ESTUDAM Bilişim Dergisi
Cilt 2, Sayı 1 , 34-41, 2021

Journal of ESTUDAM Information
Volume 2, Issue 1, 34-41, 2021

(Geliş Tarihi / Received Date: 28.12.2020, Kabul Tarihi/ Accepted Date: 29.01.2021)

ASP.NET Ve MVC Temelli Esnek (Responsive) Web Uygulaması

Muammer AKÇAY*[1], Ömer KASIM[2], Zekiye TAŞDELEN[3]

[1]Kütahya Dumlupınar Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, Kütahya
[2]Kütahya Dumlupınar Üniversitesi, Simav Teknoloji Fakültesi, Elektrik-Elektronik Mühendisliği Bölümü, Kütahya
[3]Kütahya Dumlupınar Üniversitesi, Bilgi İşlem Daire Başkanlığı, Kütahya

Anahtar Kelimeler:
Model-View-Controller,
E-ticaret,
Web Programlama,
Dağıtık sistemler,
Bilgi teknolojileri

Özet: Günümüzde İnternetin gelişimi birçok alanda değişiklik ve yeniliklerin oluşmasına imkân sağlamıştır. Bu alanlardan biri de Elektronik Ticaret alanı alanıdır. Elektronik Ticaret'in gelişimi ve değişimi İnternet'ten sonra büyük ölçüde değiştiren ve geliştiren ise Mobil Dünyadaki gelişmeler ve değişimler olmuştur. Mobil Araçların gelişimi ve yaygınlaşması ile birlikte kullanıcıların İnternet'e ve dolayısı ile Elektronik Web Sitelerine ulaşmaları ve alışveriş yapma oranlarında önemli artış olmuştur.
Elektronik Ticaret (E-ticaret) alanında faaliyet göstermek isteyen firmalar tanıtımlarını yapabilecekleri ve ürün veya hizmetlerini sunabilecekleri tasarım ve yazılım açısından güçlü ve etkileşimli Elektronik Ticaret Web sitesine sahip olmak istemektedirler.
Web site tasarımında kullanılan ve Microsoft tarafından geliştirilen C sharp, ASP (Active Server Page - Aktif Sunucu Sayfası) , MVC (Model View Controller – Model Görünüm Kontrol) gibi teknolojilerin gelişimi E-ticaret sitelerinin daha üst seviyede güvenlikli ve daha rahat tasarlanıp kodlanmasını sağlamaktadır. Bu çalışmada geliştirilen dinamik web içeriklerinin kullanılmasıyla MVC teknolojisini temel alan bir E-ticaret sayfası geliştirilmiştir. Model, görünüm ve kontrol bölümlerinin ayrı ayrı gerçekleştirilmesi ile ön taraf (Frontend) ve arka tarafa (Backend) yapılar birbirinden ayrı ayrı olarak geliştirilmiştir. Ön taraf bölümü ASP kaynak kodları ile geliştirilen çalışmada arka taraf bölümünde kontrol ve veri tabanı yer almaktadır. Bu tasarım ile kod optimizasyonu, kodun genişletilmesi, kodun yeniden kullanılması ve kodun güncellenmesi imkânları sunulmaktadır.

ASP.NET And MVC Based Responsive Web Application

Keywords:
Model-View-Controller,
E-commerce,
Web Programming,
Distributed systems,
Information technologies

Abstract: Nowadays, the development of the Internet has allowed changes and innovations in many areas. One of these areas is the area of Electronic Commerce area. The developments and changes in the Mobile World have been the major factors that have changed and improved the development and change of electronic commerce after the internet. With the development and diffusion of Mobile Vehicles, there has been an important increase in the rate of the users reaching the Internet and therefore the Electronic Websites and making purchases. Companies wishing to operate in the field of electronic commerce want to have a strong and interactive Electronic Commerce Web site in terms of design and software that they can promote and offer their products or services.
The development of technologies such as C #, ASP (Active Server Page) and MVC (Model View Controller), which are used in website design and developed by Microsoft, enable e-commerce sites to be designed and coded at a higher level, more secure and more comfortable. Using the dynamic web contents developed in this study, an e-commerce page based on MVC technology was developed. Front End and Back End structures have been developed separately with the realization of model, view and control parts separately. In the project whose Frontend part is developed with ASP source codes, there is a Controller and a database in the Backend part. With this design, opportunities are provided for code optimization, expansion and reuse of code, and updating the code.

*İlgili yazar / corresponding author: makcay26@gmail.com

1. GİRİŞ

Elektronik ticaretin (E-ticaret) hayatımıza girmesi ve hızlı bir şekilde yayılması ile birlikte e-ticaret Web Sitelerinde kullanılan yöntem ve teknolojilerde hızlı bir şekilde gelişmiştir [1]. E-ticaret ile kullanıcıların Web siteleri üzerinden hızlı ve güvenilir bir şekilde alışveriş yapmalarına imkân sağlaması da önemlidir. Bazı kullanıcıların E-ticaret Web Sitelerini ziyaret ederken bu ziyaret süresinin çok kısa sürdüğü gözlemlenmektedir [2, 3]. E-ticaret Web Sitelerinin ürünleri listeleyen kısımları müşterilerin kolay ve rahat bir şekilde gezinmelerine ve o web sitesi üzerinden kolaylıkla alışveriş yapabilmelerine olanak sağlayabilmesi çok önemlidir. Günümüzde teknolojik gelişmeler ile yazılım alanında daha hızlı daha güvenilir web uygulamaları geliştirmek için gerekli altyapıya sahip çok sayıda programlama dili vardır. Bunlardan bir tanesi ASP (Active Server Page) ile kodlanan MVC (Model View Controller) teknolojisidir [4].

