ANKARA ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BLM 4061 PROJE RAPORU

E-ticaret Sitesi

Hüseyin Burak Ermiş

19290243

Enver Bağcı

12/2022

ÖZET

Bitirme projemin konusu Apple ürünleri satan bir E-ticaret sitesidir. Dil olarak Asp.NET'i kullandım. Mimari desen olarak bir E-ticaret sitesi olduğu için ona en uygun mimari desen olan MVC (model-view-controller) mimari desenini kullandım. Aynı zamanda E-ticaret sitemiz aktif olarak bir veritabanına bağlıdır. İlişkisel veri tabanı yönetim sistemi (RDBMS) olarak Microsoft Sql Server'ı kullandım. Ön yüz (Front-End) tarafında sitemizi tasarlarken Bootstrap kütüphanesinden yararlandım. Sitemize giriş yapma sayfası (login page), kayıt olma sayfası (register page), anasayfa, ürünlerin listelendiği bir sayfa, her ürüne özel detay sayfası (Ürün ismi, fiyatı, stok durumu ve ürün açıklaması gibi bilgileri içeren bir sayfa), sipariş için girilecek adres bilgi sayfası, sipariş ödemesi için kart bilgilerinin girileceği ödeme bilgileri sayfası, Döviz kurları hakkında bilgi veren sayfa (Verileri Türkiye Merkez Bankasından API kullanarak çektim.) ve hakkımızda sayfasını tasarladım. Bunların dışında yönetici (admin) paneli için ürün ekleme, çıkarma ve günceleme; kategori ekleme, çıkarma ve güncelleme bunların dışında kullancı siparişleri için sipariş detaylarının gözüktüğü ve sipariş süreçlerinin (Siparişin ne durumda olduğunu belirlemek) yönetildiği bir sayfa tasarladım. Asp.NET Identity ile üyelerin giriş (login), çıkış (out), yetkilendirme, token vs. tüm işlemleri hızlı bir şekilde gerçekleştirmemizi sağladım. Yetkilendirme sayesinde kullanıcılar kullanıcı (user) ve yönetici (admin) olmasına göre hangi sayfalara erişimleri varsa o sayfalara erişim sağlıyorlar eğer erişim izinlerinin olmadıkları bir sayfaya girmeye çalışırlarsa giriş (login) ekranına yönlendiriliyorlar. Identity kullanmamın sebebi önceki nesillere nazaran herhangi bir kısıtlaması olmaksızın uygulamalarımızı destekleyen çağdaş bir üyelik sistemi olmasıdır. Aynı zamanda kullanıcı bilgilerimiz (Kullanıcı ismi, soyismi, kullanıcı adı, mail adresi), ürün detayları (Ürün ismi, fiyatı, stok durumu, açıklaması) , kategori detayları (Kategori ismi ve açıklaması) ve bunların dışında sipariş detayları (Kullanıcı bilgileri, alınan ürün bilgileri ve siparişin teslim edileceği adres) kullanıcı ile bağlantılı olacak şekilde veri tabanına kaydediliyor. Sitemizde alışveriş sepeti bulunmaktadır ve aktif olarak çalışmaktadır. Alışveriş sepetine ürünler ekleyerek toplam tutarı görüntüleyebilir, istediğiniz ürünleri sepetinizden çıkarabilirsiniz. Projem şu an canlı ortamda yayınlanmaktadır (Hosting). Sitemin URL'si şu şekildedir: http://www.IStoree.somee.com

İÇİNDEKİLER

1. GİRİŞ

Bitirme projemin konusu Apple ürünleri satan bir E-ticaret sitesidir. Dil olarak Asp.NET'i kullandım. Mimari desen olarak bir E-ticaret sitesi olduğu için ona en uygun mimari desen olan MVC (model-view-controller) mimari desenini kullandım. Aynı zamanda E-ticaret sitemiz aktif olarak bir veritabanına bağlıdır. İlişkisel veri tabanı yönetim sistemi (RDBMS) olarak Microsoft Sql Server 2019'u kullandım. Ön yüz (Front-End) tarafında sitemizi tasarlarken Bootstrap kütüphanesinden yararlandım. Sitemizin giriş yapma sayfasında (Login page) kullanıcıdan kullanıcı adı ve şifre bilgilerini form yoluyla alan ve kontrol eden bir yapıya sahip. Kayıt olma sayfası (Register page) kullanıcıdan kullanıcı adı, ad, soyad, email adres ve şifre bilgilerini bir form yoluyla alan ve sisteme kaydeden bir yapıya sahibiz. Bu işlemleri yaparken Asp.NET Identity üyelik sisteminden yararlandık ve aynı zamanda kullanıcı bilgilerimiz veritabanımıza kaydediliyor. Sitemin anasayfasında reklam ve ödeme hakkında bilgiler içeren bir slider ve yönetici panelinden anasayfada görünürlüklerine izin verdiğim ürünler listeleniyor. Her ürünün kendine özel detay sayfası bulunmakta ve bu sayfa ürünün ismi, fiyatı, stok durumu, ödeme bilgisi ve ürün açıklaması gibi bilgileri barındırıyor. Sitemizde sepeti onayladıktan kullanıcının adres bilgilerini girmesi için bir adres bilgilerinin alındığı bir sayfaya yönlendiriliyor. Bu sayfada form yoluyla kullanıcının adı, soyadı, sipariş vereceği açık adresi, sipariş edeceği şehri, semti, mahallesi ve posta kodu alınmaktadır. Bu bilgiler aynı zamanda veritabanına kaydedilmekte ve sipariş detaylarında görüntülenmektedir. Kullanıcı adres bilgilerini girdikten sonra ödeme bilgilerinin alınacağı ödeme bilgileri sayfasına yönlendiriliyor. Bu sayfada form yoluyla kullanıcının adı, ödeme yapacağı kartın numarası, son kullanma tarihi (ay ve yıl şeklinde) ve CVC güvenlik kodu alınmaktadır. Eğer işlem başarıyla gerçekleşirse sipariş verildiğinin onayını gösteren bir bilgilendirme sayfasına yölendiriliyor. Sitemizde farklı döviz kurlardan alışveriş yapacak müşterilerimizi de düşünerek Döviz kurları hakkında bilgi veren sayfa tasarladım. Bu sayfada Türk lirasının diğer para birimlerindeki karşılıklıklarını gösteren bir tablo yer almaktadır. Tablodaki verileri Türkiye Merkez Bankasından Uygulama Programlama Arayüzü (API) kullanarak sitemize getirdim. Şirketimiz geçmişinden bugüne geçtiği süreçler hakkında bilgi veren bir hakkımızda sayfası tasarladım. Aynı zamanda sitemizin her sayfasının üst kısmında bir navbar bulunmaktadır. Bu navbarı kullanarak

anasayfaya, tüm ürünlerin listelendiği sayfaya, döviz kurlarının listelendiği sayfaya ve hakkımızda sayfasına ulaşabilirsiniz. Bunlara ek olarak navbarın sağ kısmında ürünlerin eklediğiniz sepete giden bir buton, siparişlerinizi ve çıkış yapmanızı sağlayacak bir dropdownbutton bulunmaktadır. Bu kısım kullanıcı giriş yapmadığı zamanlarda sadece giriş yap ve kayıt ol butonlarını barındırmaktadır. Bunların dışında yöneticinin (admin) ürünleri, kategorileri ve sipariş görüntüleyeceği, düzenleyeceği, sileceği ve süreçlere dahil olacağı bir yöneti paneli tasarladım. Panelde bulunan ürün sayfasından yönetici ürün ekleme, çıkarma, günceleme ve ürün araması gibi işlemleri gerçekleştirebilir. Kategori sayfasından kategori ekleme, çıkarma ve güncelleme gibi işlemleri gerçekleştirebilirsiniz. Sipariş sayfasından kullanıcı siparişlerinin detaylarının gözüktüğü ve sipariş süreçlerinin (Siparişin ne durumda olduğunu belirlemek) yönetildiği bir sayfa tasarladım. Bu sayfadan kullanıcının kullanıcı adı, sipariş verdiği adres, sipariş tarihi, sipariş tutarı, sipariş numarası, siparişin durumu ve sipariş verdiği ürünler görüntülenmektedir. Yönetici bu sayfadan ürünün durumunu (Onay bekliyor, onaylandı, kargoya verildi, teslim edildi) güncelleyebilmektedir. Sitemizde 2 tip kullanıcı (normal kullanıcı ve admin) olduğu için yetkilendirme yapısından yararlandık. Yetkilendirme sayesinde kullanıcılar kullanıcı (user) ve yönetici (admin) olmasına göre hangi sayfalara erişimleri varsa o sayfalara erişim sağlıyorlar eğer erişim izinlerinin olmadıkları bir sayfaya girmeye çalışırlarsa giriş (login) ekranına yönlendiriliyorlar. Identity kullanmamın sebebi önceki nesillere nazaran herhangi bir kısıtlaması olmaksızın uygulamalarımızı destekleyen çağdaş bir üyelik sistemi olmasıdır. Aynı zamanda kullanıcı bilgilerimiz (Kullanıcı ismi, soyismi, kullanıcı adı, mail adresi), ürün detayları (Ürün ismi, fiyatı, stok durumu, açıklaması), kategori detayları (Kategori ismi ve açıklaması) ve bunların dışında sipariş detayları (Kullanıcı bilgileri, alınan ürün bilgileri ve siparişin teslim edileceği adres) kullanıcı ile bağlantılı olacak şekilde veri tabanına kaydediliyor. Sitemizde alışveriş sepeti bulunmaktadır ve aktif olarak çalışmaktadır. Alışveriş sepetine ürünler ekleyerek toplam tutarı görüntüleyebilir, istediğiniz ürünleri sepetinizden çıkarabilirsiniz veya siparişi tamamlamak istiyorsanız butona tıklayarak adres bilgilerinin alındığı sayfaya yönlendirilirsiniz. Bunlara ek olarak sepetteki tüm ürünleri sepetten çıkarabileceğiniz ve ürün listelendiği ürün listesi sayfasına yönlendirilebileceğiniz butonlar bulunmaktadır. Projem şuan canlı ortamda somee hosting sayfasında yararlanılarak yayınlanmaktadır. Sitemin veritabanı da aynı

şekilde aktif olarak canlı ortama aktardım ve işlemler sorunsuz bir şekilde çalışmaktadır. Sitemin 2 tip URL'i vardır. Bunlar: http://www.IStoree.somee.com ve http://IStoree.somee.com 'dır.

