T.C.
HALİÇ ÜNİVERSİTESİ
FEN BİLİMLER ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
YÖNETİM BİLİŞİM SİSTEMLERİ PROGRAMI

WEB TABANLI BİLGİ PAYLAŞIM PLATFORMU

YÜKSEK LİSANS TEZİ

Hazırlayan
Okan YAYLAGÜL

Danışmanı
Yrd.Doç.Dr.Osman ALİEFENDİOĞLU

İstanbul – 2010

T.C.
HALİÇ ÜNİVERSİTESİ
FEN BİLİMLER ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
YÖNETİM BİLİŞİM SİSTEMLERİ PROGRAMI

WEB TABANLI BİLGİ PAYLAŞIM PLATFORMU

YÜKSEK LİSANS TEZİ

Hazırlayan

Okan YAYLAGÜL

Danışmanı

Yrd.Doç.Dr.Osman ALİEFENDİOĞLU

İstanbul – 2010

T.C.

HALİÇ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ MÜDÜRLÜĞÜNE

Bilgisayar Mühendisliği Anabilim Dalı Yönetim Bilişim Sistemleri Programı Tezli Yüksek Lisans öğrencisi **Okan YAYLAGÜL** tarafından hazırlanan **"Web Tabanlı Bilgi Paylaşım Platformu"** adlı bu çalışma jürimizce Yüksek Lisans Tezi olarak kabul edilmiştir.

Sınav Tarihi : 01.07.2010

(Jüri Üyesinin Ünvanı , Adı , Soyadı ve Kurumu) : İmzası :

Jüri Üyesi : Yrd.Doç.Dr.Osman ALİEFENDİOĞLU .....
Danışman-HAL.Üniv.Bilg. Müh.ABD Öğr.Üyesi

Jüri Üyesi : Prof.Dr.Haluk GÜMÜŞKAYA .....
HAL.Üniv.Bilgisayar Mühendisliği ABD Öğr.Üyesi

Jüri Üyesi : Yrd.Doç.Dr.Vehbi BÖLAT .....
HAL.Üniv. Elek.ve Hab.Müh. ABD Öğr.Üyesi

ÖNSÖZ

Yüksek lisans eğitimim boyunca desteğini esirgemeyen Haliç Üniversitesi Bilgisayar Mühendisliği Bölüm Başkanı Prof. Dr. Haluk GÜMÜŞKAYA'ya, danışmanım Haliç Üniversitesi Öğretim Üyesi Yrd. Doç. Dr. Osman ALİEFENDİOĞLU'na, teknolojik araştırmalarıma eşlik eden ve desteğini esirgemeyen dostum, Bilg. Müh. Ali KARAYEL'e ayrıca çalışamalarım sürecinde sonsuz sabır gösteren eşim, Hülya YAYLAGÜL ve oğlum Ali Sadettin YAYLAGÜL'e sonsuz teşekkürlerimi sunuyorum.

Antalya, 2010 Okan YAYLAGÜL

İÇİNDEKİLER

KISALTMALAR

MVC : Model View Controller (model, görünüm, denetleyici)

VEK : Veriye Erişim Katmanı

İMK : İş Mantığı Katmanı

UMK : Uygulama Mantığı Katmanı

HTML : Hyper Text Markup Language (zengin metin işaret dili)

XHTML : Extensible HyperText Markup Language (genişletilebilir zengin metin işaret dili)

TOG : Test Odaklı Geliştirme

URL : Uniform Resource Locator (tekdüzen kaynak bulucu)

VB **:** Visual Basic

TKT : Temsili Konum Transferi

SEO : Search Engine Optimization (arama motoru uyumluluğu)

IIS : Internet Information Services (İnternet bilgi servisleri)

BVYS : Bağlantısal Veritabanı Yönetim Sistemi

LINQ **:** Language Integrity Query (dil bütünlük sorgusu)

SSL **:** Secure Socket Layer (güvenli yuva katmanı)

CSS **:** Cascading Style Sheet (geçişli stil sayfası)

TABLO LİSTESİ

ŞEKİL LİSTESİ

GENEL BİLGİLER

Adı ve Soyadı : Okan YAYLAGÜL

Anabilim Dalı : Bilgisayar Mühendisliği

Programı : Yönetim Bilişim Sistemleri

Tez Danışmanı : Yrd.Doç.Dr. Osman ALİEFENDİOĞLU

Tez Türü ve Tarihi : Yüksek Lisans – Haziran 2010

WEB TABANLI BİLGİ PAYLAŞIM PLATFORMU

ÖZET

Günümüzde, WEB ortamında bilginin paylaşılması çok yaygın hale gelmiştir. Araştırma yapmak isteyen kullanıcılar, kütüphanelere gitmek yerine öncelikle internet ortamında araştırma yapmakta, bilgisini paylaşmak isteyen insanlar yine internetten faydalanmayı tercih etmektedirler.

Bu tezde, kullanıcıların ihtiyaçları göz önünde bulundurularak, kullanıcıların ortak bir platformda buluştuğu, teknolojik olarak arama motorları tarafından içeriği indekslenebilen, site içi arama motoru bulunan, gerektiğinde arama terimleriyle ilgili kullanıcısına öneriler sunabilen web tabanlı bilgi paylaşım platformu geliştirilmiştir. Bu araştırma sürecinde, mevcut web geliştirme teknolojileri ve mimarileri incelenmiştir. Tam metin arama algoritmaları incelenmiş ve çalışmaya dahil edilmiştir. Ayrıca site içerisinde arama yapan kullanıcıların, bilgiye daha hızlı ulaşmalarını sağlayacak Fonetik algoritması da uygulamaya entegre edilmiştir.

Anahtar Kelimeler: Fonetik, Tam Metin Arama, MVC, Bilgi Paylaşım Platformu

GENERAL KNOWLEDGE

Name and Surname	: Okan YAYLAGÜL
Field	: Computer Engineering
Program	: Management Information Systems
Supervisor	: Assoc.Prof.Dr.Osman ALİEFENDİOĞLU
Degree Awarded and Date	: Master – July 2010

WEB BASED INFORMATION SHARING PLATFORM

ABSTRACT

Nowadays, sharing information via the internet has become very common. Users that would like to do a research prefer as a first choice to surf on the web instead of going to the library and also users that share their knowledge and findings over the web also perpetuate the same way

In this thesis, based on the user requirements, a web based information sharing platform has been developed with an active search engine where the content can indexed and suggestions are also offered to the users about their search criteria. During the development stage, instead of classical web forms, web development methods, Full Text Search algorithms and Phonetic algorithms that provides the user a faster response during their research were also integrated.

Keywords: Phonetics, MVC Design Pattern, Full-Text Search

1. GİRİŞ

Web ortamında paylaşılan bilgi ve bu bilgileri kullanan insan sayısı her geçen gün artmaktadır. Öyle ki insanlar kendi işleriyle ilgili karşılaştıkları teknik problemleri forumlara yazmakta ve destek almaya çalışmaktadırlar. Çok farklı konularda forum ve makale yayınlanmasına olanak sağlayan siteler bulunmakta ve sayıları her geçen gün artmaktadır. Mevcut siteler incelendiğinde teknolojileri, mimari ve teknik alt yapıları, kullanıcı arayüzleri, güvenlik yapılandırmaları, arama motorları tarafından ulaşılabilme başarıları, site içi arama metodları, görsel tasarımları ve performansları büyük ölçüde farklılık göstermektedir.

Bu tez sürecinde, web tabanlı bilgi paylaşım platformu uygulaması açısından gereksinimler incelenmiştir. Başlıca hedef, web ortamında geliştirilmesi muhtemel bu tür uygulamalar için mimari bir kılavuz oluşturmaktır. Bu hedef doğrultusunda, mevcut uygulamalar ve teknolojileri araştırılmış, web tabanlı uygulama geliştirme mimarileri incelenmiştir. Geliştirilen TeknoDestek projesi için, model-view-controller mimari yapısı ile klasik web formlar kıyaslanmış ve bir tercih yapılmıştır. Veritabanı yönetimi için ayrıca nesneye yönelimli bir yapı tercih edilmiştir. Kullanıcı arayüzleri üzerine çalışma yapılmış ve kullanıcılar üzerindeki etkileri de göz önünde bulundurularak kullanıcı dostu bir arayüz tasarlanmıştır. Arama motorları tarafından içeriği indekslenebilir html formları oluşturulmuş ve tam metin arama algoritmaları kullanılarak site içi arama motoru uygulamaya entegre edilmiştir. Fonetik algoritmalar üzerinde araştırma yapıldıktan sonra, kullanıcılara önerilerde bulunabilen fonksiyonlar eklenmiş ve arama seçenekleri genişletilmiştir.

2. PROJENİN TASARIMI VE MİMARİ YAPISI

2.1.Projeye Genel Bakış

Bu Tez, okuyucalara web uygulaması alanında günümüz teknolojilisinde ihtiyaç duyacakları bir çok metod ve yöntem içeren Asp.NET MVC [2] kullanılarak geliştirilmiş Bilgi Paylaşım Platformunun yazılım inşasını anlatmaktadır. İçerdiği modüller ve genel özellikler aşağıda sıralanmıştır;

- Kullanıcı Hesap yönetimi
- Makaleler
- Anketler
- Haberler
- Forum
- Tam metin arama ve Fonetik algoritmalara göre arama metodları.
- Yönetim paneli (Uygulama içerisinde bulunan tüm verilerin yönetimini sağlamak amacıyla geliştirilmiş yönetici paneli)

Yukarıda bildirilen modüller geliştirilirken aşağıdaki teknolojiler kullanılmıştır;

- Model konsepti ve yapısı
- Görünüm (View)
- View MasterPage, ViewPage ve görünüm denetleyici
- View Data, Temp Data'nın denetleyiciden görünüm elementine taşınması
- HTML ve Ajax metodları
- JQuery ve MVC entegrasyonu
- Denetleyici (Controller)
- Denetleyici Aksiyonlar (Controller Action)
- Uygulamayı güçlendirmek için Aksiyon Filtreler, Aksiyon Sonuçlar, ve Aksiyon Seçicilerin kullanımı.

- Tam metin arama teknikleri
- Fonetik Algoritmalar

Ayrıca;

- LINQ, LINQtoSQL
- Dahili metodlar (Extension Methods)
- Anonim metodlar (Anonymous)
- Üyelik Yönetimi nesnesi (Membership Management) ve Profil (Profile) nesnesi kullanılmıştır.

Uygulama geliştirmeye başlamak için yüklenmesi gereken programlar;

- Visual Web Developer 2010
  http://www.microsoft.com/express/download/
- Sql Server 2008
  http://www.microsoft.com/express/sql/download/
- Microsoft .NET Framework 4.0
  http://www.microsoft.com/net/
- Sql Server 2008 Management Studio Express(SSMSE)
  http://www.microsoft.com/express/sql/download/
- Microsoft ASP.NET MVC
  http://www.asp.net/mvc/

2.2.Site Tasarımı

Yeni bir web sitesi oluştururken yapılması gereken ilk iş, sitenin görsel ve işlevsel tasarımını oluşturmaktır [3]. Menülerin yerleşeceği bölüm, içeriğin görüntüleneceği bölüm, alt menüler ve diğer navigasyonların yetleştirileceği bölümler önceden tasarlanmalı ve site tasarımı bu düzene göre yapılmalıdır. Eğer bu işlemler önceden yapılmazsa, uygulama geliştirme esnasında sitenin görsel yapısı önceden planlanmadığı için sonradan değişiklik yapmak ve mevcut içeriği, sonradan oluşturulacak bir yapıya uydurmak çok daha zorlu olacaktır. Üstelik yazılımcı açısından sürekli olarak görsel yapıda değişiklik yapma gerekliliği oluşturabileceğinden aynı işleri tekrar tekrar yapmak büyük sıkıntı yaratacaktır.