Dinamik içeriğin geliştirme süreci .NET platformu kullanılarak gerçekleştirilmektedir. Bu platformdaki ASP ile yapılan kodlama ASP.NET olarak isimlendirilmiştir. ASP.NET herhangi bir programlama dilini kullanarak olay yönlendirmeli web uygulamaları geliştirilmesini sağlayan .NET platformunun web uygulama geliştirme teknolojisidir. ASP.NET, .NET' in XML (Extensible Markup Language) veri yapısını kullanan, MSIL (Microsoft Intermediate Language) ile platform bağımsız kendi başına çalışan geliştirilebilir, ölçeklenebilir, taşınabilir ve dağıtılabilir MSIL web uygulamaları geliştirilmesini sağlayan teknolojidir [5].

ASP.NET, MVC deseni içeren web uygulamaları oluşturmak için bir çerçevedir. MVC modeli ilk olarak 1979 yılında tanıtılmıştır. Desen farklı kod katmanlarının birbirinden ayrılarak web uygulamaları haline getirilmesi sürecini ifade etmektedir. MVC'de sunum katmanı sunucu yan kodundan tamamen ayrılmıştır. Bu durum yazılımın mimarisini daha karmaşık hale getirmesine rağmen kodu daha ölçeklenebilir ve sürdürülebilir hale getirmeye izin vermektedir. MVC mimarisinde üç ana bileşen vardır ve birbirlerinden ayrı tutulurlar [6].

Kaynak kontrolü bir dosyada veya dosya grubunda yapılan değişiklikleri izlemek için kullanılan bir yöntemdir. *Revizyon* adı verilen bireysel değişiklik zaman damgası değişikliği yapan kişi hakkında bilgiler ve bazı ek bilgileri taşıyan benzersiz bir kimlik numarasına sahiptir. Bir grup kullanıcının aynı anda bir dosyayı değiştirebileceği yazılım geliştirme sürecinde değişiklikleri düzenlemek ve sürdürmek çok önemlidir. Orijinal dosyaların kopyalarını oluşturarak ve kopyalanan sürümlerde çalışarak bu görevi elle yapmak mümkün hale gelmektedir. Bu süreçte takım üyeleri değiştirilmiş dosyaların isimlendirilmesinde belirli bir yol izlemesi gerekir. Bu şekilde hata yapma olasılığı çok yüksektir [7]. Bununla birlikte kaynak kontrol sürecini otomatikleştirmek için yazılım geliştirmede kullanılan sürüm kontrol ve kaynak kod yönetim sistemleri olan Git, Mercurial ve Subversion gibi teknolojiler kullanılmaktadır. Otomatik kaynak kontrol teknolojilerinde orijinal dosyaları genellikle *Depo* olarak adlandırılan merkezi bir sunucuda saklanmaktadır [8]. İhtiyaç duyulması halinde dosyalar yerel bir bilgisayara kopyalanır ve değişiklikler yerel olarak yapılır. Dosyaları ana depodan kopyalamak genellikle *Klonlama* olarak bilinmektedir. Bu süreçte uygulamadaki orijinal dosyanın tamamen ayrı bir sürümünde çalışmasına izin verilmektedir. Değişiklik yapılması durumunda güncellemeler *Push* (itme) işlemiyle depoya gönderilmektedir. Depo'da yapılan değişiklikler kaynak kontrol aracının sorumluluğundadır. Bir dosyanın aynı kısmı birden fazla kişi tarafından değiştirilmesi mümkündür. Böyle bir durumda çakışmalar ortaya çıkabilmektedir. Otomatik olarak araçların çoğu bu çakışmaları yönetme ve bir dosyanın farklı sürümlerinde ileri ve geri gitme becerisi de sağlamaktadır [9].

Bu çalışmada geliştirilen dinamik web içeriklerinin kullanılmasıyla MVC teknolojisini temel alan bir E-ticaret sitesi geliştirilmiştir [10-13]. Model, görünüm ve kontrol bölümlerinin ayrı ayrı gerçekleştirilmesi ile ön taraf ve arka taraf birbirinden ayrı geliştirilmiştir. Ön taraf ASP kaynak kodları ile geliştirilen çalışmada arka tarafta Kontrol ve veritabanı yer almaktadır. Bu tasarım ile kod optimizasyonu, kodun genişletilmesi ve yeniden kullanılması, kodun güncellenmesi olanakları sağlanmaktadır.

2. MATERYAL VE METOD

MVC temelli E-ticaret Web sitesi tasarım adımları için yazılım geliştirme araçları kullanılmıştır. Bu araçlar sırasıyla planlama, gereksinim analizi, üst düzey mimari tasarım, prototip oluşturma, teknik tasarım ve uyarlama aşamalarını kapsamaktadır.

2.1. Geliştirilen Modelde Kullanılan Yazılım Geliştirme Araçları

2.1.1. Planlama

Yazılım geliştirme süreci karmaşık ve yinelemeli işlemleri içermektedir. Başarılı bir uygulama geliştirmek için planlama önemli bir alandır. Planlamada geliştirilecek uygulamaya ait belirsiz ve soyut fikirler bulunmaktadır. Bu problemleri önleme noktasında gereksinimleri çıkarmak gerekmektedir. Gereksinimler belirlendikten sonra, geliştiricilerin ne yapacaklarını anlamaları ve uygulamaya ait isterlerin ne oldukları belli olmaktadır. Yazılımı geliştirmek veya mevcut yazılıma yeni özellikler kazandırmak için geleneksel şelale yöntemini izlenmektedir. Teknik dokümantasyon aşaması belirli bir özelliğin uygulanıp test edilmesidir. Bu çalışmada Visual Studio IDE kullanılarak MVC temelli bir tasarım gerçekleştirilmiştir.