2. GELİŞME

2.1 .NET

.NET, herhangi bir işletim sisteminde yerel olarak çalışabilen masaüstü, web ve mobil uygulamalar oluşturmaya yönelik açık kaynaklı bir platformdur. .NET sistemi, modern, ölçeklenebilir ve yüksek performanslı yazılım geliştirmeyi destekleyen araçlar, kütüphaneler ve diller içerir. Aktif bir geliştirici topluluğu, .NET platformunu destekler ve bakımını yapar.

Basitçe açıklamak gerekirse .NET platformu şu görevleri yapabilen bir yazılımdır:

- .NET programlama dili kodunu bir bilgi işlem cihazının işleyebileceği talimatlara çevirir.
- Verimli yazılım geliştirme için yardımcı programlar sağlar. Örneğin, mevcut saati bulabilir veya ekranda metin yazdırabilir.
- Bilgisayarda metin, sayı ve tarih gibi bilgileri depolamak için bir dizi veri türü tanımlar.

2.2 ASP.NET

ASP veya ASP. NET, web ve masaüstü uygulamaları oluşturmak için kullanılan bir çerçevedir. ASP yazılımı formlar veya web uygulamaları halinde gelir ve bu programlar Microsoft'un geliştirici platformu olan Visual Studio'da geliştirilir. ASP yazılımı, kurumsal veya küçük ofis ortamlarında bir Windows masaüstü bilgisayarında veya web barındırma sunucusunda çalışır.

ASP.NET, dinamik web uygulamaları geliştirmek için bir çerçevedir. VB.NET, C #, Jscript.NET vb. Dilleri destekler. Programlama mantığı ve içeriği, Microsoft ASP.NET'te ayrı ayrı geliştirilebilir.

ASP.Net, Microsoft tarafından sağlanan bir web geliştirme platformudur. Web tabanlı uygulamalar oluşturmak için kullanılır. ASP.NET ilk olarak 2002 yılında piyasaya sürüldü.

ASP.NET üzerinde dağıtılan ilk sürüm 1.0 idi. ASP.NET, HTTP protokolü ile çalışmak üzere tasarlanmıştır. Tüm web uygulamalarında kullanılan standart protokoldür.

ASP.NET uygulamaları çeşitli .NET dillerinde de yazılabilir. Bunlar C #, VB.NET ve J # içerir. Bu bölümde . ASP'nin tam biçimi Active Server Pages ve .NET, Ağ Destekli Teknolojilerdir.

ASP.NET bir web geliştirme teknolojisidir ve Microsoft .NET platformunun bir parçasıdır. Web uygulaması, standart Windows uygulamasından tamamen farklıdır. Bir web uygulaması bir web sunucusunda konuşlandırılır. İstemci makinedeki web tarayıcısı, HTTP kullanarak web uygulamasına erişiyor. Web tarayıcılarından gelen istekler ve web sunucularının yanıtları HTTP üzerinden yapılır

ASP.NET, söz konusu web uygulamaları, web hizmetleri alanına yönelik olarak ortak dil kullanması bakımından Java gibi açık kaynaklı veri tabanı olanağını sunabilmektedir. İçerik ve doküman olarak Microsoft tarafından geliştirilen bu XML tabanlı uygulamada Html veya XHTML belgesini kullanarak Aspx yapılandırılması şeklinde karşımıza çıkmaya devam etmektedirler. Ancak söz konusu belge özdevinimsiz yani statik bir şekilde karşımıza çıkarak yordamsal programlama özelliğiyle tepkilerin en hızlı şekilde yanıtlanmasını sağlamaktadır.

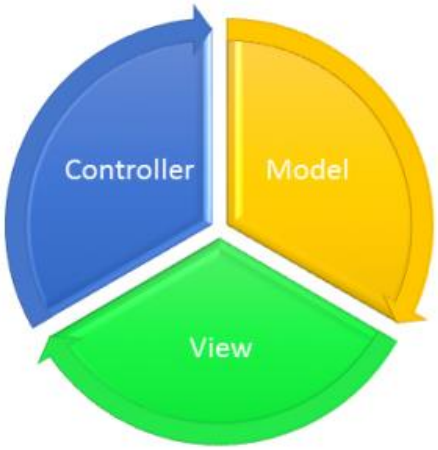
Ayrıca yine diğerlerinden farklı bir özellik konusunda öne çıkan bir durum; html belgelerinin ayrı tutularak bu sayede işletim yordamsal programlama teknolojisinin bir yeniliği olarak anlık duyarlık gösterebilmesi amaç edinilmiştir. Söz konusu bu işleme yönelik olması açısından da öne sürülen birçok gelişim çizgilerinin denetlenerek bu sayede ortaya konulan tepkilere anında bir etki oluşturulması ve etki-tepki bağının kuvvetlendirilmesi sağlanmaktadır. Web sayfalarında özellikle komutların anında tepki verebilmesinde son derece önemli olan bu özellik bakımından son derece işlevsel/fonksiyonel olma niteliğini korumaktadır.

2.3 ASP.NET MVC

ASP.NET MVC, web teknolojilerinin gelişmesiyle birlikte MVC pattern'ini ASP.NET'e eklemek için Microsoft'un 2007 yılında geliştirdiği ASP.NET içerisine eklenen bir yazılım framework'tür.

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, oysa 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.



2.3.1 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.

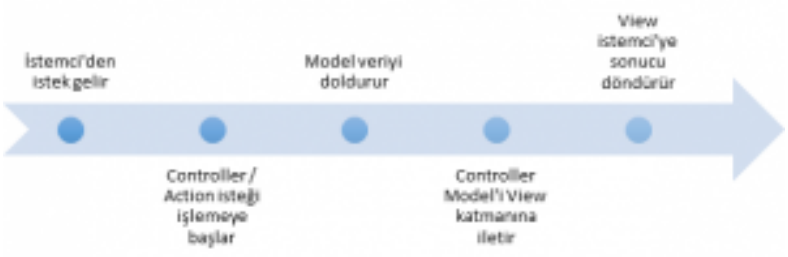
2.3.2 View

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.

2.3.3 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.



MVC Yaşam Döngüsü

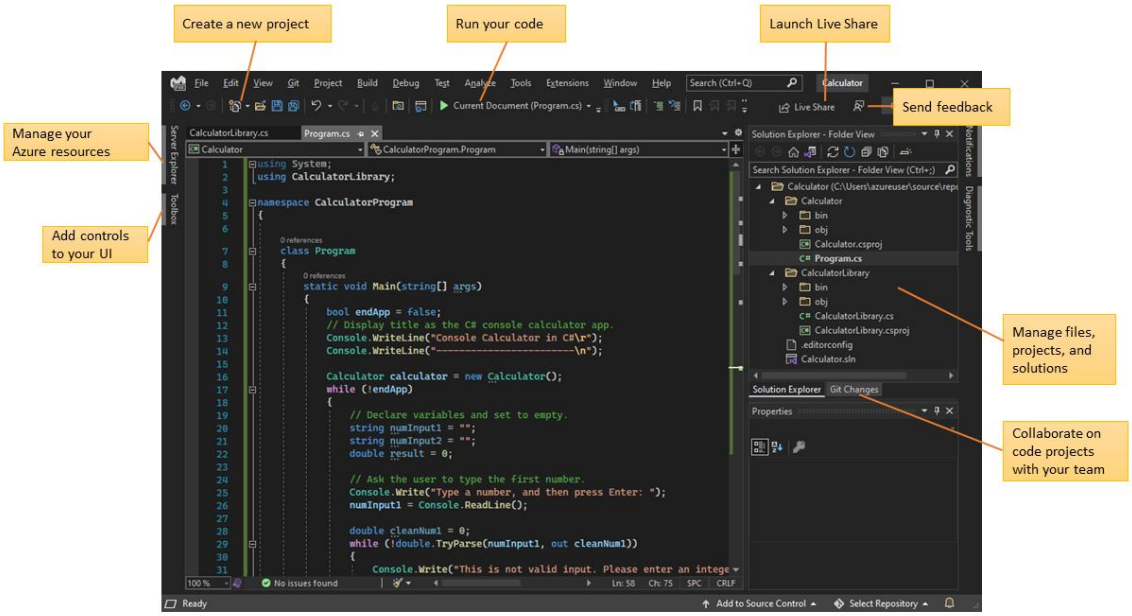
MVC'nin diğer bir önemli yapıtaşı Routing mekanizmasıdır. Routing, istemci'nin uygulamaya yaptığı isteği uygun Controller ve Action'a yönlendiren yapıdır. İstemci, isteği uygulamanın belli bir adresine gönderir, routing mekanizması sayesinde ilgili adres için en uygun Controller ve içerisindeki Action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