Aslında bir web sitesinin görnümünü tasarlamak çok kolay bir iş gibi görünse de, oldukça zor ve önem gerektiren bir iştir. İçeriğin fazla karmaşık görünmemesi, butonların dikkat çekici biryerde olması ve çabuk ulaşılması, içeriğin temiz olması, kullanıcının uygulamadan hem azami verim almasını sağlar hem de site içerisinde gezinirken kullanım sorunları nedeniyle siteden sıkılmasını engelleyecektir. Aksi bir durumun söz konusu olması, kullanıcının siteden kötü izlenimlerle ayrılmasına ve siteye tekrar girmek istememesine neden olacaktır. Üstelik uygulama, teknolojik açıdan nekadar başarılı olursa olsun kullanıcının gitmesine engel olamayacaktır. Çünkü kullanıcılar kaynak kodların nekadar temiz ve iyi düzenlenmiş olduğunu göremezler, çok iyi tasarlanmış bir iş mantık katmanını (İMK) göremezler veya son derece normalizasyon kurallarına uyulmuş bir veritabanını göremezler bu nedenle öncelikli olarak görmeleri gereken şey, temiz görünümlü ve kullanışlı bir arayüz olmalıdır. Bu özellikler kullanıcının siteye olan güvenini artıracaktır. Sitenin görsel tasarımı tamamlandıktan sonra bir diğer önemli nokta da, web sitesinin farklı tarayıcılarda aynı şekilde çalışabilmesidir.

ASP.NET, web sunucu üzerinde çalışır ve .NET iskelet sisteminin sağladığı fonksiyonelliklerden faydalanır. ASP.NET kullanıcının bilgisayarında çalışmak yerine, dinamik olarak elementleri kullanarak tarayıcının kullanabileceği bir sayfa oluşturur. Bu elementler sunucu tarafından tarayıcıya gönderilir. Elementler, Html, resimler, JavaScript, Jquery [8] ve geçişli stil sayfalarından (Cascading Style Sheets, CSS) olşur [15]. CSS'ler HTML döküman ve nesneleri için renkleri, boyutları, nesnelerin yerleşeceği yerleri belirleyen tasarım dosyalarıdır.

HTML bir çok farklı yolla düzenlenebilir. Asp.NET içerisinde bulunan görsel form tasarlayıcısı kullanılalarak sürükle bırak yönetimi gibi görsel yöntemlerle düzenlenebilir. Ancak bu yöntem ASP.NET MVC'de tavsiye edilmemektedir çünkü ASP.NET MVC, HTML üzerinde daha fazla kontrole sahiptir. CSS kullanılarak HTML kodlarıyla oluşturmak çok daha profesyonel bir arayüz tasarımı için en uygun yol olacaktır.

ASP.NET MVC'de kod arkası (code-behind) bölümü kaldırılmış ve Denetleyici nesnesine taşınmıştır. Bu ayrım görünüm ekranlarının yönetimini kolaylaştırmış ve kopyala yapıştır yöntemiyle çoğaltılmasını kolay hale getirmiştir.

Unutulmamalıdır ki; sitenin dış görünüşü çok önemlidir. CSS kullanımı ve grafik tasarımı başlı başına bir eğitim ve çalışma gerektirir. Bu bilgiler elbette bir maliyet demektir. Internet üzerinde bir çok sitede tasarımlar satılmakta veya ücretsiz

olarak dağıtılmaktadır. Maliyetten kaçınmak adına hazır tasarım satın almak iyi bir çözüm olabilir. Free CSS Templates(www.freecsstemplates.org) adresinden CSS tabanlı profesyonel ve şık tasarımlar indirilebilir. Template Monster (www.templatemonster.com) adresi aracılığıyla da CSS tasarımları indirilip, kullanılabilir.

2.2.1. Stil Tasarımında CSS Kullanımı

Bu tezde CSS kullanımı hakkında çok detaylı bilgi verilmeyecek olmasına karşın genel kavramı ve temel bazı kullanım özellikleri anlatılacaktır [15]. Ayrıntılı bilgi için CSS hakkında daha detaylı kaynak taraması yapılması tavsiye edilir. HTML etiketleri içinde kullanılan nesnelerin font büyüklüğü, renkleri gibi özelliklerini her nesne için ayrı ayrı HTML etiketleri içerisinde belirleyebilme imkanı olsa da bu özellikleri nesnelerin sınıf isimlerine veya ID isimlerine göre belirleyip farklı bir dosyada saklanabilmektedir. Bazen bu nesnelerin aşağıda gösterilen örnekte ki gibi HTML sayfası içinde düzenlendiği görülmektedir.

```
<div style="align: justify;
     color: red;
     background-color:
     yellow;
font-size:12px;"> içerik bilgi </div>
```

Bu kullanım şekli tavsiye edilmemektedir. Çünkü tüm site içerisinde bu şekilde kullanıldığı varsayılırsa, site tasarımında herhangi bir değişiklik yapılmak istendiğinde tüm web sayfalarında bulunan nesneler için ayrı ayrı düzenleme yapılması gerekecektir. Bunu engellemek için stil ayarlarını web sayfası dosyalarından ayrılmış css uzantılı stil dosyalarında saklamak en doğru yöntemdir. Daha sonra bu css dosyaları için her web sayfasının en başına yazılacak kodla dahil edilmesi yeterlidir. Stil dosyası kullanıldığında, aynı özelliğe(renk, yazı stili, font ismi) sahip olması istenilen html nesnelerine aynı sınıf ismi verilir ve bu sınıf ismine sahip nesneler için stil dosyasında özellikler belirlenir. Stil dosyasında ilgili sınıfla ilgili yapılan tüm değişiklik, sitede bulunan ve aynı sınıf ismini kullanan tüm nesnelerin özelliklerini de değiştirmiş olacaktır. Bu kullanım şekli yazılımcıya çok

büyük zaman kazandırdığı gibi yalnızca stil dosyasını değiştirerek tüm sitenin görünümünü de değiştirmeyi mümkün kılar.
Aşağıdaki örnekte div nesnesi için CSS kullanım şekli gösterilmektedir. Stiller.css adında bir dosya oluşturulur ve içerisine aşağıdaki kod eklenir;

```
.benimstilim
{
align: justify;
color: red;
background-color: yellow;
font-size: 12px;
}
```

Daha sonra aşağıdaki satır, .aspx veya .html dosyasının en başına eklenmelidir. Bu kod stiller.css dosyasının, kodun eklendiği .aspx veya .html dosyasına dahil edilmesini sağlayacaktır.

```
<head>
<link href=”/Content/stiller.css” text=”text/css”
rel=”style sheet” />
</head>
```

Son olarak div nesnesinin adını, css dosyasının içerisinde belirlenen sınıf dosyasının adını yazmak yeterli olacaktır. Artık ilgili div, stiller.css içerisinde belirlenen özellikleri kazanmış olacak.

```
<div class=” benimstilim”> içerik yazısı </div>
```

Eğer stil, “.” İle başlıyorsa sınıf adını ifade eder ve tarayıcı bu adı taşıyan tüm nesnelere ilgili stil ayarlarını uygular. Eğer türlerine göre tüm HTML nesneleri için bir stil tanımlanmak istenirse <p>, <body> gibi etiket adını yazmak yeterlidir. Örneğin aşağıdaki stil, sınıf ismi veya id ismi gözetmeksizin tüm body ve p nesnelerine uygulanır.

```
Body {
margin: 0px;
```

```
font-family: Verdana;
font-size: 12px; }

p {
align: justify;
text-size: 10px;
}
```

Stil "#" ile başlarsa HTML nesnesinin ID ismine göre uygulanır.

```
#baslik {
padding: 0px;
margin: 0px;
width: 100%;
height: 184px;
background-image: url(resimler/baslik.gif);
}
```

Html sayfasına id ismi "baslik" olan bir div eklenirse stil özelliği yukarda belirtildiği gibi olacaktır. Aşağıdaki örnekte görülmektedir.

```
<div id="baslik">içerik yazısı</div>
```

2.3.Mimari Tasarım

Bu tezde geliştirilmiş olan uygulama içerisinde, anket, makaleler, haberler ve forum gibi bir çok uygulama bulunmaktadır. Bir web uygulamasında bukadar çok uygulama bulunması, tasarım açısından önemli sorunları da beraberinde getirir. Mimari tasarım güçlü olmadığı taktirde uygulamanın test süreci, bakımı ve hata ayıklama süreci zorlu geçecektir [1].

Klasik Web Form yapısında tüm uygulamayı, iş mantığı katmanı, kullanıcı arayüz ve veriye erişim katmanınlarına tamamiyle ayırmak mümkün olmamaktadır. Çünkü ASP.NET, sürükle bırak yöntemiyle kullanılabilen bir çok sunucu taraflı kontrolü barındırır. Bu kontrollerden örnek olarak GridView nesnesine ekstra metod yazabilmek için iş mantığı katmanına yazılması gereken kodlar, kullanıcı arayüz

kısmına yazılmaktadır. Örneğin GridView için sıralama yapmak veya filtre uygulatmak için bir butonun tıklanma olayına kod yazmak gerekir. MVC sisteminin geliştirilmesiyle birlikte tüm bu yapı değiştirilmiştir. Kullanıcı arayüz ve iş mantığı katmanı tamamen ayrılarak, kullanıcı arayüzü içinde bulunan kod arkası (code-behind) tamamen kaldırılmıştır. Bu sayede artık çok katmanlı (n-tier,multi-tier) web uygulamaları geliştirebilmek de mümkün olmaktadır.

Teknodestek web uygulamasında;

- Veriye Erişim Katmanı (Data acces layer), iş mantığı katmanı ve kullanıcı arayüz katmanı kodları biribirlerinden ayrılmış durumdadır. Bu yapı, uygulamanın teknik bakımının, iyileştirme ve hata ayıklama işlemlerinin çok daha rahat yapılabilmesine olanak sağlamaktadır. Bu yapıya kısaca çok katmanlı tasarım denmektedir.
- Veriye erişim mimarisi uygulamadan tamamen ayrılmıştır. Bu özellik farklı veritabanlarının kullanılabilmesine olanak sağlar. Uygulamada SQL Server 2008 kullanılmışken, iş mantığı katmanında değişikliğe gidilmeden Oracle gibi farklı bir veritabanı sistemine geçilebilir. Bu yapıya, kademelerin ayrımı (decoupling the tiers) adı verilmektedir.
- İş Mantığı Katmanı (İMK) dataları, Veriye Erişim Katmanı (VEK) tarafından nesneye yönelimli biçimde getirilmektedir. Bu yapı, ilişkisel veri haritalama süreci (process of mapping relational data) adıyla anılmaktadır.
- İMK, ön bellekleme metoduyla desteklenmiştir. Bu işlem, sürekli kullanılan verileri ve gereksiz veritabanı bağlantısı yaratan işlemlerde kullanılan verileri önbellekte saklamaya yarar. Bu sayede işlemci kullanımı, veritabanı kaynakları ve ağ trafiği azalır böylece çok daha başarılı genel bir performans sağlar.

2.4.Uygulama Katmanlarının Tasarımı ve Çok Katmanlı Yapı (N-Tier)

Özellikle son yıllarda geliştirilen projelerde duyulan çok katmanlı (multi-tier, n-tier) [1] yazılım tasarım sözcüğü bir kaç kelimeyle açıklanacak olursa, uygulamanın katmanlara göre tasarlanmasını ifade etmektedir. Uygulama tasarımları genellikle 4 veya 5 katmana ayrılırlar;

- **Veri saklama:** Verilerin ait olduğu yeri ifade eder. Bu ilişkisel bir veritabanı olabilir, XML olabilir, bir .txt dosyası veya verilerin sakalanabileceği farklı bir veritabanı yapısı olabilir.
- **Veriye Erişim Katmanı (VEK):** Verilerin, veri saklama katmanından getirilmesi ve manipule edilme işlemlerinin yürütüldüğü katmandır. Ayrıca veri saklama katmanındaki veriyi, "veri saklama şeması" gibi çok küçük detaylarla uygulamanın kullanımına sunar, gerekirse doğrulama bilgilerini hazırlar ve iş mantık katmanının kullanımına hazırlar.
- **İş Mantık Katmanı (İMK):** Bu katmandaki kodlar, iş kurallarının ve bilgi alanı nesnelerinin oluşturulmasından sorumludur.
- **Uygulama Mantığı Katmanı (UMK):** Kullanıcı arayüz katmanıyla iş mantık katmanı arasındaki etkileşimi sağlayan kodları içerir. Bu tür kodlara örnek olarak klasik web form'larda bulunan kod arkası gösterilebilir. Kod arkası, butonlara tıklama, post back gibi işlemleri idare eder.
- **Kullanıcı Arayüz Katmanı:** Kullanıcıya sunulması gereken ekran görürnümünün ve kodların tanımlandığı katmandır. Kullanıcı arayüz ve kullanıcı deneyimleri, menüler ve diğer site navigasyonlarını da içerir.