Bu çalışmanın amacı WCF ve ASP.NET MVC mimarisiyle oluşturulan esnek (responsive) özellikli sayfaların kazandıracağı esnek platform ile tek sayfa tasarımı, modülerliği oluşturulmayan tasarımlar ve sayfa yayımlandıktan sonraki yeni sürümlerin aktif

edilmesindeki problemlerin çözülebildiğinin gösterilmesidir.

Amaca ulaşmak için gereksinim analizi ve üst düzey mimarinin kısa bir açıklamasını içeren bir web uygulaması geliştirmenin farklı yönlerinin araştırılmasıyla proje başlayacaktır. Mercurial, Ninject ve IIS (Internet Information Service) gibi teknolojilerin netleştirilmesinin yanı sıra, ASP.NET MVC ve WCF (Windows Communication Foundation) kullanılarak geliştirme yapıldığında elde edilen uygulamanın detaylı açıklaması ve kazandırdığı avantajlar hakkında bilgilendirme yapılarak projenin yayımlanması gerçekleştirilir [14].

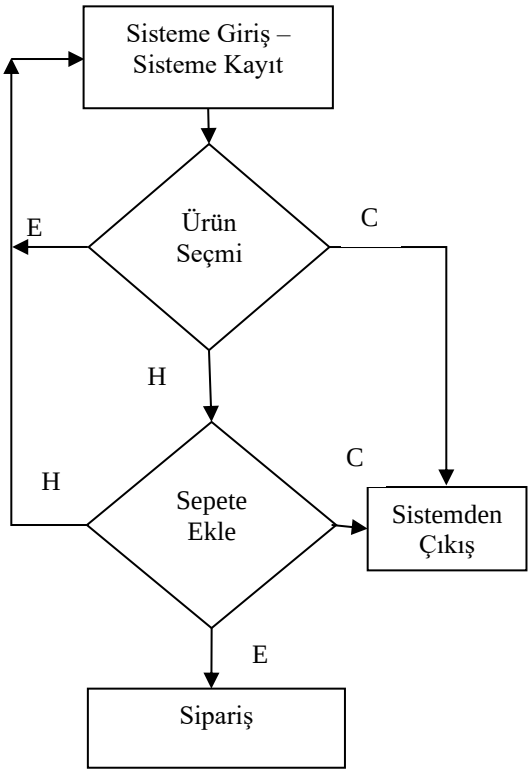
Özellikle MVC ile C# kodlarının yazımıyla parça ve ASP ile esnek özellikli web uygulaması geliştirmek iki önemli katkıdır. Nesneye dayalı bir programlama mimarisinin sunduğu avantajların C# kodları ile kullanılabilmesi modüler bir programlama yapılabilmesi, sınıfların siber saldırı yapacak saldırganlara kapatılabilmesi, test kodlarının yazılabilmesi, model görünüm-kontrol kodlarının birbirinden bağımsız yazılabilmesi çalışma sonrasındaki yeniliklerdir [15].

2.1.2. Gereksinim Analizi

Gereksinim analizi, ürün veya yazılımın ne yapması gerektiğini açıklar. Tamamen yeni bir yazılıma veya mevcut bir yazılıma yeni özellikler kazandırmada kullanılmaktadır [13]. Bu çalışmada odaklanılan web uygulaması bir E-ticaret sitesinin prototipi olarak hazırlanmıştır. Ana faaliyetleri gerçekleştirmek için geliştirilen prototip E-ticaret sitesinin başka bir web uygulamasında yer alan WCF sunucusuna bağlı olarak tasarlanmıştır. Tasarlanan sisteme ait akış şeması Şekil 1.'de verilmiştir.

Tüm gereksinimler belirlendikten sonra akışın analizi yapılmalıdır. Bu işlem ile karışıklık en aza indirilmelidir. Sistemin içerisindeki kod akışının nasıl yapılacağı açık bir şekilde görülebilmektedir. Hem isterlerin hem de geliştiricinin çapraz kontrolü sayesinde gereksinimler önceliğe göre sıralanabilir. Öncelik sağlandıktan sonra görevlerin atanması yapılabilmektedir Geliştirici düşük seviye mimariyi tasarlama işini yönetebilir ve her görev için zaman çizelgesini takip edebilmektedir. Her öncelik gereksinimi görevlere ayırdıktan sonra geliştiriciler ve isterler tüm iş sürecini bir kez daha gözden geçirir ve gereklilikleri tamamlar [16].

Uygulamada ürün yönetimi, fiyat yönetimi, stok yönetimi, içerik kalite yönetimi, resmi hesaplarının açılması, afiş tasarımları, e-posta şablon tasarımları, site içi doğru yön bulma, ilave eklenebilecekler bu çalışmada belirlenen gereksinimlerdir.



Şekil 1. Prototipin basitleştirilmiş iş akışı

2.1.3. Üst Düzey Mimari Tasarım Aşaması

Yazılım tasarım belgesinin (Software Design Document) amacı ilgili taraflara gereksinimlerin nasıl uygulanması gerektiğini açıklayarak ortak bir yönde yönlendirmektir. SDD hem geliştiriciler hem de yöneticiler için hazırlanmaktadır.