2.4 Visual Studio

Tümleşik geliştirme ortamı (IDE), yazılım geliştirmenin birçok yönünü destekleyen zengin özelliklere sahip bir programdır. Visual Studio IDE, kod düzenlemek, hata ayıklamak ve derlemek ve ardından bir uygulama yayımlamak için kullanabileceğiniz yaratıcı bir başlatma bölmesidir. Çoğu IDE'nin sağladığı standart düzenleyici ve hata ayıklayıcının üzerinde, Visual Studio yazılım geliştirme sürecini geliştirmek için derleyiciler, kod tamamlama araçları, grafik tasarımcılar ve daha birçok özellik içerir.

Aynı zamanda Microsoft Visual Studio'nun hem ücretsiz bir "topluluk" hem de ücretli bir "ticari" sürümü mevcuttur. Ücretsiz olması da kullanıcıların bu ortama kolay ulaşabilmesi adına önemli bir etkendir. Yazılım işlerinin mutfağında olanların hayatının bir kısmında

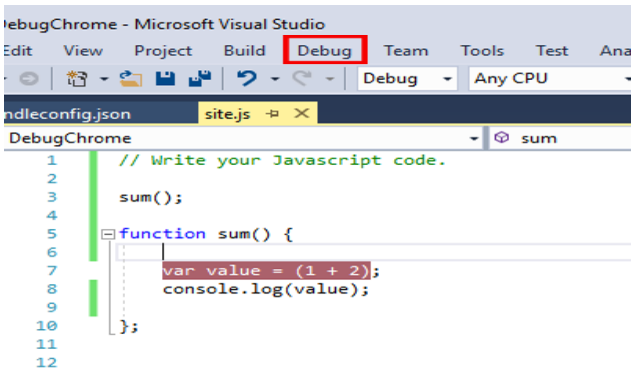
ya da tamamında yer edinmiş bu programın geçmişi 1995 yılına dayanmakla birlikte güncel sürümü 2019 yılında yapımcısı Windows tarafından kullanıcılara sunulmuştur.



2.4.1 Visual Studio Özellikleri

2.4.1.1 Etkili düzenleme ve hata ayıklama (Debugger):

Çeşitli türlere, işlev tanımlarına dayalı bir akıllı kod tamamlamaları olan IntelliSense ile üretkenliği arttırabilme yapabilmekte ve ”Tanıma Git”, “Tüm Başvuruları Bul” gibi özelliklerle de büyük kod tabanlarında rahatlıkla gezinebilme seçeneği düzenlemenin kolaylığı açısından iyi olmakla beraber kesme noktalarını, tam çağrı yığınını ve etkileşimli bir konsol kullandığı için düzenleyiciden kod üzerinde hata ayıklamak son derece basit halde gerçekleşir.

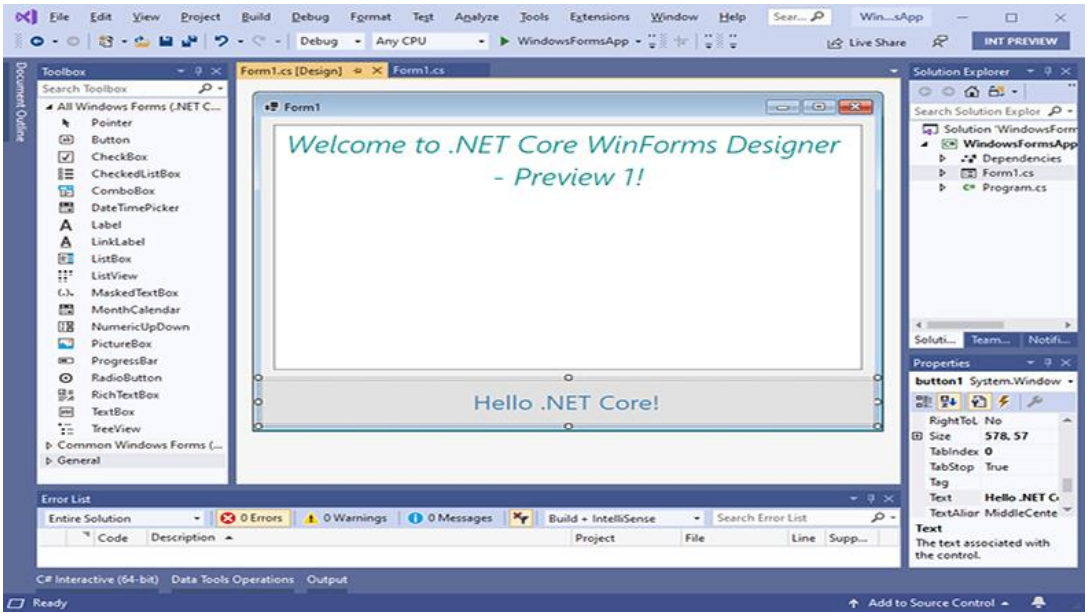


2.4.1.1 Yüzlerce programlama dili için destek

Visual Studio Code, başlıca programlama dillerinin tümünü destekler. JavaScript, TypeScript, CSS ve HTML gibi sık kullanılan web tabanlı diller ve Visual Studio marketinden elde edilen zengin uzantılar diğer yüzlerce programlama dili için tamamlanma, lint uygulama, hata ayıklama ve yeniden düzenleme desteği sunar. Birçok programlama dilini desteklemesiyle de az önce okuduğunuz tüm işlevselliklerini direkt olarak korumaktadır.

2.4.1.2 Tasarımcı (Designer)

Visual Studio uygulamaları, kullanıcılarına arayüzlerini geliştirmek ve yardımcı olmak adına birtakım görsel tasarım aracı sağlamaktadır. Bu araçlar şunlardır; Windows Form Tasarımcısı (Windows Form Designer), WPF Tasarımcısı (WPF Designer), Web Tasarımcısı/Geliştiricisi (Web Designer/Development), Sınıf Tasarımcısı (Class Designer), Veri Tasarımcısı (Data Designer), Eşleştirme Tasarımcısı (Mapping Designer).



2.4.1.3 Genişletilebilirlik

Visual Studio, kullanıcılarına programın işlevselliğini arttırmak adına kodlarını Visual Studio uzantılaryla yazmalarına olanak tanır. Bu uzantılar, Visual Studio'ya "takılır" ("plug into") ve onun işlevselliğini genişletmeyi hedefler. Bu uzantıları; makrolar, eklentiler ve paketler şeklinde görebiliriz.

Makrolar, geliştiricilerin kayıt, yeniden oynatma ve dağıtma için programatik olarak kaydetmesi için tekrarlanabilir görev ve işlemleri temsil eder fakat makrolar, yeni komutları kullanamaz ve araç pencereleri oluşturamazlar. Eklentiler, Visual Studio nesne modeline erişim sağlar ve IDE araçları ile etkileşim için kullanılır.

2.5 Bootstrap

Bootstrap, HTML, CSS ve JavaScript ile yazılmış kullanışlı, yeniden kullanılabilir kod parçalarından oluşan dev bir koleksiyondur. Ayrıca, geliştiricilerin ve tasarımcıların hızla tam olarak duyarlı web siteleri oluşturmasını sağlayan bir frameworktür.

2.5.1 Bootstrap Özellikleri

2.5.1.1 Bootstrap Responsive Yapı Sağlar

Bootstrap kendine ait kullanışlı bir grid sistemi ile gelmektedir. Böylece kendi grid sisteminizi tasarlamak için bir sürü zaman harcamanıza gerek kalmaz. Siz sadece içeriklerinize odaklanabilir, daha verimli olabilirsiniz.

2.5.1.2 Bootstrap Responsive Resimler Sağlar

Bootstrap, görüntüleri geçerli ekran boyutuna göre otomatik olarak yeniden boyutlandırmak için kendi koduyla birlikte gelir. Görüntülerinize img duyarlı sınıfı eklemeniz yeterlidir; önceden tanımlanmış CSS kuralları gerisini halleder.

2.5.1.3 Bootstrap Faydalı Komponentler İçerir

Bootstrap, web sayfanıza kolayca yapıştırabileceğiniz bir dizi özellik ile birlikte gelir. Web sayfanıza sadece göz alıcı tasarım öğeleri eklemek bir esinti olmakla kalmaz, aynı zamanda bunları görüntülemek için kullanılan ekran boyutu veya cihaz ne olursa olsun her birinin harika görüneceğini bilerek emin olabilirsiniz.