Asp.NET MVC iskelet sisteminde, çok katmanlı bir uygulama şu şekilde katmanlara ayırılabilir;
İş mantığı katmanı ve veri erişim katmanı Model içerisine, uygulama mantığı katmanı Denetleyici (controller) içerisine, kullanıcı arayüz katmanı da Görünüm (view) içerisine yerleştirilerek tasarlanabilir.

Şekil 2-1 Çok katmanlı yapı

Bu tez çalışmasında yukarıda belirtilen tasarım temel alınarak TeknoDestek adında web uygulaması geliştirilmiştir. Temel alınan tasarım şekli, web platformu uygulamasının gereksinimleri göz önünde bulundurularak tercih edilmiştir. Farklı projeler için daha farklı uygulama katmanları tasarlanabilir. Örneğin çok daha giriş

seviyesinde ve basit bir uygulama için uygulama mantığı katmanı ve iş mantığı katmanı birleştirilebilir.

Araştırmanın ilerleyen bölümlerinde Web Form ve MVC çok katmanlı uygulama arasındaki katman ve tasarım farklılıklarıyla ilgili örnekler ve çalışmalar anlatılacaktır. Konu daha ayrıntılı ele alınarak iki iskelet yapı arasındaki farklar açıkca anlaşılabilir olarak örneklerle ifade edilecektir. Asp.NET MVC iskelet sisteminin önemli farklarından birisi de uygulama içerisinde yapılan küçük değişikliklerden sonra tüm uygulamanın tekrar derlenmesine gerek kalmadan yalnızca değişiklik yapılan modül derlenerek çalışabilmesidir. Bu işlem, tüm uygulamanın gereksiz yere sürekli olarak derlenmesinin önüne geçerek zaman açısından önemli avantaj kazandırır.

2.5.Veri Saklama Yerinin Seçimi (Veritabanı Seçimi)

TeknoDestek web uygulamasında, veriye erişim katmanı geliştirilirken esnekliğe büyük önem verilmiştir. Farklı veritabanı seçimine olanak sağlanması hedeflenmiştir. Bu sayede farklı veritabanları seçilebilir veya uygulamada, veriye erişim katmanı hariç, hiç bir katmanda değişikliğe gerek olmadan farkı tipte veritabanına geçilebilir. Elbette birden fazla veritabanı desteği sağlamak yine de ciddi bir çalışma gerektiren alt yapı ve zaman gerektirir [1].

Hangi veritabanı en iyisidir? Aslında en iyisi sorusuna cevap aramak yerine, geliştirilecek olan projeye en uygun veritabanı hangisidir sorusu üzerinde durmak daha önemlidir. Eğer içerisindeki veriler sabit olan ve çok nadiren değişen bir uygulama geliştirilecekse XML seçilebilir. Acces Database'i daha çok masaüstü uygulamalarında, çok fazla kullanıcısı olmayan uygulamalar ve daha az yer kaplayan veritabanı gereksinimlerinde tercih edilebilir. Eğer ihtiyacı biraz daha yukarı taşımak gerekirse ozaman, modern bağlantısal veritabanı yönetim sistemleri (BVYS) tercih edilebilir. SQL Server, Oracle, DB2, MySQL, PostgreSQL v.b. uygulamalar iyi tercihler arasında sayılabilirler. Bu tezde yalnızca birtanesi üzerinde çalışılmıştır. Veritabanı olarak SQL Server 2008 tercih edilmiştir.

Gerçek hayatta, müşteriler ve kullanıcılar, uygulamayı geliştiren yazılımcıların tercihi yerine kendi tercih ettikleri farklı bir veritabanı kullanılmasını isteyebilirler. Bu durum tamamlanmış bir uygulama için veritabanını değiştirmek anlamına gelir. Aslında çok karşılaşılan bir durum olduğunu söylemek yanlış olmaz.

Ellerinde mevcut kullandıkları lisanslı bir veritabanı olabilir. Doğal olarak yeni bir veritabanı satın almak istemeyebilirler.

Bir çok .NET platformunda uygulama geliştiren yazılımcı, bir çok nedenden dolayı BVYS olarak SQL Server'ı tercih ediyor. Sebepler arasında, Visual Studio.NET ile SQL Server arasındaki entegrasyon kolaylığı başlıca nedenler arasında yer alıyor. Windows server ile SQL Server uyumu ve tüm uygulamaların lisanslarını tek firmadan tedarik etmenin getirdiği maliyet avantajları da bu nedenler arasında sayılabilir. Ayrıca SQL Server'ın LinqToSQL'i ve Entity Framework'u desteklemesi de bu avantajlar arasındaki önemli bir görevi üstleniyor.

Bu tezde geliştirilen uygulamada Microsoft SQL Server 2008 (Express Edition veya Enterprise, Standart) BVYS olarak tercih edilmiştir. SQL Server 2008 versiyonlarının tümü Visual Studio 2010 ile tam entegrasyon sağlamaktadır. Bağlantısal veritabanı diyagramları, tetikleyici (trigger), saklı yordam (stored procedure), tam metin arama (Full Text Search) gibi bir çok özelliği de beraberinde taşımaktadır.

2.6.Veriye Erişim Katmanının Tasarlanması

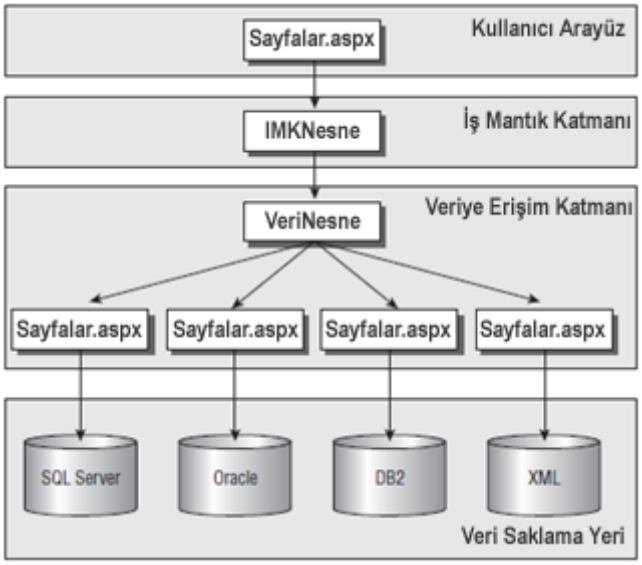
Veriye erişim katmanı [1], veritabanı sorgularını yürütüp verileri getirerek bir alt katmana ileten, ayrıca veritabanında silme, ekleme, güncelleme ve kaydetme işlemlerinin yapıldığı kodları içeren katmandır. Veritabanına en yakın katmandır. Veritabanıyla ilgili tablo şemaları, alan isimleri, saklı yordamlar ve görünümler (view) tamamiyle bu katman tarafından bilinirler ve kullanılabilirler. Veritabanını doğrudan ilgilendiren kodlar, aşağıdaki nedenlerden dolayı site sayfalarındaki kodlardan tamamen ayrılmalıdırlar ve doğrudan bu katmana yazılmalıdırlar. Nedenleri kısaca özetlenirse;

- Kullanıcı arayüz kodlarını geliştiren yazılımcı ile veriye erişim katmanının kodlarını geliştiren yazılımcılar farklı olabilirler. Aslında orta ölçekli ve büyük projelerde her katmanı tamamen farklı yazılımcılar kodlarlar. Arayüz tasarlayan yazılımcıların asli görevi ve uzmanlık alanı arayüzle ilgili olduğu için, veritabanını ilgilendiren bir çok işi göz ardı edebilirler. Ayrıca tabloların tüm ayrıntıları nesne haline çevrilmiş olduğu ve veritabanı katmanını ilgilendirdiği için bu durum oldukça da muhtemeldir.

- Bazen veriye erişim katmanı tarafından oluşturulan bir sorgu farklı sayfalar tarafından kullanılıyor olabilir. Eğer kullanıcı arayüz sayfasına doğrudan bir kod yazılırsa ve sonradan sorguda bir değişiklik olursa, örneğin bir tablo alan ismi değişirse veya sıralama kuralları değişirse, yazılımcının her sayfayı tek tek gözden geçirip yazdığı her kodu test etmesi ve değiştirmesi gerekir. Bunun yerine tüm işlemler veriye erişim katmanında tamamlanır ve sayfadan çağırılırsa, yalnızca veriye erişim katmanında değişiklik yapmak yeterli olacaktır. Arayüz sayfalarına dokunmaya gerek kalmaz.

- Kullanıcı arayüz katmanına doğrudan kod yazmak, yeni versiyon BVYS'ler tarafından desteklenmeyeceği için çok güçtür ve birden fazla BVYS kullanılması gereken durumlarda neredeyse tamamen angaryadırlar.

- MVC sisteminde veritabanını ilgilendiren veriye erişim katmanı kodlarının tümü model katmanı içerisinde kullanılmaktadırlar.



Şekil 2-2 Ugulama mimarisi

2.6.1. LINQ'nun veriye erişim katmanına etkisi

ASP.NET 3.5. Framework versiyonuyla birlikte tanıtılan, dil bütünlük sorgusu (Language Integrity Query, LINQ) özelliğiyle .NET yazılımcıları tanışmış oldular [2]. LINQ teknolojisi, diziler, sözlükler (Dictionary), XML ve veritabanı gibi neredeyse tüm kolleksiyon tipindeki sorguların tek bir yapı aracılığıyla

uygulanabilirliğine olanak sağlamak amacıyla geliştirildi. LINQ, ayrıca yazılımcılara .NET kodları içerisinde doğrudan SQL tipinde kod yazma olanağı da sağlar.

SQL

```
Select * from makaleler as c
```

LINQ

```
var makaleler = from c in datacontext.Makalelers
Select c;
```

Yukarıdaki örneklerde hem SQL'de hem de LINQ'da nasıl "select" işlemi yapılacağı gösterilmiştir. İki sorgu da aynı anlamdadır ve makaleler tablosunu listelemektedir. Ancak LINQ kullanımında dönen sorgu özel bir kolleksiyon olan Iqueryable<Makaleler> tipinde çağrılırmış olur ve içerisinde standart foreach ile dönülebilir. İlk bakışta "var" tipi henüz belirlenmemiş bir değişken gibi görünse de C# derleyicisi derleme anında değişken tipini kendiliğinden belirleyecektir ve doğru tipe dönüştürecektir. Bu tür kullanımda, SQL sağlayıcı tarzı bir kullanıma gerek olmaması, LINQ'nun pratiklik özelliklerinden birisidir. LINQ'nun en önemli avantajlarından birisi ise güvenli ve zengin sorgu olanaklarıdır.

LINQ kullanımında sorguda meydana gelen problem, derleyici tarafından hata kodu fırlatılarak bildirilir ve hatalı olan kod parçasının yeri gösterilir. Saklı yordamda olduğu gibi tüm sorgu hatalarında, tüm kodu baştan aşağı taramaya gerek kalmaz. Bu özellik yazılımcı açısından büyük kolaylık anlamına gelmektedir. LINQ ayrıca daha fazla üstünlük sağlamak için veritabanı sorgularında tablo bilgisini detaylarıyla birlikte getirir. Aşağıda kısa bir örnek verilmiştir.

```
Var makale = (from s in datacontext.Makalelers
              Where s.makaleID== 2
              Select s).FirstOrDefault();

Foreach (var makalegrupBilgisi in  makale.MakaleGruplari)
makalegrupBilgisi.kayitTarihi =  DateTime.Now;
```

```
dataContext.SubmitChanges();
```

Yukarıdaki kod örneğinde ilk önce ID numarası 2 olan makale kaydı seçildi. Sondaki FirsOrDefault() oldukça önemlidir çünkü, dönen değerin yalnızca tek kayıt veya "null" değer olacağını belirtir. LINQ, veritabanındaki ilişki değerlerini göz önünde bulundurarak makaleGruplari bilgisini de bu kayıtla birlikte getirir. Bu örnekte MakaleGruplari tablosundaki kayittarihi değeri değiştirilmiştir. DataContext.SubmitChanges(); kodu da değişikliklerin kaydedileceğini belirtir. Çünkü LINQ, transaction işlemini otomatik olarak yürüttüğü için, submitChanges() yapılmadığı taktirde değişiklikler veritabanına yazılmayacaktır.