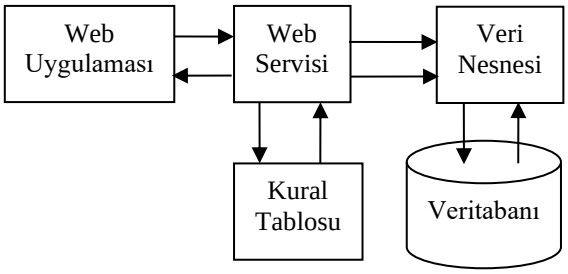
Bu çalışmada veri tabanı tasarımı belgesi (Database Design Document) adı verilen tasarımda veri tabanı ayrı bir sunucuda tutulmaktadır. Veri tabanına çekirdek üzerinden ulaşılmaktadır. Ayrıca prototip uygulama Çekirdek hizmeti ana bilgisayar belgeleri, geliştiricileri ve yöneticileri, üst düzey mimari hakkında kısa bir fikir vererek aynı kaynağa yönlendirmektir [14].

2.1.4. Prototip Uygulama Aşaması

Teknik dokümantasyon yazılımın fonksiyonelliğinin açıklanmasını ifade etmektedir. Bunun amacı yazılımın neler yapabileceği ve nasıl yaptığı hakkında hedef kitleye net bir görüntü sağlamaktır. Bu çalışmada uygulamanın çevresel kurulumundan başlayarak bir prototip geliştirme ve çevrimiçi ortamda yayınlamaya kadar olan süreçler tanımlanmıştır.

2.1. 5. Teknik Tasarım ve Uygulama

Bu çalışmada geliştirilen E-ticaret uygulamasına ait veritabanı ayrı bir sunucuda tutulmaktadır. Çekirdek veritabanına çekirdek üzerinden ulaşılmaktadır. Çekirdek içinde varlık çerçevesi (Entity Framework) veritabanındaki her tablo için veri varlıkları oluşturmada kullanılmaktadır. Web uygulamasının içerisinde bir WCF istemcisi oluşturularak gerekli tüm veri varlıkları almaktadır. WCF istemcisi ve WCF sunucusu yapılandırılabilir uç noktaları kullanarak birbirleriyle veri varlıklarını iletmektedir. Bitiş noktası hem WCF istemcisinin hem de sunucunun karşılıklı el sıkışma köprüsü oluşturmak için kullandığı özellikler kümesidir. WCF sunucusundaki farklı servisler çeşitli iş kurallarını kullanarak veriyi almak için veritabanına bağlanmaktadır [15]. Çalışmanın üst düzey mimari tasarımı Şekil 2.'de gösterilmiştir.



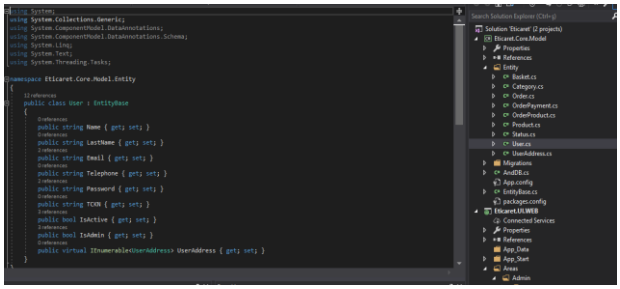
Şekil 2. Üst düzey mimari (High Level Architecture)

2.2. ASP.Net MVC E-Ticaret Projesinin MVC Mimarisi

E-ticaret web uygulaması geliştirilirken MVC Mimarisi genel hatlarıyla anlaşılmaya çalışılmaktadır. E-ticaret projesi default olarak MVC yapısı ile geliştirilmiştir. MVC'nin baş harfleri olan model (Models), görünüm (Views) ve kontrol (Controllers) isimli üç element örnek E-ticaret web uygulamasında MVC projesinin yapıtaşlarını oluşturmaktadır. Views katmanı içerisinde kullanıcıya sunmuş olduğumuz sayfalar yer almaktadır [17]. Ana sayfayı temsilen `Views -> Home` altındaki `index.cshtml` sayfasını göstermek için geliştirilmiştir. Bu projenin Ana sayfasıdır. Burada @ işareti ile başlayan bir syntax gözükmektedir. Bu syntax `Razor View Engine` den gelen bir syntax yapısıdır. Projede `cshtml` sayfaları harmanlanarak bir `html` sayfası ortaya çıkmaktadır. Bu `cshtml` içerisinde sunucuda bir işlem gerçekleştirmek için `Razor syntax`'indan faydalanılmıştır. Bu şekilde sayfa ile iç içe bir yazım stili gerçekleştirilmiştir.

Görünüm (Views) katmanı kontrol (Controllers) katmanı olmadan bir anlam ifade etmemektedir. Görünüm (Views) altındaki ev (Home) sayfasının kontrolü (Controller) `Controllers` altındaki `HomeController`'dır. Kontrol (Controllers) kısmının görünüm (Views) kısmı ile birebir eşleşmesi gerekmektedir. Sayfaya taşınan bilgiler model ismi verilen sınıf (Class)'lar yardımıyla düzenli hale getirilmiştir. `Get` ve `Post` işlemlerinde sınıf döndürüp sınıf alma işlemi gerçekleştirilmektedir.

ASP.Net MVC E-Ticaret projesinin veri katmanın hazırlanmasında kullanılan varlık (Entity)'ları `Eticaret.Core.Model` katmanı içerisinde tanımlanmıştır. Her tabloda olması gereken alanlar için `EntityBase.cs` sınıfı oluşturarak diğer sınıfların bu alandan miras alması sağlanmıştır. Veri (Data) katmanına ait görünüm Şekil 3.'te verilmiştir.