2.5.1.4 Bootstrap JavaScript Desteği Sağlar

Bootstrap ayrıca geliştiricilerin bir düzineden fazla özel JQuery eklentisinden faydalanmalarını sağlar. JQuery, modal pop-up'lar, geçişler, görüntü karuselleri ve kişisel favorilerimden biri için bir sayfada gezinirken gezinme çubuğunuzu otomatik olarak güncelleyen bir eklenti için etkileşimle oynamak için daha da fazla alan sunar.

2.5.1.5 Bootstrap Güçlü Bir Dökümantasyon Desteği Sunar

Bootstrap'ın dökümantasyon desteği çok başarılıdır. Her kod parçası, web sitelerinde açık bir şekilde tanımlanmış ve açıklanmıştır. Açıklamalar ayrıca temel uygulama için kod örneklerini de içerir ve en yeni başlayanlar için bile süreci basitleştirir.

2.5.1.6 Bootstrap İhtiyaca Göre Özelleştirilebilir

Frameworkler karşılaştırılırken en önemli parametrelerden birisi performanstır. Eğer framework uygulama üzerinde ağırlık yaratıyor ise uygulamanın yavaş açılmasına neden olabilir. Bootstrap'ın CSS dosyasının boyutu 119 KB'dır. Bu bir CSS dosyası için büyük bir boyuttur.

2.5.1.7 Geniş Topluluğa Sahiptir

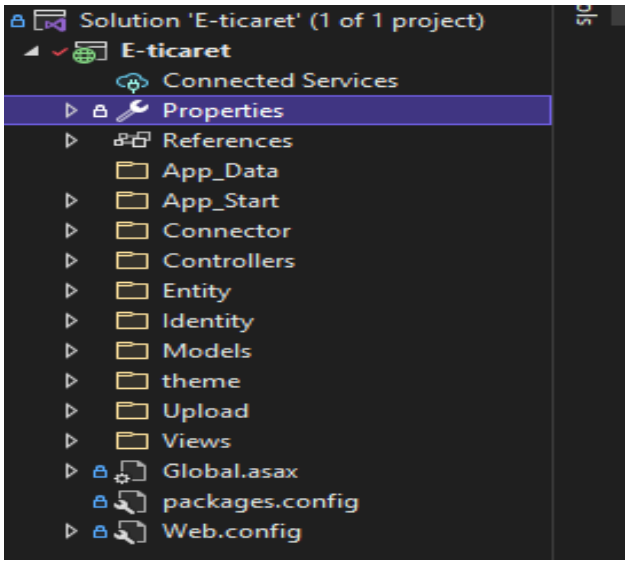
Birçok açık kaynaklı projede olduğu gibi, Bootstrap'ın arkasında büyük bir tasarımcı ve geliştirici topluluğu vardır. GitHub üzerinde barındırılmak, geliştiricilerin Bootstrap'in kod tabanını değiştirmesini ve bunlara katkıda bulunmasını kolaylaştırır.

2.5.1.8 Tema Desteği Sunar

Bootstrap'ın popülaritesi arttıkça, insanlar web geliştirme sürecini daha da hızlandırmak için Bootstrap tabanlı şablonlar oluşturmaya başladılar.

2.5.1 Proje Bilgisi

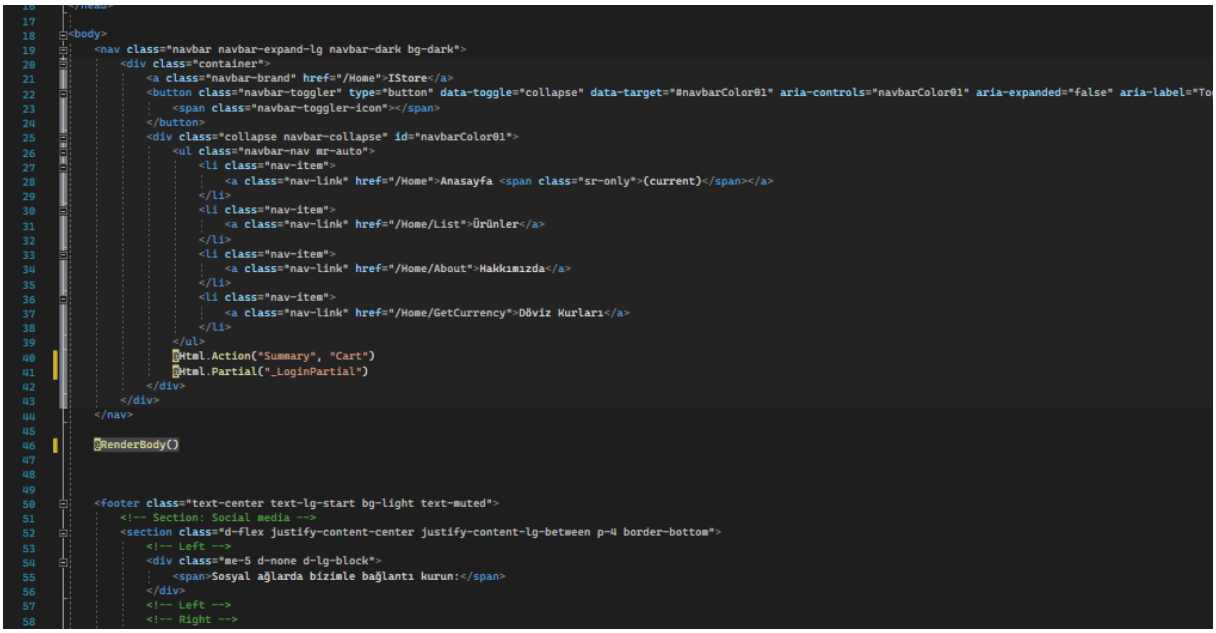
Projemizi çalıştırmak için gerekli olan paketleri şu şekilde sıralayabiliriz: Microsoft.AspNet.MVC, Microsoft.AspNet.Razor, Microsoft.AspNet.WebPages, NEST, Owin, Newtonsoft.Json, Microsoft.Owin, Entity Framework ve Microsoft.AspNet.Identity.



2.5.2.1 Kullanıcı Paneli

2.5.2.1.1 Layout Katmanı

Projemizin her sayfası bootstrap framework'ü kullanılarak tasarlanmıştır. Layout katmanında @RenderBody() kullanılarak diğer sayfalardan gelecek olan içeriği sayfaya yerleştirmek mümkün hale gelmektedir. Layout katmanı master page mantığında kullanıldığından dolayı navbar, footer komponentleri bu katmanda kullanılmaktadır.



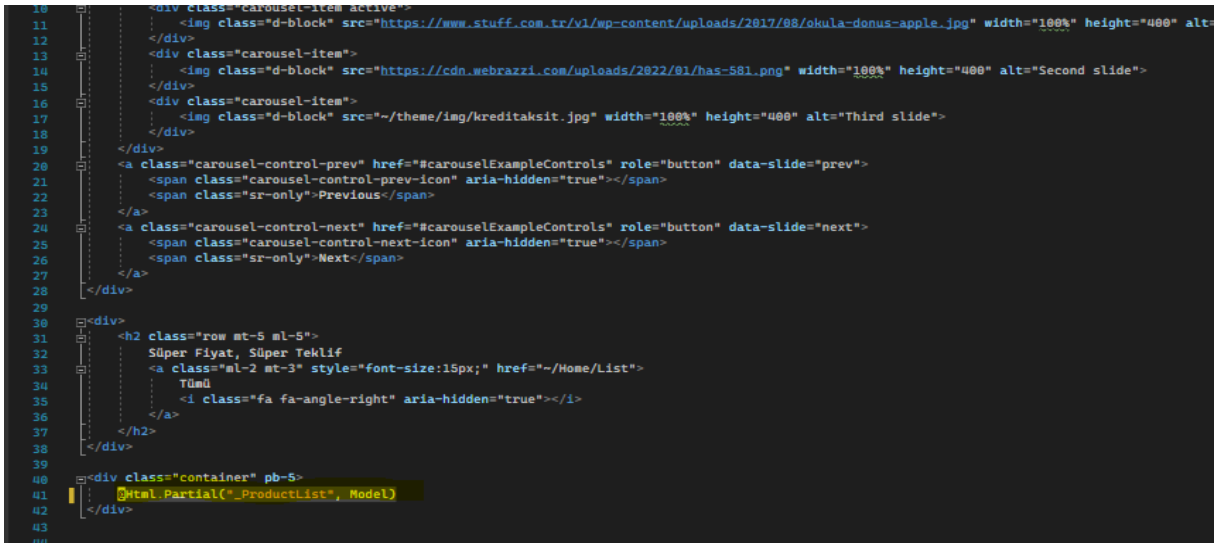
```
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <div class="container">
            <a class="navbar-brand" href="/Home">IStore</a>
            <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarColor01" aria-controls="navbarColor01" aria-expanded="false" aria-label="To
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarColor01">
                <ul class="navbar-nav mr-auto">
                    <li class="nav-item">
                        <a class="nav-link" href="/Home">Anasayfa <span class="sr-only">(current)</span></a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="/Home/List">Ürünler</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="/Home/About">Hakkımızda</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="/Home/GetCurrency">Döviz Kurları</a>
                    </li>
                </ul>
                @Html.Action("Summary", "Cart")
                @Html.Partial("_LoginPartial")
            </div>
        </div>
    </nav>

    @RenderBody()



    <footer class="text-center text-lg-start bg-light text-muted">
        <!-- Section: Social media -->
        <section class="d-flex justify-content-center justify-content-lg-between p-4 border-bottom">
            <!-- Left -->
            <div class="me-5 d-none d-lg-block">
                <span>Sosyal ağlarda bizimle bağlantı kurun:</span>
            </div>
            <!-- Left -->
            <!-- Right -->
```

