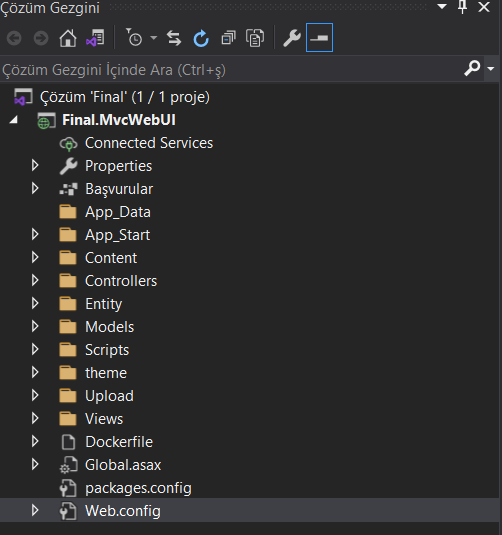
## **2.3 Çalışma Planı**

Her rapor metninde mutlaka bir blok şema ve bir akış diyagramı olmak zorundadır. Blok şema gerçekleştirilen sistem hakkında görsel olarak fikir veren bir şema olmalıdır. Akış şeması ise, geliştirilen sistemdeki tüm kodun şeması olabileceği gibi sadece belirli bir fonksiyonun akış şeması da olabilir. Şekil 1’de örnek bir blok şema verilmektedir.