Şekil 3. ASP.Net MVC E-Ticaret projesinin veri katmanı ekranı 1

Varlık (Entity) isminde bir dizin oluşturulmuş ve varlık dizini üstünde sağ tuşa basılarak sırası ile projede kullanılacak sınıflar oluşturulmuştur. Projede veri erişimi sağlanırken varlık çerçevesi kullanılmıştır. "`Package Manager Console`" üzerinden "`enable-migrations`" yazılarak göç (Migrations) yapısını aktifleştirilmiştir.

Varlık sınıfları varlık çerçevesi getirerek bu nesneler üzerinden veri erişimi sağlanmıştır. Varlık çerçevesi verilerin getirilmesini veya gönderilen verilerin veritabanına kaydedilmesini sağlamaktadır.

Çalışmada kullanılan varlık çerçevesi Codefirst (kod öncelikli) yaklaşımı içermektedir. *Codefirst* yaklaşımında kodlar tasarlanmıştır. Kod üzerinde ne tür tablolar olacağı hangi varlıklar olacağı belirlenmiş ve veri tabanının otomatik olarak oluşturulması sağlanmıştır.

Gereksinimlerin belirlenmesi ve veri varlıklarının oluşturulmasının ardından Mercurial uygulaması ile versiyon hesabı oluşturulmuştur. Bu işlem için Bitbucket (Bit kovası) depo oluşturmak ve sahte sunucu olarak çalışması için ayarlanmıştır. Bu işlemden sonra Visual Studio IDE (Integrated Development Environment) kullanarak web uygulaması oluşturulmuştur.

2.2.1. Bitbucket'te Depo Oluşturma

Bitbucket kullanmak için öncelikle ir hesap oluşturlmakıdır. Hesap oluşturmak için "https://bitbucket.org/" adresine gidilerek "`Ücretsiz kayıt olun`"> kullanılmıştır.

Bitbucket'te bir hesap oluşturduktan sonra, bir depo oluşturmak mümkündür. Bunu yapmak için ana menüden "`Depolar oluştur`" bölümüne gidilerek gerekli bilgiler girilmiş ve "`depo oluştur`" a tıklanmıştır.

Bir sunucudaki bir depo oluşturulduktan sonra Visual Studio IDE kullanılarak ASP.NET MVC projesi

oluşturma süreci başlamıştır. Visual Studio IDE'de, Dosya> Yeni> Proje> ASP.NET MVC Web Uygulaması seçilerek gerekli tüm bilgiler doldurulmuştur.

2.2.2. Yerel Depoyu Yapılandırma

Projeyi Visual Studio IDE'de oluşturduktan ve depodaki sunucuyu yapılandırdıktan sonra proje sunucuda depolanmaya hazırdır. Bunu yapmak için önce TortoiseHG'nin kurulması gerekmektedir. Windows için yürütülebilir dosyayı indirmek için http://tortoisehg.bitbucket.org/ adresi kullanılmıştır. Kurulum sonrası başlık altındaki gerekli tüm Mercurial dosyaları yüklenecektir. TortoiseHG yüklendiğinde MVC 5 projesinin oluşturulduğu klasöre sağ tıklayarak "burada depo oluştur" seçilir ve tamam'a basılır. Proje klasöründe ".hg" adlı bir dizin ve ".hgignore" adlı bir dosya oluşturacaktır. ".Hgignore" dosyası TortoiesHG'ye projeyi ilk kez sunucuya gönderirken hangi dosyaların yok sayılacağı hakkında bilgi vermektedir. Sonrasında proje dizinine fare ile tekrar sağ tıklanarak TortoiseHG seçilir. Senkronize et yapıldığında kullanıcı adı ve şifresini kullanarak daha önce oluşturulmuş hesap ile Bitbucket web sitesine yönlendirilir. Daha sonra Genel Bakış kısmında mevcut bir projem var tıklanır. Buradaki link Eşitle penceresine yapıştırılır. Bir takma ad verilerek kaydet tıklanır. Son olarak proje dizinine farenin sağ tuşu tıklanarak TortoiseHG'nin sunucuya göndermesi gereken tüm dosyaları takip ettiğini gösteren bir pencere açılacaktır. .Hgignore dosyasında belirtilen dosyalar izlenmemektedir. Ardından proje dizininde fare sağ tuşu tıklanarak hg workbench .s uzantılı tüm dosyalar Bitbucket sunucusuna gönderilir.

2.2.3. Servis Referansı Oluşturma

Bu çalışmada geliştirilen E-ticaret sayfası hizmetleri bir WCF sunucusu aracılığıyla yürütülmektedir. WCF sunucusu birlikte çalışabilir hizmetler oluşturmaya izin veren bir uygulamadır. Bir WCF sunucusundaki sınıflar ve arayüzler .NET platformuna özgü nitelikleri barındıracak şekilde oluşturulmaktadır. Visual Studio IDE içerisinde yer alan ana proje dosyasının altındaki çözüm gezgininde Servis Referansı bulunmaktadır. Bu araç kullanılarak referanslar projeye eklenmektedir. Servis referansına WCF ana bilgisayarının adresi yazıldığında sınıflar ve arayüzler otomatik olarak oluşturulmaktadır. Bu aşamadan sonra E-ticaret sayfasını kullanacak olan geliştiriciler ve harici kişiler sınıfların örneğini oluşturarak kullanabileceklerdir. WCF 443 nolu portunu kullanmaktadır. Bu portun kullanım durum bazı erişim hatalarına sebep olmaktadır. Bu problemin sebebi sertifika doğrulamasının yapılmamasından kaynaklanmaktadır. Güvenli iletişim için her zaman sertifika doğrulaması yapılması gereklidir. Kullanılan kod örneği Şekil 9.'da gösterilmiştir.