2.6.2. LINQ-to-SQL

Model katmanında LINQ'yu kullanabilmek için Visual Studio 2010 tarafından sağlanan, LINQ-to-SQL tasarlayıcı kullanılabilir [2]. Bu tasarlayıcı SQL sunucusuna doğrudan bağlanarak tüm tabloları, ilişkileri, saklı yordam ve view'leri sürükle bırak yöntemiyle projeye taşınabilmesini mümkün kılar. Bu görsel tasarlayıcı, tüm veri erişim katmanını yalnızca bir kaç dakikada oluşturabilmeyi mümkün kılar. Üstelik tek satır bile kod yazılmasına ihtiyaç duymaz. SQL Server dışında Oracle, DB2, MySQL veya farklı veritabanı tiplerini Linq-to-SQL desteklememektedir. Ayrıca Linq-to-SQL veritabanındaki şemalarda meydana gelen değişiklikleri otomatik olarak güncellememektedir. Bu tür durumda görsel tasarlayıcıyı açıp, tüm nesneleri kaldırıp tekrar eklemek gerekmektedir. Yukarıda belirtilen iki durum, dezavantaj veya geliştirilmesi gereken özellikler olarak sayılabilir.

Not: LINQ-to-SQL Microsoft firmasının yazılım ekibi tarafından geliştirilmiş bir isim uzayıdır (namespace). Ayrıca Microsft ADO.NET ekibi tarafından geliştirilmiş olan versiyonuna da Entity Framework denmektedir.

2.6.3. Veriye Erişim Katmanının Güvenliği

Önceden, uygulama içerisinde kullanıcı arayüz katmanında doğrudan SQL kodu kullanılan durumlarda SQL enjeksiyon saldırısı sıkça rastlanan bir durumdu [16]. Örneğin web isteklerinde sorgu sicimi (quesryString) kullanımı veya postback içindeki metin kutuları problem teşkil edebiliyordu. Standart bir web isteği şu şekilde görünür;

http://www.orneksite.com/makaleler/goster.aspx?ID=22

Daha sonra sorgu siciminden bu ID numarası alınır ve doğrudan veritabanına şu şekilde gönderilir;

C#

```
String sqlText = "Select * from Makaleler";
     +="Where ID=" + QueryString["ID" ]+ "  ";
     +="and KayitDurum=1";
//Sql text'i çalıştır
```

Kod çalıştırıldığında veritabanına aşağıdaki gibi bir kod gönderilir ;

SQL

```
Select * from makaleler where ID=22 and kayitDurum=1
```

SQL'e doğrudan gönderilen metin üzerinde inceleme yapıldığında eğer sorgu siciminden dönen değer 22 olarak gönderilirse geriye 22 ID numaralı makale kaydı dönecektir. Böyle bir durumda herhangi bir sorun görünmüyor. Aslında bu tür kod yapısı temel uygulamalar içerisinde en çok kullanılan metod olarak ta bilinmektedir. Ancak bu metod, çok ciddi bir probleme davet göndermek gibidir. URL, şu şekilde düzenlenip kod tekrar gözden geçirilirse, tehlikenin boyutu da anlaşılabilir.

URL

http://www.orneksite.com/makaleler/göster.aspx?ID=32;+drop+table+makaleler;--

C#

```
String sqlText = "Select * from Makaleler";
     +="Where ID=" + QueryString["ID" ]+ "  ";
     +="and KayitDurum=1";
//Sql text'i çalıştır
```

SQL

```
Select * from makaleler where ID=32; drop table
Makaleler;-- and kayitDurum=1
```

Yukarıdaki sorgu sicimiyle (query string) gönderilen değerlerle birlikte SQL cümleciği çalıştırıldığında enteresan bir problem de beraberinde geliyor. Kod çalıştığında, gönderilen ID değerine göre ilgili makale kaydı geliyor ve hemen ardındaki kod satırında makaleler tablosu artık hiç bir veri içermeksizin veritabanından siliniyor. "and kayitDurum=1" kısmı da "--" ile açıklama haline getirildiği için hiç bir işe yaramıyor. Artı tek çare SQL server yedeklerinden eski kayıtlara geri dönmek.

LINQ, dönen sorguyu otomatik olarak kontrol eder ve bu tür saldırılara karşı kodları inceleyerek tehlikeli kodları elimine eder [13]. Diğer büyük bir sorun da bağlantı dizesi (connectionstring) bilgisine erişimin engellenememesidir. Bir çok sunucu sorgu sicimi değerinin, web config dosyasında güvenliğini sağlasa da en garanti yöntem güvenilir bağlantı (trusted connections) işlemidir. Bu işlemin en büyük avantajı bağlantı cümlesi içerisindeki şifre değerinin metin biçiminde yazılmamasıdır. Web config dosyasına hacker'lar tarafından erişilse de, veritabanı ele geçirilemeyecektir.

2.7.Uygulama Mantığı Katmanının Geliştirilmesi

Uygulamadaki veriye erişim katmanının, LINQ-to-SQL aracalığıyla oluşturulan sınıflardan meydana geldiği anlatılmıştı [1]. Bu sınıflar aracılığıyla veriler, veritabanından özel veri kolleksiyonu olarak getiriliyor veya tablo bilgileriyle birlikte getiriliyordu. Ancak dönen veri hala ham durumdadır. Veriler çok düzenli kolleksiyonlara dönüştürülmüş olsa da henüz uygulamaya özel hiç bir veritabanı işlemi kodu yazılmamıştır. Uygulama mantığı katmanı bu verileri işleyerek kullanıcı arayüz katmanına iletilmesi için yorumlar ve bu katmanla aradaki etkileşim görevini üstlenir. Ayrıca onaylama (validation) tanımlamaları bu katmanda yapılır. Kısmi sınıf (Partial class) metodları aracılığıyla veri ekleme, güncelleme, kayıt ve silme işlemleri için ekstra metodlar inşa eder ve veriye erişim katmanı tarafından oluşturulan sınıfların son halini alması sağlanır. Burdaki işlemler tamamlandığında veritabanı artık tamamen nesneye yönelimli olarak hazırlanmış olur.

2.7.1. Denetleyici (Controller)

Yeni bir MVC projesi geliştirmeye başlarken klasörlere göz atıldığında, "controllers" isminde bir klasör görülmektedir. Bu klasör içerisinde denetleyici tipindeki sınıflar oluşturulur [1]. Bu sınıflar, modelden gelen veri kolleksiyonlarını işlemek ve ilgili görünüme (View) yönlendirerek kullanıcı arayüzüyle model arasındaki koordinasyonu sağlamakla yükümlüdür. Denetleyici sınıfları sayesinde, klasik web formlarda kullanılan kod arkası (code-behind) tamamen kaldırılmıştır. Önceden kod arkası tarafından yürütülen tüm uygulama mantığı katmanı işlemleri artık denetleyici içerisinde yönetilmektedir. Denetleyicilerin avantajları kısaca özetlenecek olursa;

- Uygulama birim testleri çok daha kolay yapılabilmektedir. Bunun başlıca nedenleri, iş mantığı katmanının tamamen arayüzden ayrılmış olarak denetleyici içerisinde veya model içerisinde bulunmasıdır. Sunucu kontrollerinin tamamen kaldırılması da büyük katkı sağlar.
- Güvenlik kontrollerini uygulamak çok daha kolaydır çünkü tüm işlemler denetleyici ve model içerisinde yürütülmektedir. Kullanıcı arayüzü uygulama mantığı kodlarından arındırılmıştır.
- Birden fazla görünüm (view) için tek bir kontrol denetleyici sınıfı kullanılabilir. JSON [8] kullanılabilir.

2.7.2. Güvenlik

MVC kalıbında güvenlik ve yetki tanımlamaları için ilgili aksiyonun başına rol tanımlaması yapmak yeterlidir. Böylece ilgili aksiyonu gerçekleştirmeye yetkili olan kullanıcı rolü belirtilmiş olur [2].

```
[Authorize(Roles = "Admin")]
public ActionResult IcerikDuzenle(int id, ...)
{
return View();
}
```

Yukarıdaki örnek kodda, ilgili aksiyonu tetiklemeye yetkili olarak yalnızca "Admin" tanımlanmıştır. Doğrudan bir aspx sayfasına navigasyon özelliği de kaldırıldı. Çünkü yönlendirme motoru tüm url isteklerini devralır ve uygun

denetleyiciye kendisi yönlendirir. Bir kullanıcı, denetleyici katmanı üzerinden geçmeden doğrudan bir görünüme (view) ulaşamaz. Bu çok önemli bir özelliktir. Çünkü önceden özel bilgiler içeren web sayfalarına URL hacking teknikleri aracılığıyla dışarıdan erişilebilmesi mümkün olabiliyordu. Bu durum yazılım geliştiriciler ve web sitesi sahipleri açısından büyük risk teşkil ediyordu. Güvenlik için ayrıca önlemler almak ve özel teknikler kullanmak gerekiyordu. Sayfanın bulunduğu klasöre web.config tanımlaması yaparak rol isimlerine göre erişim yetkileri tanımlamak veya sayfa içerisine yetki kontrolleri yazmak gerekiyordu. Örneğin IcerikGoster.aspx sayfasını gezen bir ziyaretci IcerikDuzenle.aspx adında bir sayfa olabileceğini tahmin edip sayfaya ulaşmaya çalışabilir, yeterli teknik bilgiye sahipse bu durumda içerik bilgilerine ulaşıp düzenlemesi de muhtemeldir. URL yönlendirme ile bu durum da çözülmüş kabul edilmektedir.

2.7.3. Daha İyi Performans İçin Veriyi Önbellekleme (Caching Data)

Bir çok web sitesinde bazı veriler çok sık değişmemesine karşın kullanıcılar tarafından çok sıklıkla görüntülenirler. Örneğin makale kategorileri, ürün kategorileri, ülke isimleri gibi. Bu tür kullanımlarda, sitenin performansını artırmak için en başarılı çözüm yolu, veriyi önbellekleme yönetimi kullanarak sunmaktır [2]. Kullanıcıdan bir istek geldiğinde veri bir kez veritabanından bağlantı oluşturularak istenir, dönen veri ön belleğe yazılır. Eğer aynı istek herhangi bir kullanıcı tarafından tekrar gerçekleştirilirse bu defa veritabanına yeni bir bağlantı oluşturulmaksızın, bilgi doğrudan önbellekten okunur ve kullanıcıya iletilir. Bu işlem, işlemci kullanımını azaltır, ağ trafiğini düşürür ve kullanıcıya verinin daha hızlı iletilmesini sağlar. Bu sayede kullanıcılar, uygulama içerisinde herhangi bir işlem yaparken çok daha az zaman harcarlar. Önbellekleme, aynı zamanda sistemdeki yükü de azaltacağı için hem kullanıcı açısından hem de sistem kaynaklarının verimli kullanılması açısından önem arz etmektedir. Yine de önbellekleme yapılabilecek işlemler dikkatle seçilmeli ve kullanılmalıdır. Örneğin, çok sık değişen bir tablodan raporlanacak veriler için önbellekleme yapılması, raporların sağlıklı veri iletmesini engelleyebilir.