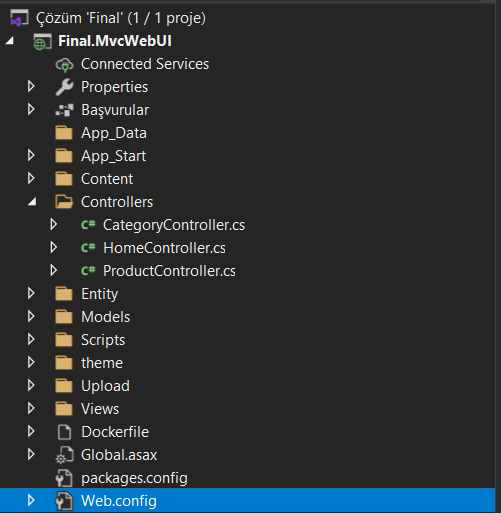


Şekil 1

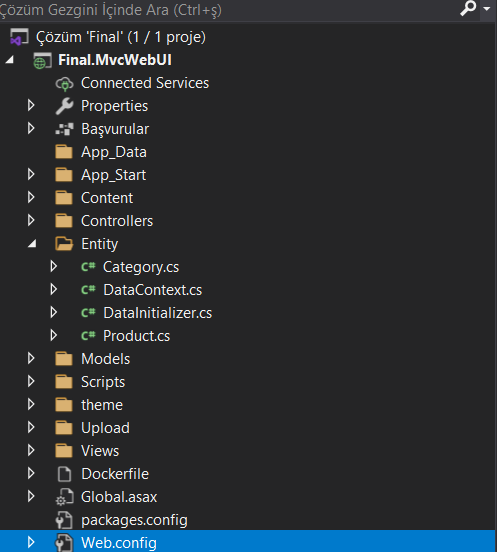
## **2.4 Program Çıktıları**



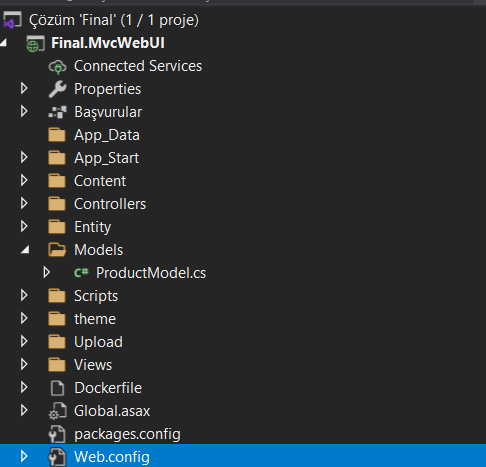
PROJE SLN



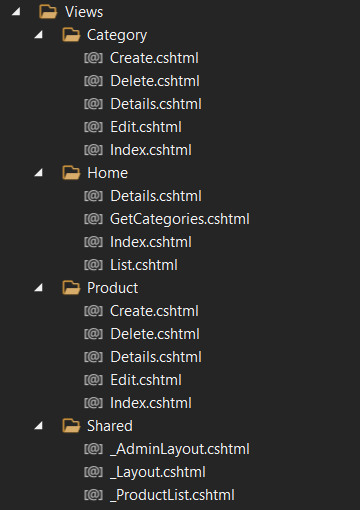
CONTROLLERS KATMANI



ENTİTY KATMANI