Şekil 4.'teki kod parçacığında ServicePointManager sınıfı kullanılmaktadır. Bu sınıfın ServerCertificateValidationCallback özelliği kullanılarak güvenli sertifika oluşturma ServerCertificateValidationCallback üzerinden sağlanmaktadır.

```
namespaceProductsWeb.Controllers
{
using System.Collections.Generic;
using System.Web.Mvc;
using Models;

publicclassHomeController : Controller
    {
// Return a view of the products
inventory.
public ActionResult Index(string
Identifier, string ProductName)
{

ServicePointManager.ServerCertificateVali
dationCallback +=
customXertificateValidation

            private static bool
customXertificateValidation(object
sender, X509Certificate cert, X509Chain
chain, SslPolicyErrors error)

var products = new List<Product>
                {new Product {Id =
Identifier, Name = ProductName}};
return View(products);
        }
     }
}
```

Şekil 4. WCF Bağlanma kodları

2.2.4. ASP.Net MVC E-Ticaret Projesi Entity Model Oluşturma

Veri sınıfı (Data Class) varlık çerçevesinin (Entity Framework) anlayacağı dile göre düzenlenmelidir. Sınıfların varlık çerçevesinin anlayacağı şekilde düzenlenmesi önemlidir. Bunun için o alanın tekil anahtar (primary key) olmasını sağlayan [KEY] özelliği (attribute) kullanılmıştır. Şekil 5.'te varlık model görünümü yer almaktadır.

Şekil 5. ASP.Net MVC E-Ticaret projesi EntityBase oluşturma ekranı

2.2.5. ASP.Net MVC E-Ticaret Projesi Veritabanı İşlemleri

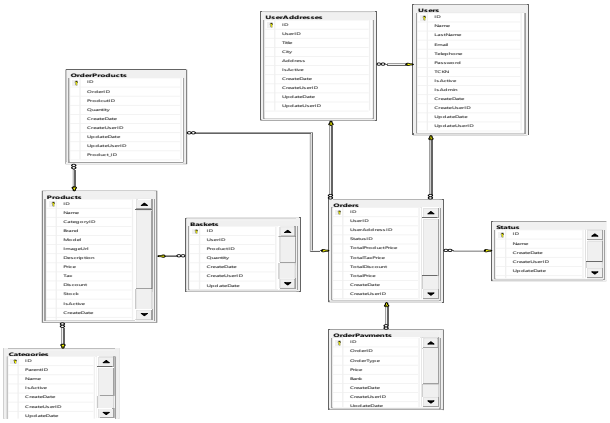
E-Ticaret web uygulaması geliştirilirken model nesneleri özelleştirilmiş ve verilerin getirilmesini sağlayan bir `context` tanımlaması yapılmıştır. `Eticaret.Core.Model` projesi içine de bir sınıf oluşturulmuştur. `DbContext` sınıfı Şekil 6.'da sunulmuştur.

Şekil 6. ASP.Net MVC E-Ticaret projesi `Dbcontext` sınıfı oluşturma ekranı

Bu çalışma kapsamında E-Ticaret web projesinde önemli olan bir noktada projenin veri tabanının nerede oluşturulacağı konusudur. Eğer herhangi bir ayar yapılmayarak proje çalıştırdığında Visual Studio kurulumu ile gelen yerel veri tabanında tabloları oluşturacaktır. Bunun önüne geçmek için veritabanı bağlantısı için gerekli olan `connectionstring` tanımlamaları yapılmıştır.

E-ticaret projesinde bulunan `Dbcontext` sınıfını test etmek için bir `Console` Projesi eklenmiştir. Eklenen projeye referans olarak `Eticaret.Core.Model` projesi eklenmiştir. Varlık çerçevesinde çalışan kodun test edilmesi amacıyla `Package Manager Console`'dan varlık çerçevesi referans olarak eklenmiş ve çalıştırıldığında veritabanı ve veritabanında bulunan tabloların otomatik oluşturulduğu gözlenmiştir.

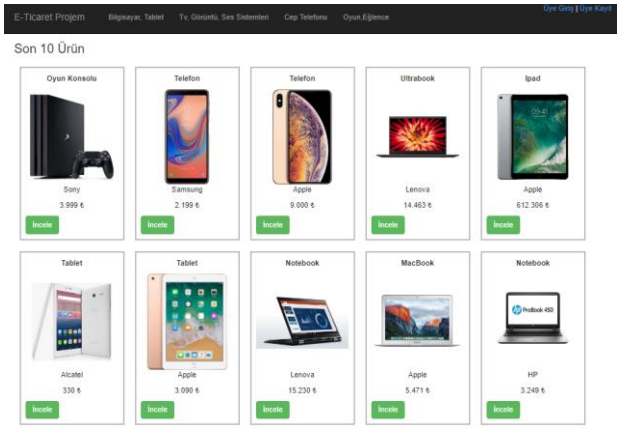
E-ticaret projesinde bulunan `Dbcontext` sınıfını test etmek için bir `Console` Projesi eklenmiştir. Eklenen projeye referans olarak `Eticaret.Core.Model` projesi eklenmiştir. Varlık çerçevesinde çalışan kodun test edilmesi amacıyla `Package Manager Console`'dan varlık çerçevesi referans olarak eklenmiş ve çalıştırıldığında veritabanı ve veritabanında bulunan tabloların otomatik oluşturulduğu gözlenmiştir. Veritabanı tasarımına ait UML (Unified Modelling Language) diyagramı Şekil 7.'de gösterilmiştir.