Kullanıcı giriş yaptıktan sonra navbarın sağ kısmında siparişlerim ve çıkış yap itemlerini içeren dropdownbutton bulunuyor. Eğer kullanıcı giriş yapmazsa navbarın sağ kısmında giriş yap ve kayıt ol butonları yer alıyor. Bu kısımlar _LoginPartial katmanında yer alıyor.

```
<ul class="navbar-nav my-2 my-lg-0">

    @if (Request.IsAuthenticated)
    {
        // kullanıcı adını ve logout linklerini çıkar
        <div class="dropdown">
            <button class="btn btn-secondary dropdown-toggle btn-sm ml-4" type="button" id="dropdownMenuButton" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false"
                Hesabım
            </button>
            <div class="dropdown-menu ml" aria-labelledby="dropdownMenuButton">
                <a class="dropdown-item" href="/Account">Siparişlerim</a>
                <a class="dropdown-item" href="/Account/Update">Hesap Bilgilerim</a>
                <a class="dropdown-item" href="/Account/Logout">Çıkış Yap</a>
            </div>
        </div>
    }
    else
    {
        // login ve rsgister linkleri çıkar

        <li class="nav-item active">
            <a class="btn btn-outline-light btn-sm ml-4" href="/Account/Login">Giriş Yap</a>
        </li>
        <li class="nav-item active">
            <a class="btn btn-outline-light btn-sm ml-2" href="/Account/Register">Kayıt Ol</a>
        </li>

    }
</ul>
```

2.5.1.1.2 Anasayfa

Kullanıcı giriş yaptığında bootstrap kullanarak tasarladığım anasayfa bizi karşılıyor. Anasayfamızda bir slider ve anasayfada görünürlük izinleri verilen ürünler yer alıyor. Anasayfa'da @Html.Partial helper'ı kullanılarak _ProductList katmanındaki view'a göre ürünleri ekrana basılıyor.



2.5.1.1.3 Ürün Sayfası

Kullanıcı navbarda bulunan ürünler linkine tıklayarak ürünlerin listelendiği sayfaya yönlendiriliyor. Bu sayfada da @Html.Partial helper'ını kullanarak _ProductList katmanındaki view'a göre ürünleri ekrana basılıyor. Aynı zamanda ürünleri kategorilere filtrelemek için list-group-item'lar kullanılıyor. Aynı zamanda ürünler içinden arama yapmak için search bar yer alıyor ve bu işlemlerin back-end tarafı homecontroller içinde yer alıyor.

```
public ActionResult SearchProducts(string SearchString) //Ürün arama
{
    var products = _context.Products
        .Where(p => p.IsApproved)
        .Select(p => new ProductModel()
        {
            Id = p.Id,
            Name = p.Name.Length > 50 ? p.Name.Substring(0, 47) + "..." : p.Name,
            Description = p.Description.Length > 50 ? p.Description.Substring(0, 47) + "..." : p.Description,
            Price = p.Price,
            Stock = p.Stock,
            Image = p.Image ?? "nophoto.jpg",
            CategoryId = p.CategoryId
        }).AsQueryable();

    if (!String.IsNullOrEmpty(SearchString))
    {
        products = products.Where(s => s.Name.Contains(SearchString));
    }

    return View(products.ToList());
}

[HttpGet]
0 references
public PartialViewResult GetCategories()
{
    return PartialView(_context.Categories.ToList());
}
```

2.5.1.1.4 Hakkımızda Sayfası

Navbarda bulunan hakkımızda sayfası linkine tıklayarak hakkımızda sayfasına yönlendiriliyoruz. Bu sayfa'da şirketimiz hakkında bilgi veriliyor ve image yer alıyor.

2.5.1.1.5 Döviz Kurları Sayfası

Navbarda bulunan bir diğer linkimiz olan Döviz Kurlarına tıkladığımızda Merkez Bankasından API ile çektiğimiz döviz kurlarının tablo şeklinde basıldığı ekrana yönlendiriliyoruz.

```
<div class="container" pb-5>
    <h2>Döviz Kurları</h2>
    <table class="table-bordered mb-5" width="100%">
        <thead>
            <tr>
                <th>Döviz Kodu<br />Currency Code</th>
                <th>Birim<br />Unit</th>
                <th>Döviz Cinsi<br />Currency</th>
                <th>Döviz Alış<br />Forex Buying</th>
                <th>Döviz Satış<br />Forex Selling</th>
                <th>Efektif Alış<br />Banknote Buying</th>
                <th>Efektif Satış<br />Banknote Selling</th>
            </tr>
        </thead>
        <tbody>
            @foreach (var d in dovizler)
            {

                if (d.CurrencyCode != "XDR")
                {

                    string img = "http://www.tcmb.gov.tr/kurlar/kurlar_tr_dosyalar/images/" + d.CurrencyCode + ".gif";

                    <tr>
                        <td>
                            <img src="@img" />
                            @d.CurrencyCode / TRY
                        </td>
                        <td>@d.Unit</td>
                        <td>@d.Isim</td>
                        @*<td>@d.CurrencyName</td>*@
                        <td>@d.ForexBuying</td>
                        <td>@d.ForexSelling</td>
                        <td>@d.BanknoteBuying</td>
                        <td>@d.BanknoteSelling</td>
                        @*<td>@d.CrossRateUSD</td>
                            <td>@d.CrossRateOther</td>*@
                    </tr>
                }
            }
        </tbody>
```

HomeController içerisinde yer alan GetCurrency methodunda yeni bir xml dökümü ve bir list oluşturuyoruz. Ardından bağlantı kuruyoruz ve count değerini almak için ana boğumu seçiyoruz. Ana boğum altındaki kur boğumunu seçiyoruz. Toplam kur boğumu sayısını elde ediyor ve for döngüsünde kullanıyoruz. Elde ettiğimiz kur bilgilerini listeye ekliyoruz.

```
[HttpGet]
0 references
public ActionResult GetCurrency()
{
    XmlDocument xml = new XmlDocument(); // yeni bir XML dökümü oluşturuyoruz.
    xml.Load("http://www.tcmb.gov.tr/kurlar/today.xml"); // bağlantı kuruyoruz.
    var Tarih_Date_Nodes = xml.SelectSingleNode("//Tarih_Date"); // Count değerini olmak için ana boğumu seçiyoruz.
    var CurrencyNodes = Tarih_Date_Nodes.SelectNodes("//Currency"); // ana boğum altındaki kur boğumunu seçiyoruz.
    int CurrencyLength = CurrencyNodes.Count; // toplam kur boğumu sayısını elde ediyor ve for döngüsünde kullanıyoruz.

    List<Currency> dovizler = new List<Currency>(); // Aşağıda oluşturduğum public class ile bir List oluşturuyoruz.
    for (int i = 0; i < CurrencyLength; i++) // for u çalıştırıyoruz.
    {
        var cn = CurrencyNodes[i]; // kur boğumunu alıyoruz.
        // Listeye kur bilgirini ekliyoruz.
        dovizler.Add(new Currency
        {
            Kod = cn.Attributes["Kod"].Value,
            CrossOrder = cn.Attributes["CrossOrder"].Value,
            CurrencyCode = cn.Attributes["CurrencyCode"].Value,
            Unit = cn.ChildNodes[0].InnerXml,
            Isim = cn.ChildNodes[1].InnerXml,
            CurrencyName = cn.ChildNodes[2].InnerXml,
            ForexBuying = cn.ChildNodes[3].InnerXml,
            ForexSelling = cn.ChildNodes[4].InnerXml,
            BanknoteBuying = cn.ChildNodes[5].InnerXml,
            BanknoteSelling = cn.ChildNodes[6].InnerXml,
            CrossRateOther = cn.ChildNodes[8].InnerXml,
            CrossRateUSD = cn.ChildNodes[7].InnerXml,
        });
    }

    ViewData["dovizler"] = dovizler; // dovizler List değerini data ya atıyoruz ön tarafta viewbag ile çekeceğiz.
    return View();
}
```

2.5.1.1.6 Ürün Detay Sayfası

Ürün kartlarının içinde ürün detaylarını gösteren bir link (ürün incele) yer alıyor, linke tıklandığında her ürünün kendine özel olan sayfasına yönlendiriyor. Aynı şekilde ürün ismine de tıklandığında aynı işlemleri gerçekleştiriyor. Ürün detay sayfada ürünün fotoğrafı, ismi, fiyatı, açıklaması, stok bilgisi, ödeme bilgisi ve sepet ekleme butonu yer alıyor. Eğer ürünün stoğu yoksa hem sepete ekle butonu yerine ürün tükendi butonu geliyor ve herhangi bir eylem gerçekleştirmiyor hem de ürün stoğu hakkında bilgi veren kısımda stokta yok yer yazısı yer alıyor.