MVC, önbellek yönetiminin denetleyici seviyesinde yapılabilmesine imkan tanır;

```
[OutputCache(Duration = 60)]
public ActionResult HerhangiDenetleyiciMetod()
```

```
{
// kod yazılacak...
}
```

Yukarıdaki örnekte 60 saniyelik periyodla verinin ön bellekte saklanacağı bildirilmiştir. Ön bellek yönetimi ile ilgili parametreye bağlı veri saklamak gerekirse kullanımı şu şekildedir;

```
[OutputCache(Duration = 60, VaryByParam = "id")]
public ActionResult HerhangiDenetleyiciMetod (int id)
{
// kod yazılacak...
}
```

2.7.4. Sıradışı Durum, Hata Yönetimi

Bazen uygulama esnasında herşey doğru gitmez. Kod geliştirme aşamasında hatalı geliştirilen kodlar veya beklenmeyen bağlantı problemleri kullanıcıya hata mesajı fırlatılmasına neden olabilir [2]. Bu hataların kullanıcıya tarayıcının kötü görünümlü hata mesaj sayfasında doğrudan gösterilmesi yerine, meydana gelen hata yakalanabilir, hangi biçimde ve hangi ekranda kullanıcıya iletileceğinin seçilebilmesi sağlanabilir veya doğrudan mail yoluyla sisteme hata bildirimi bile yapılabilir. Aşağıda basit şekilde yazılmış hata yönetim kodu gösterilmiştir;

```
[HandleError]
public ActionResult HerhangiDenetleyiciMetod()
{
// kod yazılacak...
}
```

Aşağıdaki kod bloğunda, yalnızca güvenlik tipindeki hata mesajlarını yakalamak için yeni bir hata yönetimi tanımlanmıştır. Ayrıca kod, kullanıcıyı bir görünüme yönlendiriyor. Görünüm, muhtemelen kullanıcının yetkisi olmadığı için ilgili web isteğini gerçekleştiremeyeceğini bildiren yazılar içeriyor olacak.

```
[HandleError(ExceptionType = typeof(SecurityException),
View = "UnauthorizedView"]
public ActionResult HerhangiDenetleyiciMetod()
{
// kod yazılacak...
}
```

2.8.Kullanıcı Arayüz Katmanı

Bu katman, kullanıcının doğrudan gördüğü ve uygulamayla etkileşim sağladığı katmandır [1]. Diğer taraftan en önemli katmandır çünkü uygulamanın ünü ve başarısı kullanıcılar tarafından bu katmanın verimliliği ve pratikliğiyle doğru orantılı olarak belirlenir. Web kullanıcıları genelde siteye girdikten yalnızca birkaç dakika sonra site hakkındaki kararlarını verirler. Bu ilk izlenimle ya siteyi kullanmaya devam ederler, beğenirler ya da ayrılırlar ve birdaha kullanmazlar. Web sitesinin renklerindeki uyum, tasarımın netliği ve açık oluşu, site içeriğine ulaşımın yeteri kadar kolay oluşu ve menülerin yeteri kadar anlaşılır olması gibi fonksiyonlar, ilk izlenimi oluşturan başlıca parametrelerdendir.

Bu durum kısaca şöyle açıklanabilir, yeni birisiyle tanışan herhangi bir birey, yalnızca bir kaç dakika içerisinde henüz tanıştığı kişi hakkında net yorumlar yapabilir durumda olur. Saç rengi, kıyafetlerinin uyumu ve temizliği, konuşması, göz rengi ve fiziki yapısı hızlıca insan beyninde değerlendirilip ilk izlenimleri oluşturur.

2.8.1. Görünüm Katmanı İçin Mühendislikte Kullanılan En Başarılı Yöntemler

- Otomatik açılan pencerelerden(Po-up) kaçınılmalıdır. Otomatik açılan pencereler web sitesinin görünümünü engellerler ve kullanıcıyı sıkarlar. Büyük olasılıkla pop-up pencere, tarayıcıdaki engelleyiciler tarafından engelleneceği için kullanıcıya ayrıca rahatsızlık verirler ve zaten açılamayacakları için hedeflerine ulaşamazlar [3].
- Web sitesi yayınlanmadan önce mümkün olduğunca tüm farklı tarayıcılar tarafından test edilmelidir.
- Web tasarımında genel yerleşim için tablo kullanımından kesinlikle kaçınılmalıdır. Bu yöntem eskiden en çok kullanılan yöntemdi ancak son

dönemde tüm tarayıcılar tabloları farklı şekillerde görüntülüyorlar. Tablo kullanmak yerine genel yerleşim için “div” kullanılmalıdır.

- Tüm sayfanın post edilmesi gereken durumlarda Java Script kullanılmalıdır. Yalnızca kullanıcının farklı bir sayfaya gittiği durumlarda tam post back yapılmalıdır. Seçilen açılır kutular, doğrulamalar kaybolmamalı ve kullanıcı bu bilgileri tekrar tekrar girmek zorunda kalmamalıdır.

- Site tasarımında kullanılacak renkler çok dikkatli seçilmeli ve önceden tasarlanmalıdır. Örneğin kırmızı renk tehlikeyi ifade eder. Mavi ve tonları ise güvenlik ve rahatlık duygusunu tetikler. Tüm renklerin insan algıları üzerinde farklı etkileri olduğu bilinmektedir. Ancak renk körlüğü, yeşil ve kırmızı arasındaki farkın algılanmasını engellemektedir. Renk körlüğü olan insanlar ancak aradaki renklerin zıtlığını farkederler. Bu tür kullanıcılar için örneğin beyaz arka plan rengi üzerine siyah yazılar uygun bir seçim olabilir.

- İçeriği mantıksal ve önem derecesine dikkat ederek yerleştirmek önemlidir. Örneğin çok önemli olan içerik, sayfanın tam ortasına yerleştirilmeli ve navigasyon gibi daha az önemli bilgiler dış çerçeveye yakın bölgelere yerleştirilmelidir. Farklı anlamlara sahip içerikler muhakkak gözle görülür biçimde ayrılmalıdırlar. Bu işlem için div kenarlık renkleri kullanılabilir. En önemli bilgiler muhakkak ana sayfaya eklenmelidirler. Çünkü bir çok kullanıcı ilk 30 saniye içerisinde sitede faydalı bilgi bulamadığı taktirde ayrılır.

- Bölüm başlıkları muhakkak kullanılmalı, kalın fontlarla ve belirgin renklerle yazılmalıdır. Bu başlıklar sayesinde kullanıcılar, ilgili bölüm içerisinde ne olduğunu ilk bakışta anlayacak ve tümünü okumak için zaman kaybetmek yerine tüm uygulamayı kavramasına katkı sağlayacaktır.

- Kullanıcıya ön bilgi vermeden uzun süren işlemlerden kaçınılmalıdır. Uzun sürecek olan işlemlerde, ilerleme hakkında bilgi gösteren ilerleme çubuğunu

veya işlem adım sayısını kullanıcılara bildirmek gerekir. Çok yavaş ve tepki göstermeyen sayfalar da kullanıcıyı yorar ve uzaklaştırır.

2.8.2. Görünümler (Views)

Görünüm ekranı, model katmanının kullanıcıya sunulduğu kısımdır [2]. Aslında HTML kodlarının tarayıcı için dönüştürülmüş halidir. Klasik web formla neredeyse aynı durumdadır ancak klasik web formlarda olduğu gibi kod arkası dosyası yoktur. Bunun yerine tüm kod, denetleyici sınıfı içerisindedir. Peki şu soruları sormak muhtemel midir? Şu ana kadar bir sürü farklı firma tarafından da geliştirilmiş olan gridview, detailsview gibi onlarca sunucu kontrolüne ne olacak? Bu sorunun aslında bir iyi bir de kötü cevabı var. Kötü kısmı artık kullanılamaz olacaklar. İyi kısmı ise gerek de kalmayacak. Burda en önemli nokta, HTML içeriğinin çok daha temiz ve daha verimli oluşturuluyor olmasıdır.

```
<label for=”etiket”>Başlık</label><br />
<%= Html.TextBox(“Baslik”) %>
```

Bu kod, bir HTML TextBox nesnesinin oluşturuluşunu göstermektedir. Textbox nesnesinin ismi ViewData sözlüğünü işaret etmektedir. Bu isim, denetleyici katmanında karşılık bulur ve veriyi görünüm içerisinde gösterir. Ayrıca bir model nesnesi, görünüm sayfasının yazılan model (typed model) özelliği kullanılarak doğrudan sayfaya aktarılabilir. Aşağıda kullanım şekli gösterilmiştir.

```
<label for=”title”>Başlık</label><br />
<%= Html.TextBox(“Baslik”, ViewData.Model.Baslik) %>
```

Yukarıda gösterilen iki kodun da sonucu aynıdır. Aradaki tek fark birinde veri, denetleyici nesnesine bağlanarak gösterilir, diğerinde ise doğrudan modelden sayfa içerisine bir nesne çağırılarak gösterilir. HTML kaynak kodlarına bakılırsa, üretilen HTML kodunun iki kullanım şekli için de aynı olduğu görülebilir.

```
<label for=”baslik”>Başlık</label><br />
<input id=”Baslik” name=”Baslik”  type=”text” value=”Bir
Makale Adı” />
```

Visual Studio .NET'in klasik web formlardan, MVC'ye geri dönüş nedeni kısaca şu şekilde açıklanabilir. Öncelikle yazılımcılar HTML nesneleri üzerinde daha fazla kontrol sahibi olmak istediler. Bir sunucu kontrolü kullanıldığında HTML nesnesi gereksiz yere büyür ve çok yer kaplayan Java Script kodları içerir. Üstelik bir de web formlarının, windows form gibi çalışabilmelerini sağlayabilmek için ekstra Java Script kodları eklenir. Klasik web form olarak oluşturulmuş bir sayfanın HTML kaynak kodları görüntülenir ise durumun vehameti çok daha iyi anlaşılabilir. Karmaşık kodların fazlalığı bir çok açıdan dezavantaj sağlar. Bu karmaşa, arama motoru optimizasyon sorununa, gekesiz yere büyük hat genişliği kullanımına ve performans sorunlarına mâl olur.

2.8.3. Arama Motoru Optimizasyonu

Bazı web uygulamaları gerçekten inanılmaz görünseler ve işlevleri de bir okadar başarılı olsa da, ziyaretçi sayılarına bakıldığında vasatın çok daha altında kaldıkları gözlemlenmiştir [6]. Bir miktar ziyaretçileri olsa da robot ziyaretçi sayıları neredeyse yoktur. Bu tür sitelere en büyük örnek tamamen flash animasyon olarak geliştirilmiş sitelerdir.

Bir Flash uygulamasında, eşsiz grafikler ve çok iyi bir yönetim paneli geliştirilebilir. Flash siteleri çok zarif tasarlanmış ve kullanımları da çok pratik olsalar da arama motorları tarafından içerikleri tanımlanamaz ve indekslenemezler.

Bir çok yazılımcı, eski arama motoru kurallarını kullanmaya devam ediyor. Arama motorları, önceden olduğu gibi yalnızca meta-tag, tanımlama bilgisi ve anahtar kelime bilgilerine göre indekslemiyorlar. Bu tanımlamalar günümüz arama motorları açısından yalnızca herhangi bir sayfada bulunan basit bir paragraflık yazıdan farklı bir anlam taşımazlar [7].

```
<meta name=”description” content=”Bu web sitesi SEO
hakkındadır” />
<meta name=”keywords” content=”SEO, Arama Motoru
Optimizasyonu” />
```

Dikkat edilmesi gereken başlıca konulardan birtanesi url, title, h1 ve h2 etiketlerine girilen değerlerin ilgili içeriklerle ilgili net ve okunaklı bilgi içermesidir.

Her sayfa yalnızca bir tane H1 ve en fazla üç tane H2 etiketi içermelidir. Arama motorları bu HTML etiketlerini sayfanın içeriğiyle ilgili ilişki kurmak için değerlendirirler.

Bir diğer sıklıkla kullanılan strateji ise arama motoru robotları için site haritası dosyası oluşturmaktır. Site haritaları [14], web yöneticilerinin arama motorlarını sitelerindeki taranabilir sayfalar hakkında bilgilendirmeleri için kolay bir yoldur. En basit biçimiyle, bir site haritası, arama motorlarının siteyi daha akıllıca tarayabilmeleri için, her bir URL'yle ilgili ek verilerle (son güncellenme zamanı, genellikle ne sıklıkta değiştiği ve sitedeki diğer URL'lere göre ne kadar önemli olduğu) birlikte, bir siteye ilişkin URL'leri listeleyen bir XML dosyasıdır.

Web tarayıcıları, sayfaları çoğunlukla site içerisindeki veya başka sitelerdeki bağlantılardan keşfeder. Site haritaları, site haritasını destekleyen tarayıcıların site haritası listesinde bulunan tüm URL'leri toplamasına ve ilişkili meta verilerini kullanarak bu URL'ler hakkında bilgi edinmesine olanak tanır.

Yukarıda anlatılan metodlar özenle uygulansa bile optimizasyonla ilgili başarı sağlanamayabilir. Web sitesinin arama motorları tarafından kolayca taranabilecek yapıda olması gerekmektedir. Eğer site içerisinde gereksiz kodlar veya içeriği kirleten veriler mevcutsa arama motorları içerik hakkında sağlıklı veri toplayamaz ve siteyi indeksleyemezler. Asp.NET web formlarda bulunan sessionState bilgileri veya sunucu kontroller için eklenen java Script kodları örnek olarak gösterilebilir.