Şekil 7. ASP.Net MVC E-ticaret projesi UML diyagram ekranı

2.2.6. Model görünüm

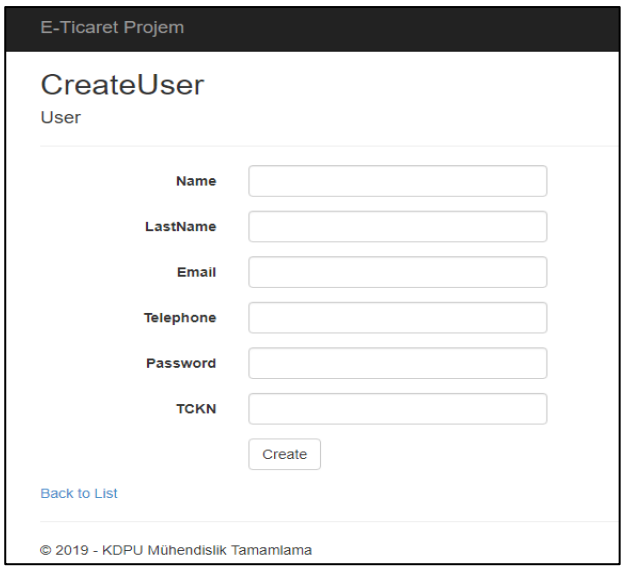
Projenin görünüm katmanı altında yer alan `Home -> Index.cshtml` sayfasına model sınıfı bağlanarak düzenlenmiştir. E-ticaret web projesinde, kontrol dizini içindeki `HomeController.cs` dosyası düzenlenmiş ve veritabanından bilgilerin okunması için `web.config` dosyasındaki `connectionstrings` ayarları yapılmıştır. Tasarıma ait görünüm Şekil 8.'de gösterilmiştir.



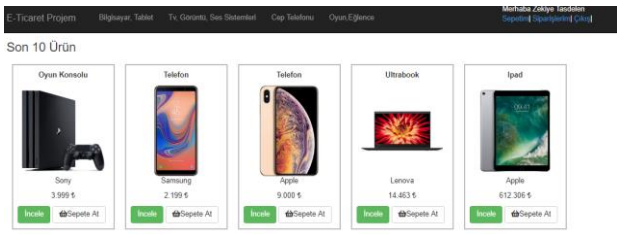
Şekil 8. ASP.Net MVC E-ticaret projesi anasayfa görünümü

Bu çalışma kapsamında örnek E-ticaret web uygulaması geliştirilirken, `Admin` alanı üzerinde yapılması gereken tüm işlemler gerçekleştirilmiştir. `Admin` yetkisine sahip olan kullanıcıların kategori ve ürünler ile ilgili ekledikleri verilerin veritabanına kaydedilmesi gerçekleştirilmiştir. E-ticaret web uygulamasında alışveriş yapmak için üye olan kullanıcılar veritabanına kaydedilmiş, kategori ve ürünlerin sergilenebilmesi için Web projesi kısmı geliştirilmiştir. Şekil 9.'da üye giriş (login) sayfası ve üye kayıt sayfası Şekil 10.'da gerçekleştirilmiştir. Kullanıcı giriş işlemi başarılı olduğunda ise Şekil 11.'deki kullanıcı sayfası gelmektedir.

Şekil 9. Üye giriş sayfası



Şekil 10. Üye kayıt sayfası



Şekil 11. Kullanıcı girişi başarılı ekranı

3. SONUÇLAR

Günümüzde insanların İnternete istedikleri zaman ulaşma özgürlüğü ve kolaylığı bulunmaktadır. Teknolojideki hızlı gelişim E-ticaret dünyasında da bazı yenilikleri ve gelişmeleri beraberinde getirmiştir. Bunun sonucunda E-ticaret siteleri hızlı, güvenilir ve kullanımı kolay olması beklenmektedir.

Bu çalışma ile ASP.NET MVC mimarisi ile varlık çerçevesi (Entity Framework) kullanarak bir E-ticaret web uygulama alt yapı sistemi gerçekleştirildi. MVC ve varlık çerçeve teknolojilerini birlikte kullanarak daha etkili işlemler gerçekleştirilmiştir. Ürünlerini web uygulamaları aracılığıyla web sitelerine yüklemek ve ürünlerini sunmak isteyen firmaların ve bu işle ilgilenecek görevlilerin işlevsel bir şekilde işlemlerini gerçekleştirebilmesi düşünülerek tasarım ve kodlama yapılmaktadır. Firmada bu işlerle ilgili web yöneticileri geliştirilen mimariyi kullanarak daha kolay ürünlerin bilgilerini, kategorilerini, resimleri ekleyerek düzenleyebilmektedirler.

Bu çalışma kapsamında günümüzde en çok kullanılan yazılım geliştirme tekniklerinden biri olan ASP.NET teknolojisi kullanılarak ürün satışının yapılabildiği E-ticaret amaçlı web sitelerinde bulunması gereken niteliklere sahip etkileşimli ve ileriye dönük genişletilebilir bir uygulama gerçekleştirilmiştir. Web tabanlı uygulama geleneksel ticaretten farklı bir değişimidir. Bu değişimlerle ilgili teknik adımlara ait karmaşık süreçlerin rahatlıkla gerçeklenebilmesi için ASP.NET'in katmanlı yapısı kullanılmıştır. Ziyaretçiler, üyeler ve yöneticiler kullanıcı gruplarının erişebileceği kaynaklar/sayfalar uygulama kapsamında görsellik ve işlevsellik olmak üzere iki farklı boyutta ele alınmıştır.