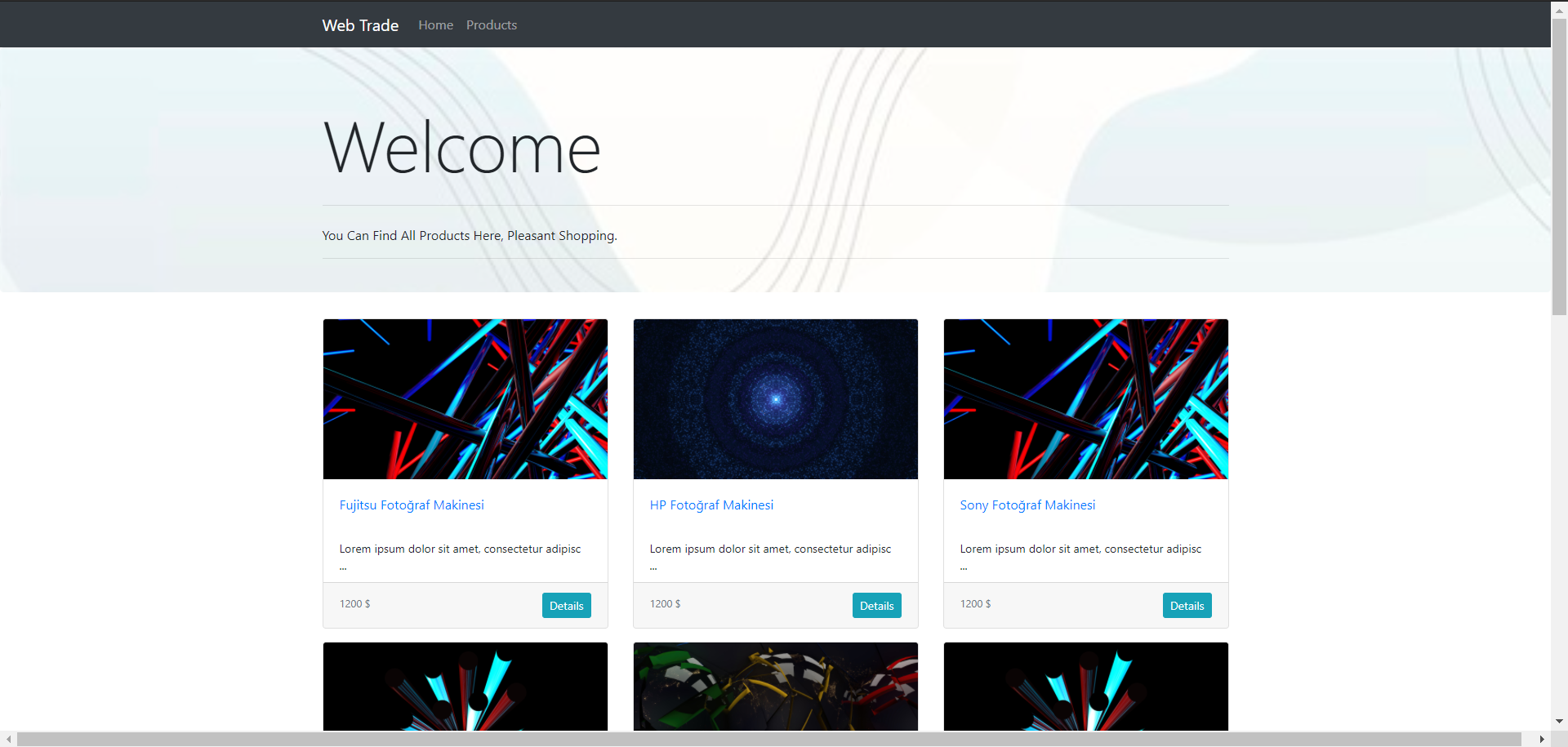


MODELS KATMANI

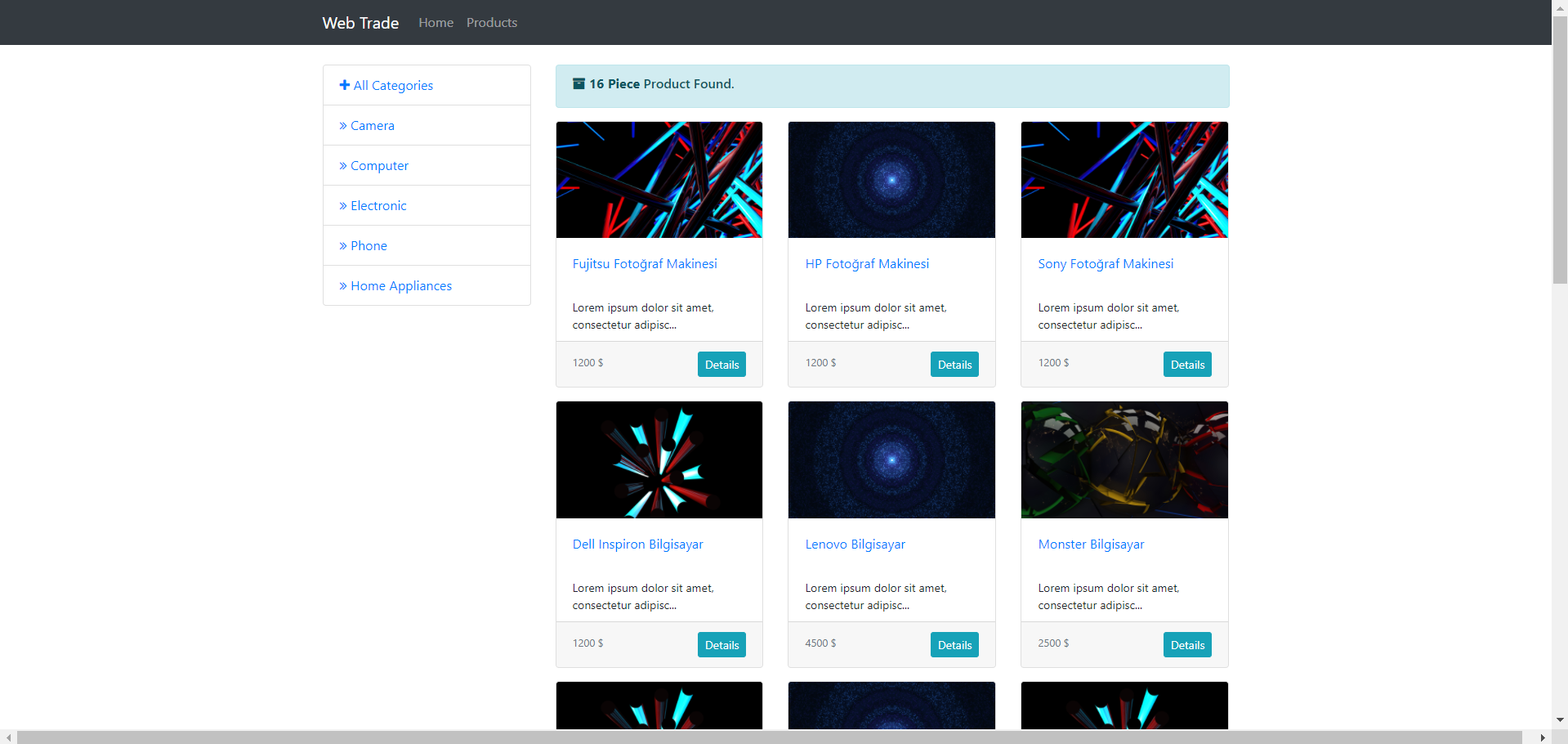


VİEWS KATMANI

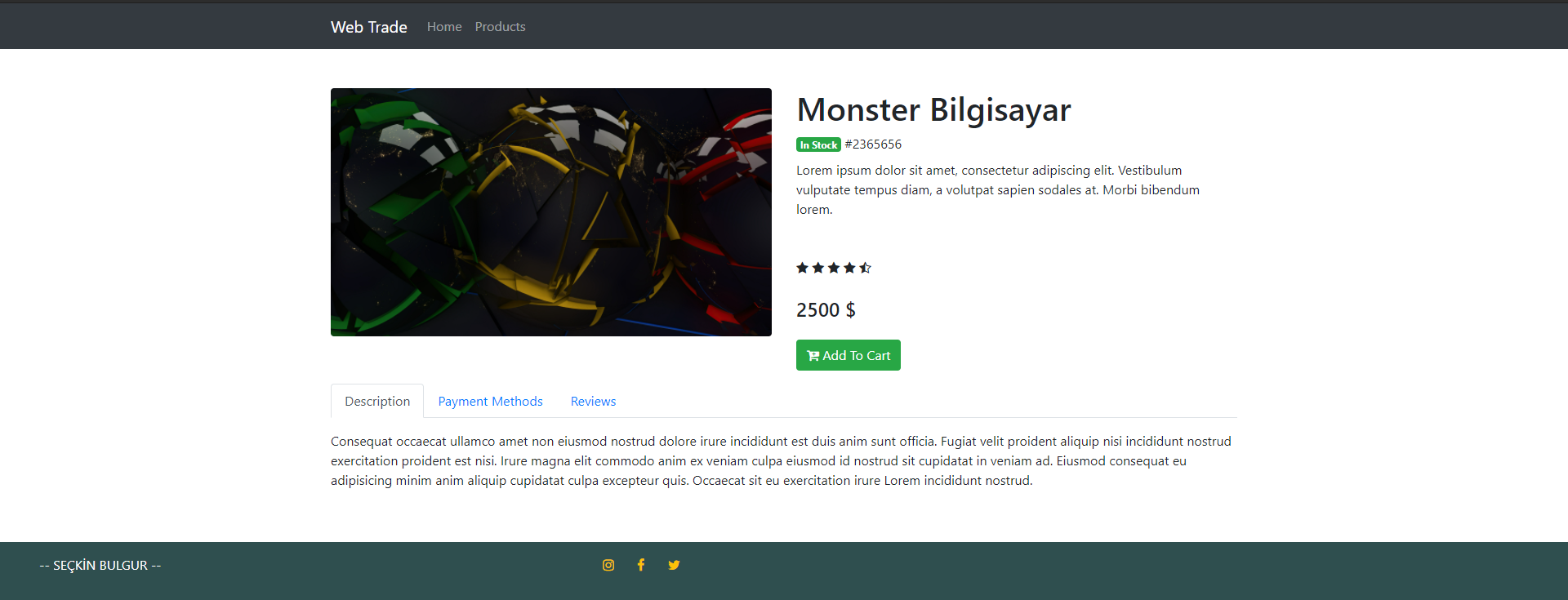
ANASAYFA



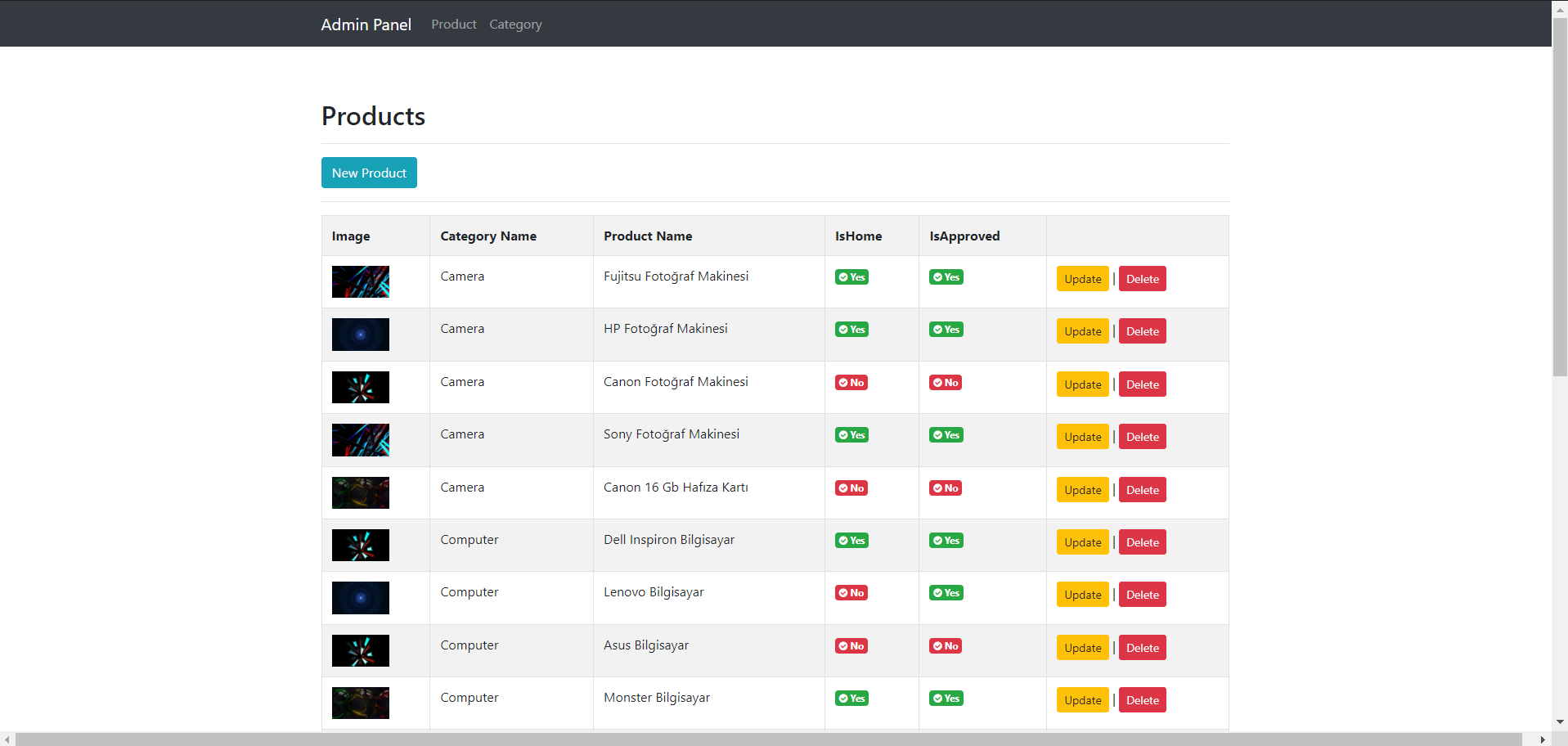
KATEGORİLER



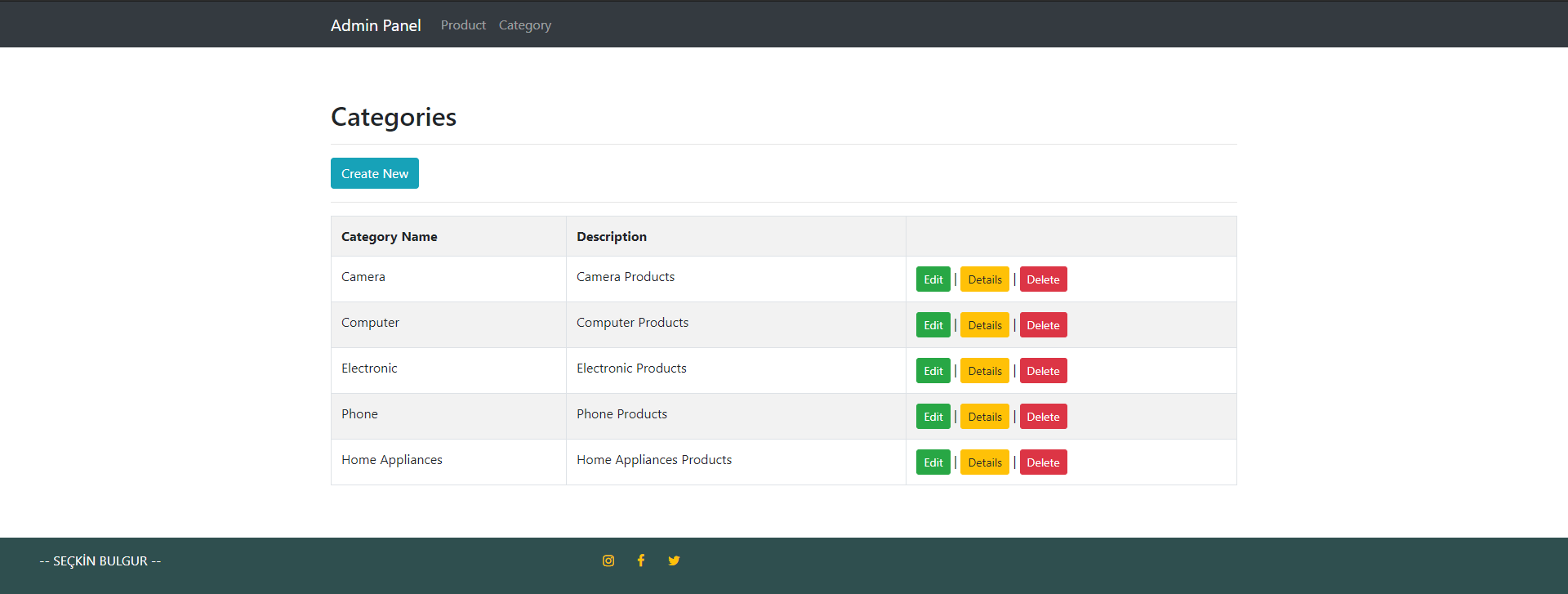
DETAY