Teknodestek web uygulaması için, html içeriğinin temiz olması ve arama motoru optimizasyonunun başarılı olabilmesi için ayrı bir özen gösterilmiş ve çaba sarfedilmiştir.

3. SİSTEM MODELLERİ

3.1.Genel Bakış

Geliştirilen proje, biribiriyle ilişkili dört ana modülden oluşmaktadır. Bu dört modül Üyelik yönetimi, Forum(Soru/Cevap) uygulaması, Anket yönetimi ve Makale yönetiminden meydana gelmektedir. Genel yapı için VisualStudio.NET 2010 platformu tercih edilmiştir. Asp.Net MVC mimarisi kullanılmıştır.

Uygulama web sitesi şeklinde sunulacak ve hizmet verecektir. Bu nedenle internet tarayıcı programı ile görüntülenecek ve kullanılacaktır. Uygulama içerisindeki roller ve işlevler kısaca aşağıda belirtilmiştir;

1) Kayıtsız Kullanıcıların
    a. Sistemde bulunan makaleleri takip edebildiği
    b. Sorulmuş soruları ve cevaplarını takip edebildiği
    c. Duyuruları görebildiği
    d. Anketleri görebildiği

2) Kayıtlı Tüm Kullanıcıların (Kayıtsız kullanıcı yeteneklerine ek olarak)
    a. Makale ekleyebildiği
    b. Makale takip edebildiği
    c. Soru sorabildiği
    d. Cevap verebildiği
    e. Sorulmuş sorulara ve cevaplara oy verebildiği
    f. Yorumlar ekleyebildiği

3) Editörlerin (Kayıtlı ve kayıtsız kullanıcılara ek olarak)
    a. Eklenen makaleleri yayına sunabildiği
    b. Gerek duyduğunda makale, soru veya cevapları silebildiği
    c. Duyuru ekleyebildiği
    d. Anket ekleyebildiği
    e. Diğer tüm kullanıcılarla irtibata geçebildiği

4) Yöneticilerin
    a. Tüm kullanıcı yetkilerini ve rollerini belirleyebildiği

3.2.Fonksiyonel Gerekler

Fonksiyonel gerekler;

Genel fonksiyon gerekleri

- Pratik ve fonksiyonel kullanıcı web arayüzünün oluşturulması. (Renkleri ve işlevleriyle kullanımı ve görüntüsü profesyonel olmalı.)
- Üye kaydı, üye girişi, kullanıcı yetkileri, kullanıcı rolleri ve yönetimi formlarının oluşturulması.

Forum Bölümü

- **Forum veri yönetimi**
    - Forum içeriği için kategori yönetimi formunun oluşturulması
    - Yeni soru kaydı ekranının oluşturulması
    - Sorulmuş sorulara cevap kaydı ekranının oluşturulması
    - Soruları puanlama ekranının oluşturulması
    - Cevapları puanlama ekranının oluşturulması

- **Forum veri izleme/okuma**
    - Site içi arama ekranının oluşturulması
    - Site içerisindeki tüm bilgilerin(soru/cevap/makale) üye olan veya olmayan kullanıcıların izleyebileceği ekranın oluşturulması

Makale Yönetimi

- **Makale veri yönetimi**
    - Yeni makale kaydı ekranının oluşturulması
    - Makalelere yorum kaydı ekranının oluşturulması

- Makaleleri puanlama ekranının oluşturulması

- **Makale izleme/okuma**
  - Site içi arama ekranının oluşturulması
  - Site içerisindeki tüm bilgilerin(soru/cevap/makale) üye olan veya olmayan kullanıcıların izleyebileceği ekranın oluşturulması

- **İstatistik Yönetimi**

  - Forumların, okunma izlenme ve oylanma sayılarının tutulması.
  - Makalelerin, okunma izlenme ve oylanma sayılarının tutulması.

3.3.Fonksiyonel Olmayan Gerekler

3.3.1. Kullanılabilirlik

Uygulama içerisindeki verilerden, internet bağlantısı ve web tarayıcısı olan herkes faydalanabilir. Veri okumak için herhangi bir üyelik gerekmemektedir. Yalnızca içeriğe katkı sağlamak isteyen kullanıcılar için ek olarak üyelik kaydı talebi yapılacaktır.

3.3.2. Güvenlik

Web server internet çıkışına yapılandırılmış Firewall, dışarıdan erişime tamamen kapatılmış SQL Server ve SSL sertifika sayesinde tüm uygulamanın güvenilirliği sağlanmıştır.

3.3.3. Performans

Sistemi barındırmak için Blade sunucu kullanılacak olması ve uygulama içerisinde profil nesneleri kullanılması sayesinde tek web sunucu üzerinden 8 farklı işlemci çekirdeğiyle işletilebilecektir.

3.3.4. Donanım Gerekleri

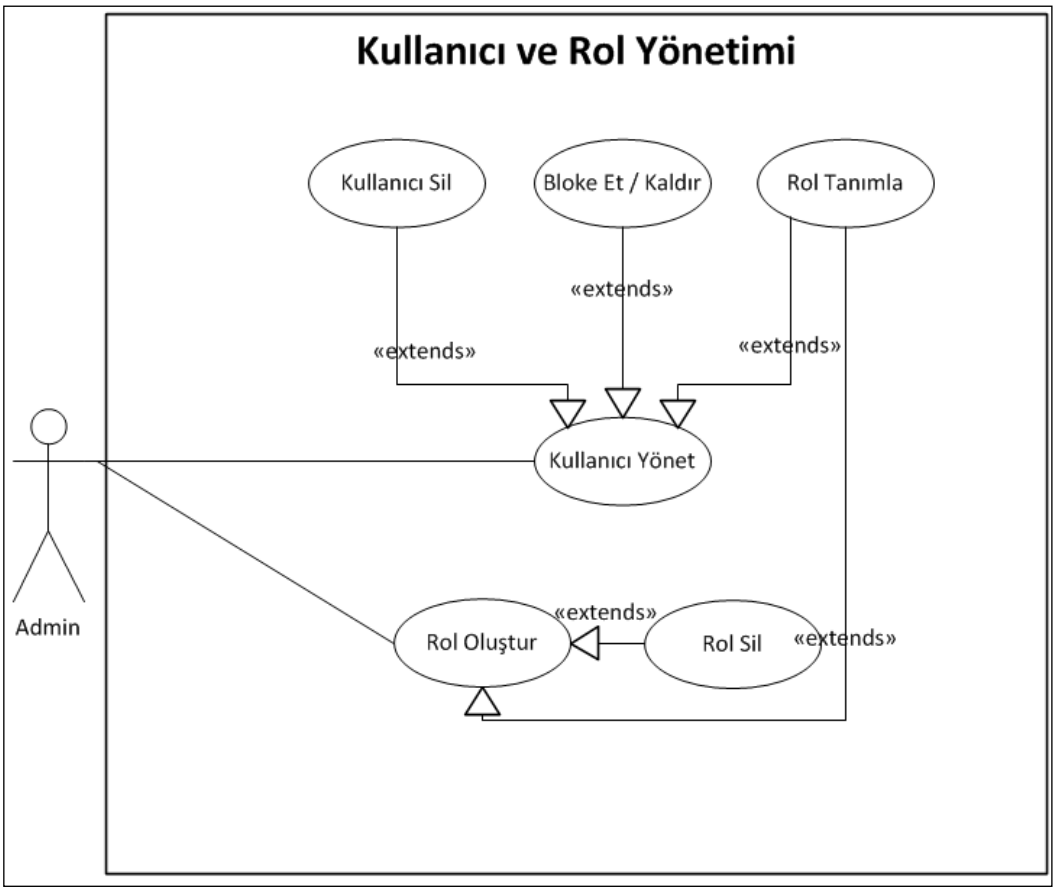
Uygulama için yalnızca Windows 2008 Server üzerine inşa edilmiş IIS 7.0 yeterlidir.

3.3.5. Sınırlamalar

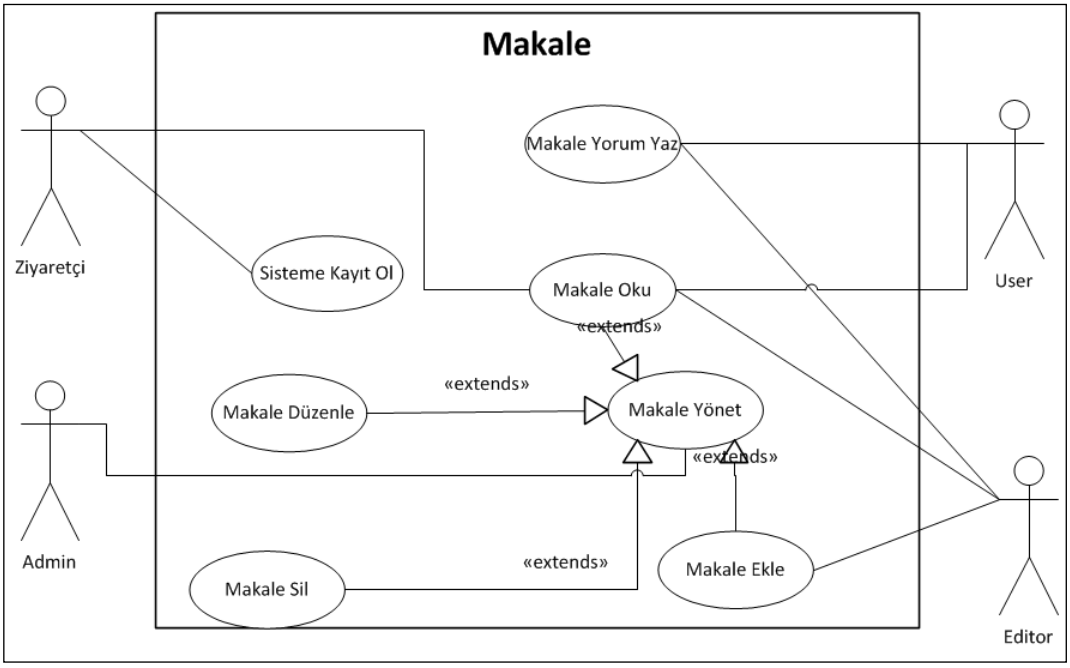
Hedef platform, Microsoft Web Server üzerine inşa edilecek ve uygulama geliştirme dili olarak Microsoft Visual Studio .NET MVC - C# kullanılacaktır.

3.4.Kullanıcı Durumu (Use Case) Modeli

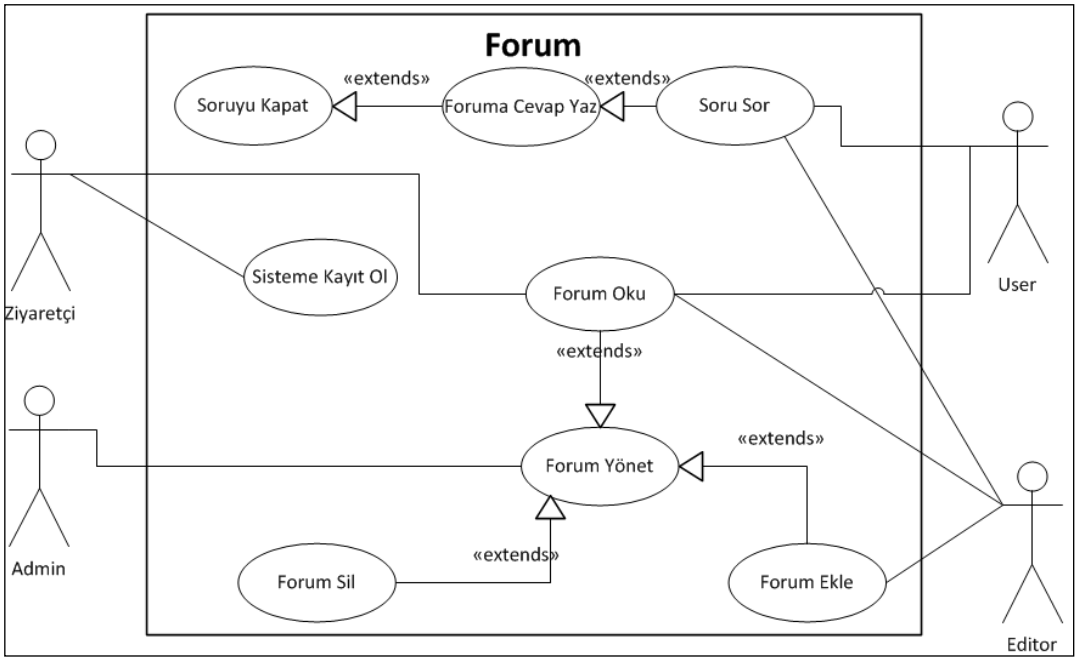
Bu bölümde sırasıyla, Kullanıcı ve Rol Yönetimi, Makale, Forum ve Anket modüllerinin kullanıcı durumu diyagramları bulunmaktadır [11].