4. TARTIŞMA

Varlık çerçevesi üzerinde çalışan geliştirilen sistemle ilgili yapılan gereksinim analizi ve UML diyagramları ile tasarım farklı bir kullanıcının gelmesi durumunda bakım yapılabilecek şekilde tasarlanmıştır. Yeni eklenecek kodlar ve veritabanına yapılacak güncellemeler ile yapılan analiz sonuçları incelenebilecektir. Diğer taraftan geliştirilen İnternet sayfası versiyonlanabilir özelliği ile yeni bir versiyona geçildiğinde eski versiyondan yeni versiyona hızlıca geçiş yapılabilmektedir. Bu özellik güncellemede web sayfasının çalışmamasını (down) önlemektedir. Esnek (responsive) tasarıma sahip İnternet sayfası farklı cihazlardaki farklı ekran modellerine doğrudan yerleştirilebilmektedir. Bu durumda ürünlerin ekranda yer değiştirmesi ve belirle ekranların gösterilememesi problemlerine çözüm sunmaktadır. Panele giriş ve ürünlerin sepete eklenmelerinde kullanıcı girişleri yapılarak `codefirst` tasarımına sahip güvenliği öne çıkaran giriş ekranı (login) ile kullanıcı takibi ve güvenlik sağlanmaktadır.

KAYNAKLAR

[1] Ingebrigtsen, E. (2015). SignalR Blueprints. Packt Publishing Ltd.

[2] Telli, G. (2013). E-ticaret Kavramlar Gelişim Ve Uygulamalar, Kriter Yayınevi, İstanbul, Türkiye.

[3] Çetinkaya, Ş. (2016). E-ticaret Uygulamalarının Makro-ekonomik Göstergelere Etkisi SWOT Analizi ve Türkiye'de E-ticaret Gelişimi İçin Bir Eylem Planı Önerisi. Yalova Sosyal Bilimler Dergisi, 6(11), 235-256.

[4] Erdem, U. Ç. A. R., Altunsöğüt, Ö. (2009). Asp. Net Teknolojisini Kullanarak Bir Satın Alma Portalı Uygulaması Geliştirilmesi. Trakya Üniversitesi Fen Bilimleri Dergisi, 10(2), 119-126.

[5] Penberthy, W. (2016). Beginning ASP. NET For Visual Studio 2015. John Wiley & Sons.

[6] Sarısakal, N., Uysal, M. (2001). Web Teknolojilerindeki Hızlı Gelişmelerin Ve Web Programlama Araçlarının İncelenmesi. Istanbul University-Journal of Electrical & Electronics Engineering, 1(1), 6-16.

[7] Civelek, M. E. (2017). İşletmeden Tüketiciye (B2C) Elektronik Ticaret Alanında Faaliyet Gösteren İşletmelerin Web Sitelerini Yönetmelerinin Net Fayda Üzerine Etkisi: Kavramsal Model Önerisi. Doktora Tezi, İstanbul Ticaret Üniversitesi, Sosyal Bilimler Enstitüsü, İstanbul.

[8] Çiftçi, A. (2011). Yazılımda Alana Özgü Modelleme. Yüksek Lisans Tezi, Yıldız Teknik Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul.

[9] Althammer, E., Falkemeier, W. P., Joubert, G. R., Kao, O. (2001). Design and Implementation Of An MVC-based Architecture For E-commerce Applications. International Journal of Computers and Applications, 23(3), 173-185.

[10] Ping, Y., Kontogiannis, K., & Lau, T. C. (2003, September). Transforming Legacy Web Applications to The MVC Architecture. In Eleventh Annual International Workshop on Software Technology and Engineering Practice (pp. 133-142).

[11] Althammer, E., Falkemeier, W. P., Joubert, G. R., Kao, O. (2001). Design and Implementation Of An MVC-based Architecture For E-commerce Applications. International Journal of Computers and Applications, 23(3), 173-185.

[12] Ciliberti, J. (2013). Test-Driven Development With ASP. NET MVC 4. In ASP. NET MVC 4 Recipes (pp. 321-373). Apress, Berkeley, CA.

[13] Garschhammer, M., Hauck, R., Hegering, H. G., Kempter, B., Radisic, I., Rolle, H., Schmiddt, H., Hegering, H.G., Nerb, M. (2001). Towards Generic Service Management Concepts A Service Model Based Approach. In 2001 IEEE/IFIP International Symposium on Integrated Network Management Proceedings. (pp. 719-732).

[14] Kırbaş, S., Şen, A. (2013). A. Yazılım Depoları Madenciliği ile Endüstriyel Yazılım Evrimi İncelemesi. Ulusal Yazılım Mühendisliği Sempozyumu (UYMS).

[15] Sosa, A. S. (2014). A Modified Securtiy Approach For Web Application Used For The Registration Of The Student At Çankaya University. Master Thesis, Cankaya University, The Graduate School of Natural and Applied Sciences, Ankara.

[16] Armutlu, H., Armutlu, Ş., Akçay, M. (2012). İyi Bir Web Sitesi Nasıl Yapılır? XIV. Akademik Bilişim Konferansı.

[17] Xie, C., & He, D. (2020). Research on the Construction of E-Commerce Precision Poverty Alleviation System Based on Geographic Information. Procedia Computer Science, 166, 111-114.