2.5.1.1.7 Giriş Yap ve Kayıt Ol Ekranları

Kullanıcı ve yöneticilerin giriş yapması için kullanıcı adı ve şifrenin alındığı bir giriş yap sayfası tasarladım. Kayıt olmamış kullanıcıların ise kayıt olması için kayıt ol sayfası tasarladım. Kayıt ol sayfasında kullanıcını ismi, soyadı, mail adresi ve şifre girmesi isteniyor.

Giriş yapma ve kayıt olma işlemlerini ASP.NET Identity'deki UserManager sınıfını kullanarak AccountController'da gerçekleştirdim. Aynı zamanda sistemde RoleManager kullanarak 2 çeşit kullanıcı rolü tanımladım: Yönetici ve kullanıcı. Siteden kaydolan kişilere otomatik olarak kullanıcı rolü atanmaktadır. Yönetici rolü IdentityInitializer sınıfı içinden tanımlanmaktadır. Yönetici paneline sadece yönetici rolü yetkisi olanlar giriş yapabiliyor.

```
public ActionResult Register()
{
    return View();
}

// POST: Account
[HttpPost]
[ValidateAntiForgeryToken]
0 references
public ActionResult Register(Register model)
{
    if (ModelState.IsValid)
    {
        // Kayıt işlemleri
        var user=new ApplicationUser();
        user.Name = model.Name;
        user.Surname = model.SurName;
        user.UserName = model.UserName;
        user.Email = model.Email;

        IdentityResult result=UserManager.Create(user,model.Password);

        if (result.Succeeded)
        {
            // Kullanıcı oluştu ve kullanıcıyı bir role atayabilirsiniz
            if (RoleManager.RoleExists("user"))
            {
                UserManager.AddToRole(user.Id, "user");
            }
            return RedirectToAction("Login", "Account");
        }
        else
        {
            ModelState.AddModelError("RegisterUserError","Kullanıcı oluşturma hatası");
        }
    }
    return View(model);
}
```

2.5.1.1.8 Sipariş Adresi ve Ödeme Bilgileri Sayfaları

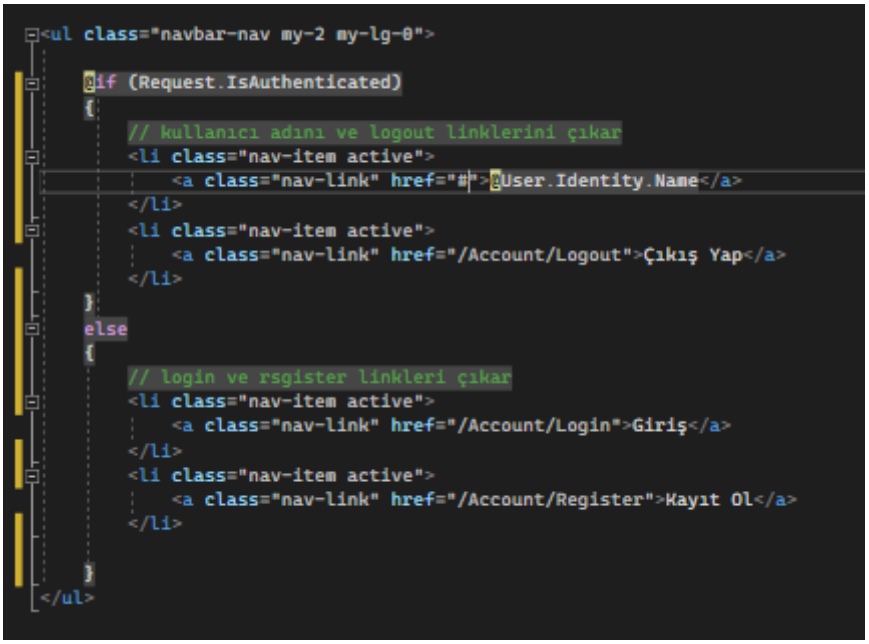
Sitemizde kullanıcı sepetinden alışverişi tamamlarsa adres bilgilerinin alındığı bir sayfaya yönlendiriliyor eğer kullanıcı adres bilgilerini girip ilerlerse ödeme bilgilerinin alındığı ekrana yönlendiriliyor. Bu bilgileri form kullanarak alıyor ve get metodu kullanarak yolluyoruz. Ödemede bir sıkıntı çıkmazsa siparişin sorunsuz bir şekilde verildiğini teyit eden bir ekrana yönlendiriliyor.

```
@using (Html.BeginForm())
{
    @Html.ValidationSummary()
    <div class="form-group row">
        <label class="col-sm-2 col-form-label">Kullanıcı Adı</label>
        <div class="col-sm-10">
            <span class="badge-warning">
                @User.Identity.Name
            </span>
        </div>
    </div>
    <div class="form-group row">
        <label class="col-sm-2 col-form-label">Ad Soyad</label>
        <div class="col-sm-10">
            @Html.TextBoxFor(x => x.FullName, new { @class = "form control" })
        </div>
    </div>
    <div class="form-group row">
        <label class="col-sm-2 col-form-label">Adres Tanımı</label>
        <div class="col-sm-10">
            @Html.TextBoxFor(x => x.AdresBasligi, new { @class = "form control" })
        </div>
    </div>
    <div class="form-group row">
        <label class="col-sm-2 col-form-label">Adres</label>
        <div class="col-sm-10">
            @Html.TextBoxFor(x => x.Adres, new { @class = "form control" })
        </div>
    </div>
    <div class="form-group row">
        <label class="col-sm-2 col-form-label">Şehir</label>
        <div class="col-sm-10">
            @Html.TextBoxFor(x => x.Sehir, new { @class = "form control" })
        </div>
    </div>
    <div class="form-group row">
        <label class="col-sm-2 col-form-label">Semt</label>
        <div class="col-sm-10">
            @Html.TextBoxFor(x => x.Semt, new { @class = "form control" })
        </div>
    </div>
    <div class="form-group row">
        <label class="col-sm-2 col-form-label">Mahalle</label>
        <div class="col-sm-10">
            @Html.TextBoxFor(x => x.Mahalle, new { @class = "form control" })
        </div>
    </div>
    <div class="form-group row">
        <label class="col-sm-2 col-form-label">Posta Kodu</label>
        <div class="col-sm-10">
            @Html.TextBoxFor(x => x.PostaKodu, new { @class = "form control" })
        </div>
    </div>
    <div class="form-group row">
        <label class="col-sm-2 col-form-label"></label>
        <div class="col-sm-10">
            <button type="submit" class="btn btn-success">Sipariş ver</button>
        </div>
    </div>
```

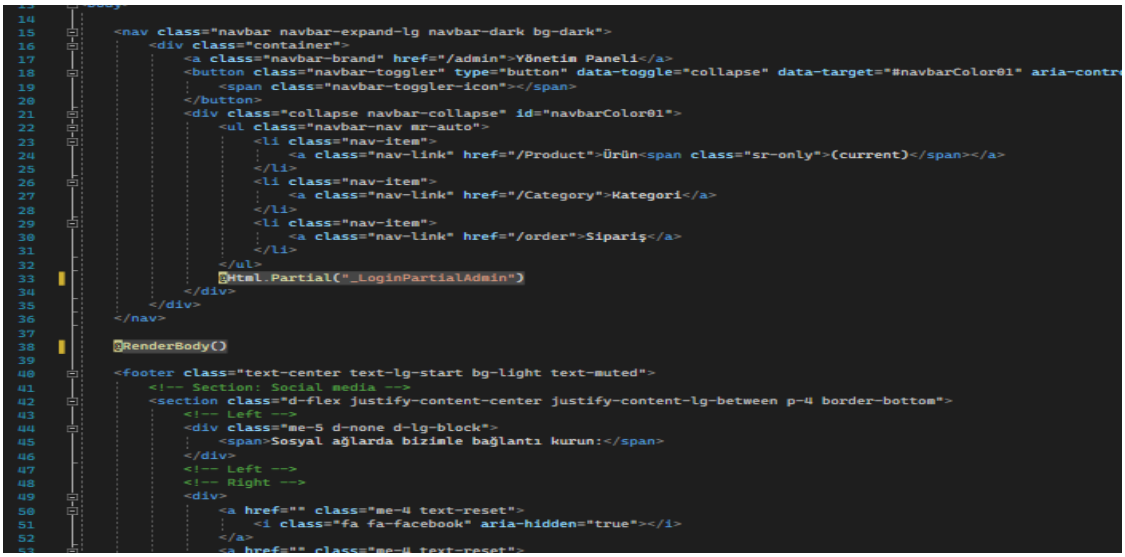

```
@using (Html.BeginForm())
{
    @Html.ValidationSummary()
    <div class="form-group row">
        <label class="col-sm-2 col-form-label">Kullanıcı Adı</label>
        <div class="col-sm-10">
            <span class="badge-warning">
                @User.Identity.Name
            </span>
        </div>
    </div>
    <div class="form-group row">
        <label class="col-sm-2 col-form-label">Ad Soyad</label>
        <div class="col-sm-10">
            @Html.TextBoxFor(x => x.FullName, new { @class = "form control" })
        </div>
    </div>
    <div class="form-group row">
        <label class="col-sm-2 col-form-label">Adres Tanımı</label>
        <div class="col-sm-10">
            @Html.TextBoxFor(x => x.AdresBasligi, new { @class = "form control" })
        </div>
    </div>
    <div class="form-group row">
        <label class="col-sm-2 col-form-label">Adres</label>
        <div class="col-sm-10">
            @Html.TextBoxFor(x => x.Adres, new { @class = "form control" })
        </div>
    </div>
    <div class="form-group row">
        <label class="col-sm-2 col-form-label">Şehir</label>
        <div class="col-sm-10">
            @Html.TextBoxFor(x => x.Sehir, new { @class = "form control" })
        </div>
    </div>
    <div class="form-group row">
        <label class="col-sm-2 col-form-label">Semt</label>
        <div class="col-sm-10">
            @Html.TextBoxFor(x => x.Semt, new { @class = "form control" })
        </div>
    </div>
    <div class="form-group row">
        <label class="col-sm-2 col-form-label">Mahalle</label>
        <div class="col-sm-10">
            @Html.TextBoxFor(x => x.Mahalle, new { @class = "form control" })
        </div>
    </div>
    <div class="form-group row">
        <label class="col-sm-2 col-form-label">Posta Kodu</label>
        <div class="col-sm-10">
            @Html.TextBoxFor(x => x.PostaKodu, new { @class = "form control" })
        </div>
    </div>
    <div class="form-group row">
        <label class="col-sm-2 col-form-label"></label>
        <div class="col-sm-10">
            <button type="submit" class="btn btn-success">Sipariş ver</button>
        </div>
    </div>
```