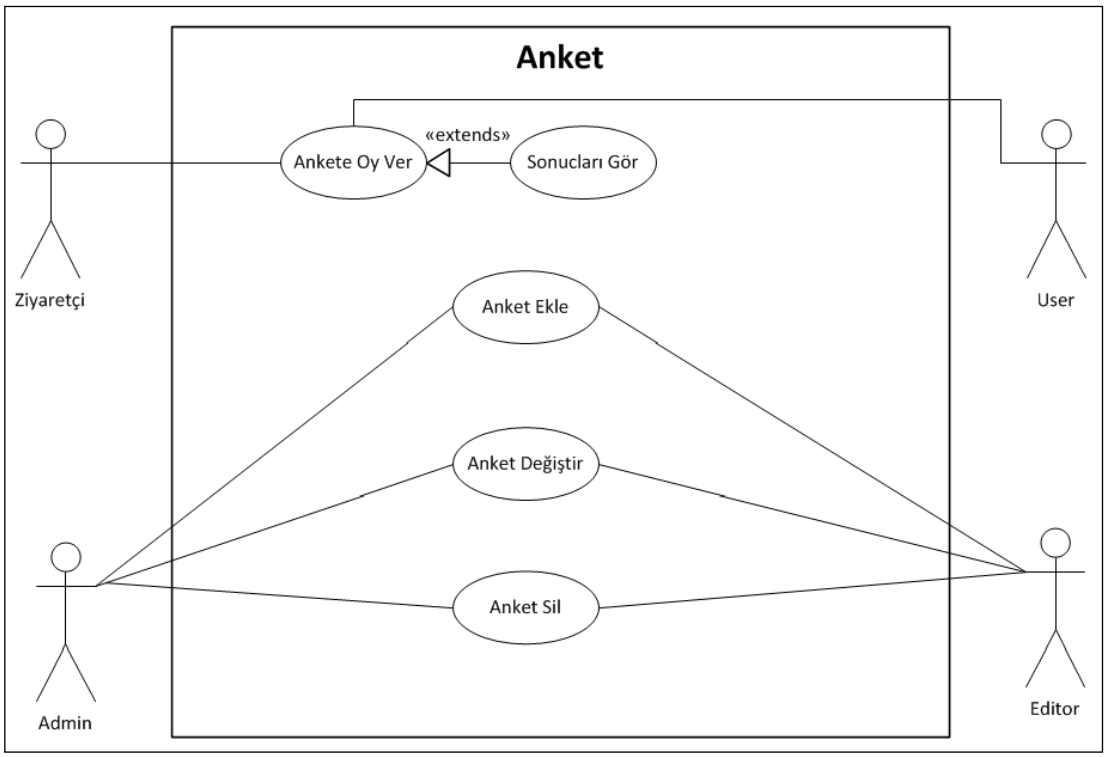
Şekil 3-1 Kullanıcı yönetimi ve rol yönetimi kullanıcı durumu modeli



Şekil 3-2 Makale kullanıcı durumu modeli



Şekil 3-3 Forum kullanıcı durumu modeli



Şekil 3-4 Anket kullanıcı durumu modeli

3.5.Kullanıcı Arayüzü

Sisteme ilk girişte, Şekil 3-5'de gösterilmiş olan anasayfa ekranı açılmaktadır. Ana sayfa ekranında kullanıcı girişi bağlantı butonu, menüler(makaleler, bilgilendirme, anketler, forum), arama bölümü bağlantı butonu görülmektedir. Sisteme ilk girişte misafir kullanıcının, içeriklerde arama yapmak ve okumak dışında, soru sormak, sorulara cevap vermek, oylama işlemleri gibi temel fonksiyonları yerine getirebilmek için sisteme kayıtlı kullanıcı olarak giriş yapması gerekmektedir. Şekil 3-6'da sisteme kullanıcı girişi için "Giriş" bağlantısı da görülmektedir.

Kullanıcıların, uygulamanın amacını ve içeriğini ilk bakışta anlamalarını sağlamak için, sisteme ilk girişte, sitede bulunan makaleler listelenmiştir.

İlerleyen arayüz çizimlerinde uygulamanın sırasıyla ekran çıktıları ve işlevleri anlatılmıştır.

Şekil 3-5 Sisteme ilk giriş, ana sayfa ekran görünümü

Şekil 3-6 Kullanıcı girişi ekran görünümü

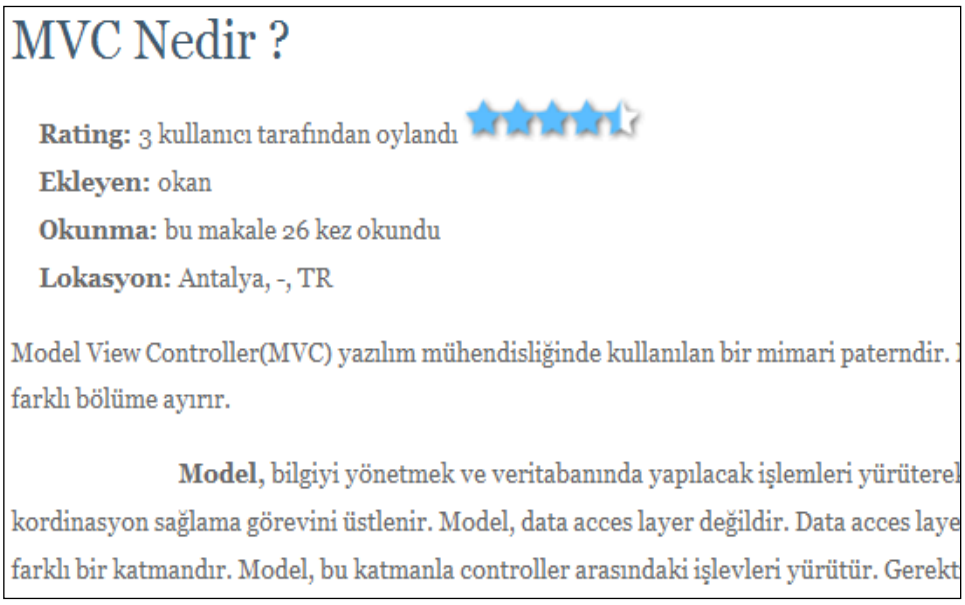
Şekil 3-7'de sisteme yeni kayıt ekranı görülmektedir. Sisteme yeni kayıt işlemini mümkün olduğunca kısa tutmak amacıyla yalnızca kullanıcı giriş bilgileriyle ilgili gerekli alanlar zorunludur. Kullanıcı, kayıt işlemini tamamladıktan sonra sisteme giriş yaparak, dilediği taktirde profil bilgileri menüsünü kullanarak profil bilgilerini güncelleyebilmektedir. Profil bilgileri, isteğe bağlı olarak güncellenmektedir.

Şekil 3-7 Sisteme yeni kayıt ekran görünümü

Şekil 3-8'de arama ekranı görülmektedir. Bu arama ekranı tam metin arama, otomatik tamamlama ve Fonetik algoritmalar kullanılarak genişletilmiştir.

Şekil 3-8 Makale arama ekranı

Şekil 3-9'da makale görüntüleme sayfasının ekran çıktısı verilmiştir. Bu sayfa aracılığıyla kullanıcılar makaleleri okuyabilir, yorum yapabilir veya oylayabilirler. Böylece istatistiki olarak en çok oy alan makaleler görüntülenebilmektedirler.

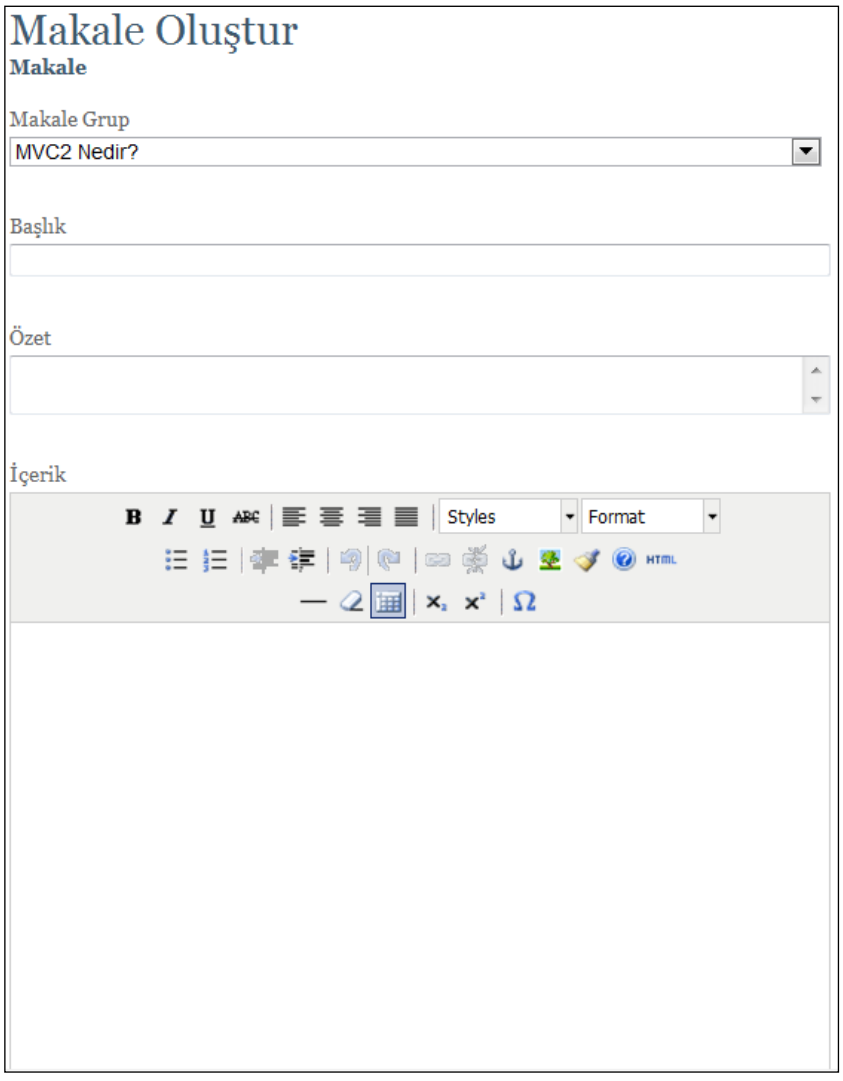


Şekil 3-9 Makale görüntüleme ekran görünümü

Şekil 3-10 Makale yorum ekleme ekran görünümü

Şekil 3-11'de makale yönetimi menüsü gösterilmektedir. Bu menü aracılığıyla Admin yetkisi olan kullanıcılar makale grubu ekleyebilir, yeni makale yazabilir, silebilir, değiştirebilir veya makale yorumlarını silebilirler. Editor seviyesinde yetkisi olan kullanıcılar ise yalnızca makale ekleyip silebilmektedirler.