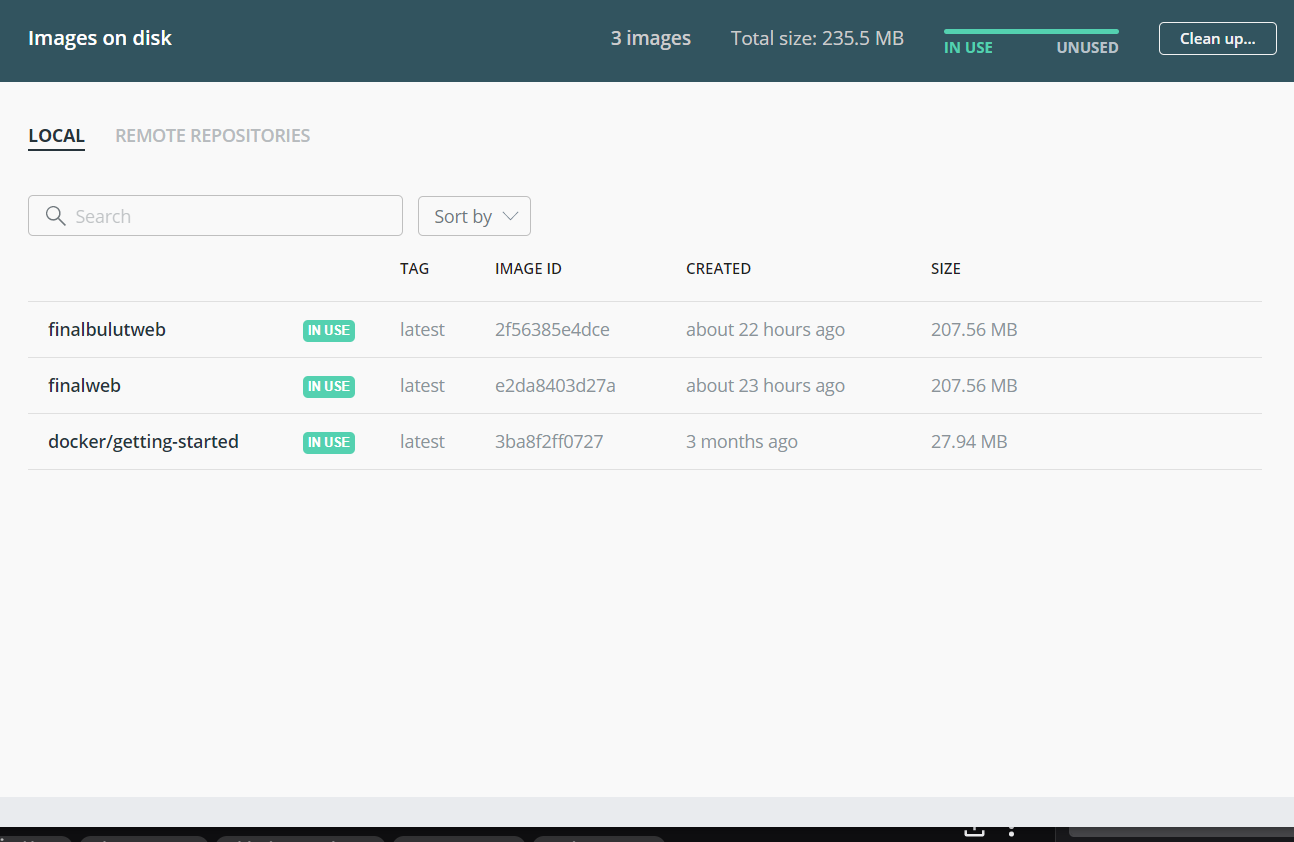
ÜRÜN ADMİN PANEL



KATEGORİ ADMİN PANEL



DOCKER İMAGE



# **BÖLÜM 3**

# **SONUÇ**

Docker, Hypervisor kullanmaz ve tam yüklü bir işletim sistemi barındırmaz bu sayede saniyeler içinde çalışır ve kullanıma hazır hale gelir.

Docker, yazılımlarınızın tüm altyapı gereksinimlerini kod olarak saklar. (versiyonlama) Bu Docker'ın en önemli özelliklerinden biridir.

Bu özellik sayesinde yazılımınızı farklı servis sağlayıcıları üzerinde kolaylıkla gezindirebilir, çoğaltabilir veya paylaşabilirsiniz.

Docker, çok az kaynakla büyük işler yapabilmektedir. Bunun ana sebebi kullandığı konteyner teknolojisidir.

Uygulamalarınıza beklenmedik bir şekilde yüksek trafik geldiğinde saniyeler içerisinde 1000'lerce konteyner hazır hale gelerek yükünüzü omuzlar.

Docker, uygulamalarınızı standart bir zemine oturtarak her platformda aynı şekilde çalışmasını sağlar.

Docker kısaca aynı işletim sistemi üzerinde birbirinden izole yazılımları sanal olarak çalıştırmamızı sağlayan yapıdır.

Bir kere oluştur her yerde kullan mantığı vardır.