2.5.2.2 Yönetici Paneli

Yönetici giriş yaptığında sonra navbarın sağ kısmında kullanıcı adını gösteren bir label ve çıkış yapmak için bir buton bulunuyor. Bu kısımlar _LoginPartialAdmin katmanında yer alıyor.



Yönetici giriş yaptığından navbar tasarımı açısından normal kullanıcının giriş yaptığından farklı bir layout katmanı (master page) yer alıyor. Bu katmanın navbarında Ürün, sipariş, kategori ekleme, silme ve güncelleme sayfalarına giden linkler yer alıyor. Yönetici Layout katmanında da @RenderBody() kullanılarak diğer sayfalardan gelecek olan içeriği sayfaya yerleştirmek mümkün hale gelmektedir.



2.5.2.2.1 Anasayfa

Yönetici yöneticine paneline giriş yaptığında onu anasayfada ürün, kategori ve sipariş düzenleme ekranlarına yönlendiren butonlar yer alıyor.

```
<div class="container" pb-5>
    <h1>Admin Paneline Hoşgeldiniz</h1>
    <hr />
    <div class="mt-2">
        <a href="/Product/Index" class="btn btn-secondary">
            Ürün Düzenleme Paneli
            <i class="fa fa-cog" aria-hidden="true"></i>
        </a>
    </div>
    <div class="mt-2">
        <a href="/Category/Index" class="btn btn-secondary">
            Kategori Düzenleme Paneli
            <i class="fa fa-cog" aria-hidden="true"></i>
        </a>
    </div>
    <div class="mt-2 mb-5">
        <a href="/Order/Index" class="btn btn-secondary">
            Sipariş Düzenleme Paneli
            <i class="fa fa-cog" aria-hidden="true"></i>
        </a>
    </div>
</div>
```

2.5.2.2.2 Ürün Sayfası

Navbar'da bulunan ürün linkine tıkladığımızda veya anasayfada yer alan ürün düzenleme butonuna bastığımızda sitemizde satışı olan ürünlerin listelendiği ekrana yönlendiriliyoruz. Bu ekran ürün ekleme, güncelleme ve silme işlemlerini yapabiliyoruz. Aynı zamanda ürünlerin arasında istediği ürünü isme göre bulması için search bar bulunmaktadır.

```
// GET: Product
0 references
public ActionResult Index()
{
    var products = db.Products.Include(p => p.Category);
    return View(products.ToList());
}

[HttpGet]
0 references
public ActionResult Index(string SearchString)
{
    var products = db.Products.Include(p => p.Category);
    if (!String.IsNullOrEmpty(SearchString))
    {
        products = products.Where(s => s.Name.Contains(SearchString));
    }
    return View(products.ToList());
}
```

2.5.2.2.3 Kategori Sayfası

Navbar'da bulunan kategori linkine tıkladığımızda veya anasayfada yer alan kategori düzenleme butonuna bastığımızda sitemizde satışı olan ürünlerin kategorilerinin listelendiği ekrana yönlendiriliyoruz. Bu ekrandan kategori ekleme, güncelleme ve silme işlemlerini yapabiliyoruz. Bu işlemler veritabanında da dinamik olarak çalışmaktadır.

```
<h2>Kategori Listesi</h2>
<hr />
<p>
    @Html.ActionLink("Kategori Ekle", "Create", null, new { @class = "btn btn-success" })
</p>
<hr />
<table class="table table-bordered table-striped">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.Name)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Description)
        </th>
        <th style="width: 135px;"></th>
    </tr>

    @foreach (var item in Model)
    {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.Name)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Description)
            </td>
            <td>
                @Html.ActionLink("Güncelle", "Edit", new { id = item.Id }, new { @class = "btn btn-primary btn-sm" })
                @Html.ActionLink("Sil", "Delete", new { id = item.Id }, new { @class = "btn btn-danger btn-sm" })
            </td>
        </tr>
    }

</table>
```

2.5.2.2.4 Sipariş Sayfası

Navbar'da bulunan sipariş linkine tıkladığımızda sitemizden sipariş vermiş olan kullanıcıların siparişlerinin listelendiği ekrana yönlendiriliyoruz. Bu ekrandan sipariş detaylarını görüntüleyebiliyor ve siparişin durumu hakkında güncellemeler yapabiliyoruz. Sipariş durumlarını enum olarak tanımladım. Bu güncelleme işlemi veritabanında da dinamik olarak çalışmaktadır.

```
namespace E_ticaret.Entity
{
    6 references
    public enum EnumOrderState
    {
        OnayBekliyor,
        KargoyaVerildi,
        Onaylandı,
        TeslimEdildi
    }
}
```

Sipariş durumu güncellemede form kullanarak sipariş durumu hakkındaki bilgiyi alıyoruz ve bu bu bilgiyi FormMethod.Post kullanarak OrderController içindeki UpdateOrderState metotuna gönderiyoruz.

```
<dt class="col-sm-3">Durum</dt>
<dd class="col-sm-9">
    <div class="row">
        @using (Html.BeginForm("UpdateOrderState", "Order", FormMethod.Post))
        {
            @Html.HiddenFor(i => i.OrderId)
            <div class="row">
    <dt class="col-sm-9">
        @Html.EnumDropDownListFor(i => i.OrderState, new { @class = "form-control" })
    </dt>
    <dd class="col-sm-3">
        <button type="submit" class="btn btn-primary">Kaydet</button>
    </dd>

            }
        </div>
    </dd>
```

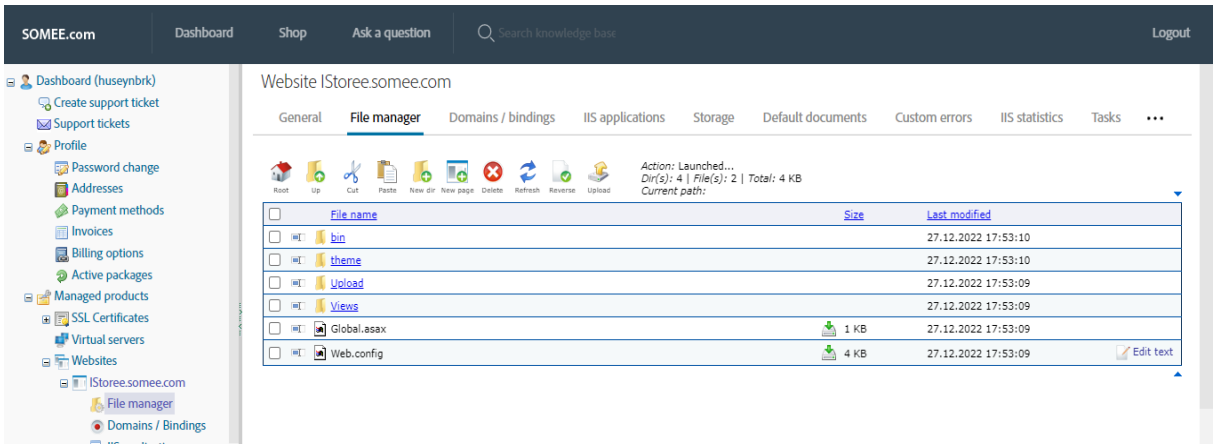
```
0 references
public ActionResult UpdateOrderState(int OrderId,EnumOrderState OrderState)
{
    var order=db.Orders.FirstOrDefault(i=>i.Id==OrderId);
    if (order != null)
    {
        order.OrderState = OrderState;
        db.SaveChanges();

        TempData["message"] = "Bilgileriniz Kayıt Edildi";
        return RedirectToAction("Details", new { id = OrderId });
    }
    return RedirectToAction("Index");
}
```

Sipariş sayfasındaki siparişlerin detayını veritabanımızdaki Orders adlı tablodan ürün ıd'sine göre where kullanarak buluyoruz. Bu tablodan elde ettiğimiz satırdaki bilgileri OrderDetailsModel adındaki model değişkenlere atıyoruz. Eğer veritabanında o ıd'ye sahip sipariş bulunmazsa RedirectToAction kullanarak siparişlerin listelendiği sayfaya yönlendiriliyoruz.

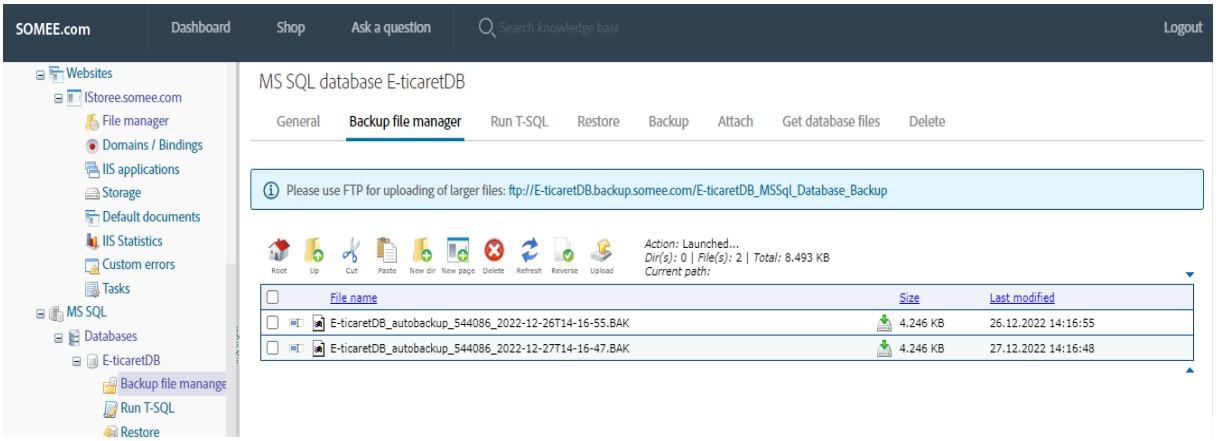
```
0 references
public ActionResult Details (int id)
{
    var entity = db.Orders.Where(i => i.Id == id)
        .Select(i => new OrderDetailsModel()
        {
            OrderId = i.Id,
            Username = i.Username,
            OrderNumber = i.OrderNumber,
            Total = i.Total,
            OrderDate = i.OrderDate,
            OrderState = i.OrderState,
            AdresBasligi = i.AdresBasligi,
            Adres = i.Adres,
            Sehir = i.Sehir,
            Semt = i.Semt,
            Mahalle = i.Mahalle,
            PostaKodu = i.PostaKodu,
            OrderLines = i.OrderLines.Select(a => new OrderLineModel()
            {
                ProductId = a.ProductId,
                ProductName = a.Product.Name,
                Image = a.Product.Image,
                Quantity = a.Quantity,
                Price = a.Price
            }).ToList()
        }).FirstOrDefault();
    return View(entity);
}
```

2.6 Somee

Ücretsiz windows hosting hizmeti veren internet sitesidir. Projemi canlı ortama aktarırken bu internet sitesinden yararlandım ve domain olarak IStoree ismini aldım. Projemi Visual Studio 2022'den publish edip zipliyerek WebSite altındaki File manager kısmına yükledim.



Veritabanı toolumdan (Microsoft SQL Server Management Studio 18) veritabanı ismine sağ tıklayarak task altındaki back up işlemini gerçekleştirdim. Elde ettiğim verileri de MS SQL altındaki Databases kısmına yükledim. Böylelikle veritabanımda canlı ortama aktarılmış oldu.



3. SONUÇ

Bitirme projesini yazarken kullandığım IDE olan Microsoft Visual Studio gerek sağladığı kısayollar olsun gerek veritabanına bağlanmada sağladık kolaylık olsun iş yükümü azalttı. Ayrıca hosting işlemi yapmak için kullanacağım dosyaları publish işlemini kullanarak dosyaları kolayca dışarıya aktardım ve eriştim. Bitirme projemde konu edindiğim E-ticaret sitesini hazırlama da ASP.NET MVC büyük kolaylıklar sağladı. Model-view-controller mimarisi kolay kavranmakta ve işlemler kolayca gerçekleştirilebilmektedir. Model katmanında sayfalarda kullanacağım değişkenlerin tanımladım. Bütün değişkenlerin model katmanında olması kolayca eklenti ve değişiklikler yapmamı sağladı. Controller tarafında viewların çalışması için gerekli işlemleri (Veritabanında veri çekmek gibi) tanımladım. Ayrıca View katmanı ön uç tarafıyla uğraşmamamı sağladı ve direkt projeye dahil olduğu için yüksek performans elde etmemi sağladı. Veritabanı bağlantısını kurarken Entity Framework'dan yararlandım. Entity framework verilerin depolandığı temel veri tabanı tablolarına ve sütunlarına odaklanmadan etki alanına özgü sınıfların nesnelerini kullanarak verilerle çalışmama olanak tanıdı. Entity Framework ile verilerle ilgilenirken daha yüksek bir soyutlama düzeyinde çalışabildim ve geleneksel uygulamalara kıyasla daha az kodla veri odaklı uygulamalar oluşturabildim ve sürdürebildim. Kullanıcılara kimlik doğrulama ve yetkilendirme işlemleri hızlı bir şekilde gerçekleştirmemi sağladı ve bunların dışında önceki nesillere nazaran herhangi bir kısıtlama olmadı. Kullanıcılar, bilgileriyle bir hesap oluşturabilir. Oluşturulan hesaplar Microsoft SQL Server veri tabanı üzerinde depolanıyor. Kullanıcı paneli tarafında bulunan döviz kurları sayesinde yabancı müşterilerimiz (Farklı kurlardan alışveriş yapacak kişiler) de yapacakları alışverişlerde kendi para birimlerine göre çevrimler yapabilir. Döviz kurları bilgilerini Uygulama Programlama Arayüzü (API) kullanarak Türkiye Merkez Bankası sitesinden çektim. API'ler kullanıcıdan ziyade geliştiricinin işlerini kolaylaştırıyor. Bu E-ticaret sitesini geliştirme sürecimde API kullanımı sayesinde iş yükünde azalma sağladım. Bütün işlevleri teker teker kodlamak yerine o işi gerçekleştirecek API'yi uygulamaya entegre ederek hem iş yükünü azalttım hem de zaman kazanılmış oldum. API'lerin diğer artılarından ikisi de güvenlik ve kod kalitesi diyebiliriz. Sayfanın ön uç kısmına gelirsek view tarafında katmanlı yapı kullanarak temiz kod ilkesini başarıyla gerçekleştirdim. @RenderBody() metodu bize diğer viewleri master page denilen sayfanın istediğimiz kısmına bastırmamızı sağlıyor. Bu da karmaşık, uzun

kodları daha kolay anlaşılabilir, erişilebilir ve değişiklik yapılabilir hale getirmektedir. E-ticaret sitemi daha da profesyonel hala getirmek için canlı ortama (Hosting) aktardım. Bu hosting somee internet sayfasını kullanarak ücretsiz şekilde gerçekleştirdim. Somee internet sayfası ücretsiz domain ve hosting işlemi yapmanızı sağlıyor bu da ufak projeleri canlıya aktarmak için yüksek ücretler ödememinizi sağlar. Domain olarak IStoree ismini aldım. Ayrıca E-ticaret sitemi veritabanına bağlı olacak şekilde canlı ortama aktardım.Yani işlemler dinamik olarak gerçekleşmeye devam etmektedir. E-ticaret sitemin URL'i şu şekilde: http://IStoree.somee.com. Özet olarak ASP.NET MVC ve gerekli frameworkleri kullanarak internet sayfası hazırlamak iş yükünüzü oldukça hafifletmekte ve yüksek performans almanızı sağlamaktadır.

4. KAYNAKÇA

1. https://aws.amazon.com/tr/what-is/net/
2. https://learn.microsoft.com/tr-tr/dotnet/welcome
3. https://learn.microsoft.com/tr-tr/aspnet/mvc/overview/getting-started/introduction/getting-started
4. https://www.borakasmer.com/asp-net-mvc-nedir-ne-ise-yarar/
5. https://getbootstrap.com/
6. https://www.w3schools.com/bootstrap/bootstrap_get_started.asp
7. https://visualstudio.microsoft.com/tr/vs/
8. https://www.webtekno.com/microsoft-visual-studio-nedir-h92228.html
9. https://fontawesome.com/
10. https://www.sadikturan.com/
11. https://www.tcmb.gov.tr/kurlar/today.xml
12. http://dotnet-tutorials.net/identity/update-user
13. https://stackoverflow.com/
14. https://learn.microsoft.com/tr-tr/aspnet/identity/overview/getting-started/introduction-to-aspnet-identity
15. https://learn.microsoft.com/en-us/ef/core/get-started/overview/first-app?tabs=netcore-cli
16. https://www.udemy.com/courses/search/?src=ukw&q=asp+net+mvc
17. https://aws.amazon.com/tr/what-is/restful-api/
18. https://somee.com/freeaspnethosting.aspx