Şekil 3-11 Admin – makale yönetim ekranı

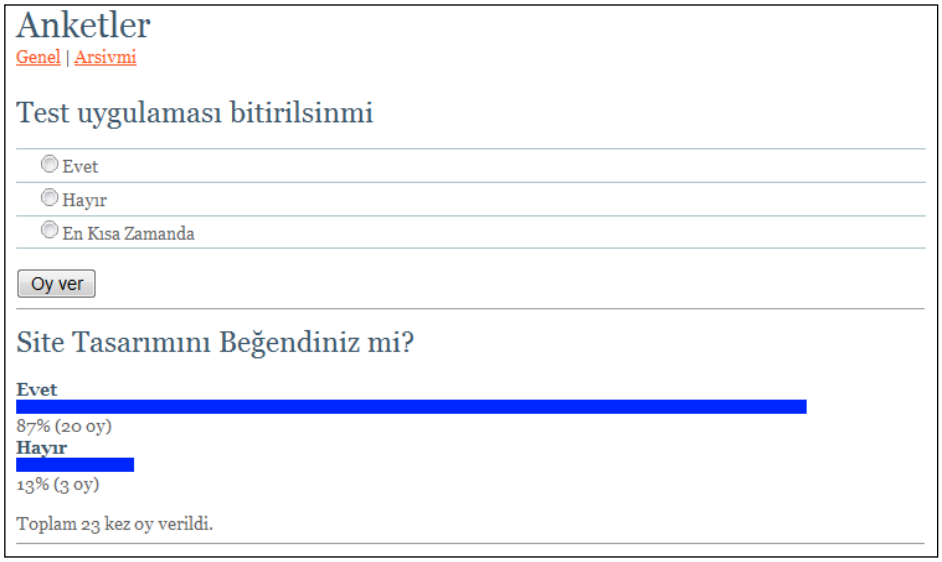


Şekil 3-12 Makale oluşturma ekranı

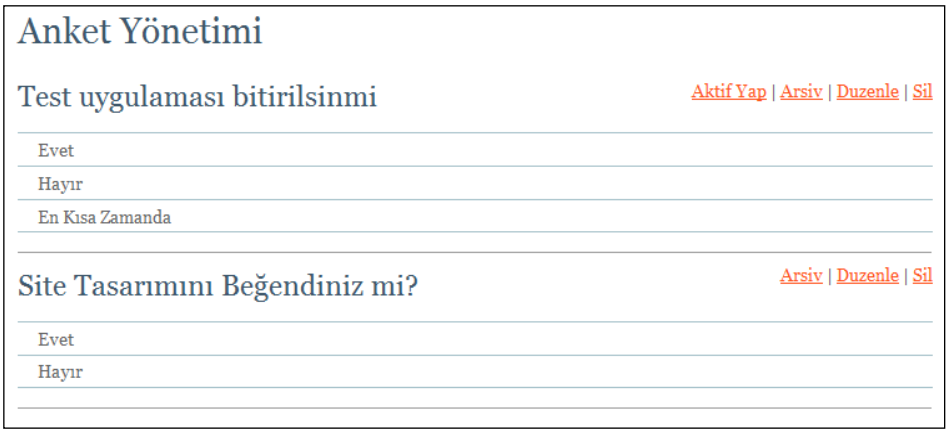
Şekil 3-13'de Anket görüntüleme ve oy verme ekranı verilmiştir. Bu bölüme, Anketler bağlantısına tıklanarak ulaşılabilmektedir. Arşivdeki anketler ve güncel

anketler isteğe göre görüntülenebilir. Kullanıcı bir kez oy verdikten sonra, ilgili anket için o ana kadar verilen oyların toplam oranı görüntülenmektedir. Bu sayede oy veren kullanıcı anlık olarak anket sonuçlarını görüntüleyebilmektedir.

Anket yönetimi işlemleri, yönetici panelinden yapılmaktadır. Yönetici paneli aracılığıyla yeni anket eklenebilir, mevcut anketler silinebilir, yine aynı şekilde anket seçenekleri silinebilir veya eklenebilir.



Şekil 3-13 Anket görüntüleme ve oy verme ekranı



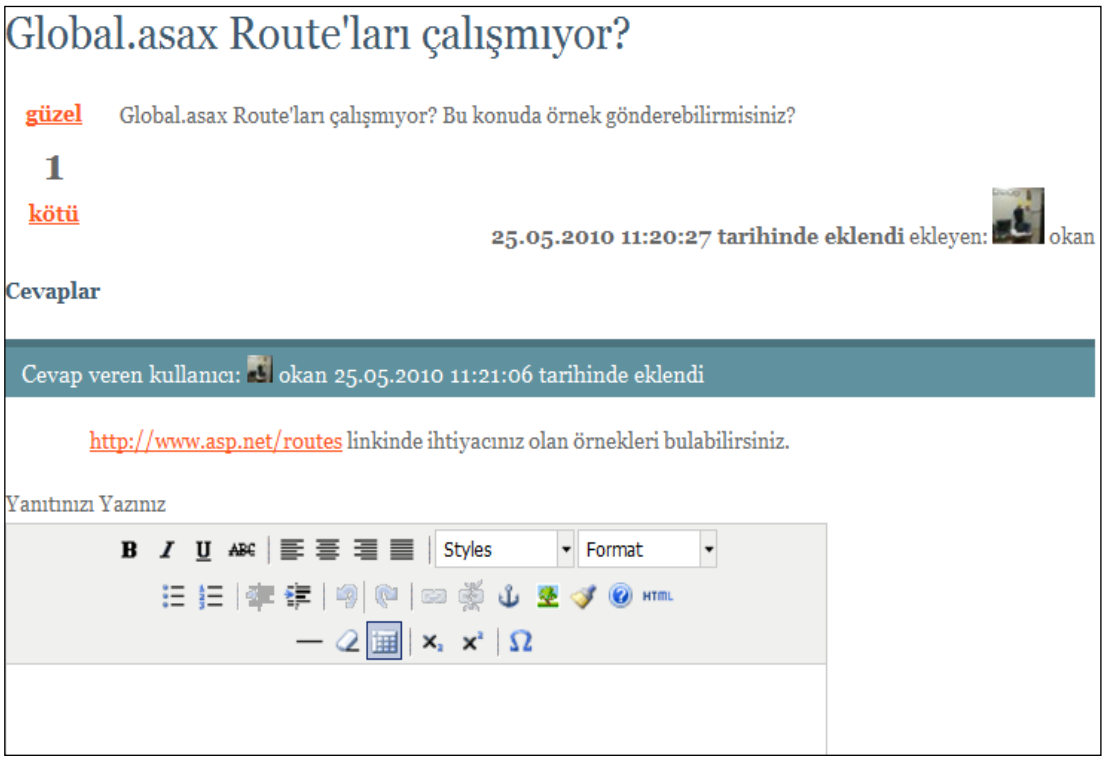
Şekil 3-14 Anket yönetimi ekranı

Şekil 3-15'de Mevcut forumların görüntülendiği ekran çıktısı verilmiştir. Bu sayfadan, soruların listesi, kaç kez oy verildiği, kaç kez cevaplandığı veya okunduğu gibi istatistik bilgileri de görüntülenmektedir. Yönetici yetkisi olan kullanıcılar,

mevcut forumları silebilirler veya “Kapalı” bağlantısına tıklayarak ilgili foruma daha fazla cevap verilmesini engelleyebilirler.

Şekil 3-15 Forum görüntüleme ekranı

Herhangi bir forum konusunun başlığında bulunan bağlantıya tıklandığında, Şekil 3-16’da ekran çıktısı verilen forum içeriği görüntülenir. Forum içeriğinde, sorulan soru ve altında verilen cevaplar bir liste halinde görüntülenmektedir. Kullanıcılar, bu menü yardımıyla sorulan soru ve tüm cevaplarını takip edebilirler. Sisteme kayıtlı olan kullanıcılar, “güzel” veya “kötü” ismindeki butonlara tıklayarak ilgili soruyu değerlendirebilirler. Bu değerlendirmeler en beğenilen soruların listesini görüntülemek için kullanılabildiği gibi ekranı ilk kez açan kullanıcıların da soru hakkında ön fikir sahibi olmalarını sağlayacaktır. Ayrıca aynı ekran içerisinden sorunun sorulduğu tarihi ve her cevap için ayrı ayrı cevaplanma tarihlerini görmek mümkündür.



Şekil 3-16 Cevap verme ekranı

Son olarak Şekil 3-17'de genel yönetici ekranının çıktısı verilmiştir. Bu ekran aracılığıyla Admin yetkisi olan kullanıcı, menüde listelenmiş olan tüm modülleri yönetebilmektedir. Yönetici paneli yardımıyla yetkisi olan kullanıcılar, sınırsız olarak kullanıcı rolü tanımlayabilirler ve bu rolleri ilgili kullanıcılara paylaştırabilirler. Rol yönetimi sayesinde yönetici menüsü sadece Admin seviyesinde olan kullanıcılar için görüntülenmektedir. Yetkisi olmayan kullanıcılar, yalnızca giriş seviyesindeki menüleri(içerik görüntüleme) ve bağlantıları görebilirler.

Bu menü yardımıyla yapılabilecek diğer işlemler aşağıda kısaca liste halinde verilmiştir;

- Kullanıcı yetkilendirme, kullanıcı silme işlemleri
- Rol oluşturma, değiştirme ve silme işlemleri
- Makale ve makale yorumlarını silme, ekleme ve düzenleme işlemleri
- Bilgilendirme ve haberleri, silme ve ekleme işlemleri
- Yeni anket ekleme, ve silme işlemleri
- Forum kategorisi oluşturma, sorulan soruları ve cevaplarını silme ve değiştirme işlemleri

Şekil 3-17 Admin ekranı

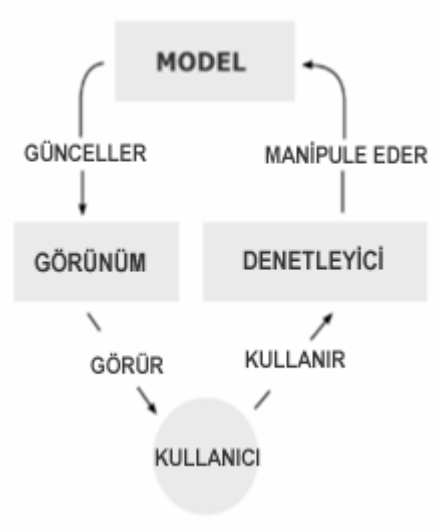
4. MVC

Model – Görünüm - Denetleyici, (Model View Controller, MVC) yazılım mühendisliğinde kullanılan bir mimari kalıptır (pattern). MVC kalıbı, uygulamayı üç farklı bölüme ayırır [2].

- **Model,** veri nesnelerini yönetir ve veritabanında yapılacak işlemleri yürüterek denetleyici bölümüyle koordinasyonu sağlar. Model, veriye erişim katmanı (VEK) değildir. VEK, modelin altında bulunan farklı bir katmandır. Model, bu katmanla denetleyici arasındaki işlevleri yürütür. Gerektiğinde denetleyiciden gelen istekleri veritabanına uygular veya veritabanındaki verileri işleyerek denetleyiciden gelen talebe göre iletir.

- **Görünüm,** modeli yorumlar ve kullanıcıyla etkileşim sağlamak amacıyla bir forma iletir. Bu form, kullanıcıdan gelen istekler doğrultusunda denetleyiciye istek gönderir veya yine bir istek doğrultusunda denetleyici aracılığyla veri çağırır.

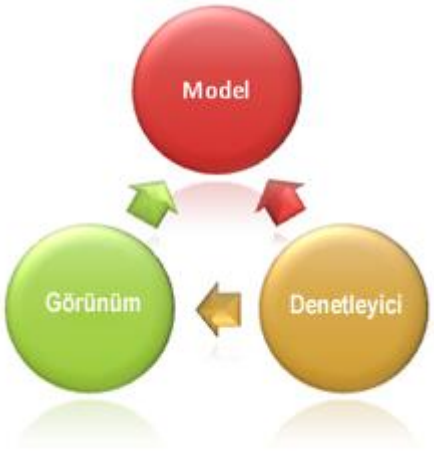
- **Denetleyici** ise görünüm katmanından gelen girdileri(input) değerlendirir ve modelle iletişimi sağlar. Denetleyici, kullanıcılar tarafından görünüm nesnesi aracılığıyla girilen veriler doğrultusunda modele talimatlar gönderir ve bu girdiler doğrultusunda aksiyonların (Action) uygulanmasını sağlar.



Şekil 4-1 Model, Görünüm, Denetleyici (Model-View-Controller) işlemleri

MVC tasarım kalıbı, genellikle web tabanlı uygulamalar geliştirmek amacıyla kullanılır. Görünüm kısmı HTML veya XHTML formlardan oluşur. Denetleyici genellikle GET, POST metodları aracılığıyla kullanıcılardan gelen istekleri yorumlar ve bu isteklerle ne yapılacağına karar vererek model katmanındaki kodları işletir. Örneğin kullanıcının yeni üye kaydı talebini alarak, veritabanında yeni kullanıcı oluşturulması için talebi modele iletir. Oluşturulan yeni kaydı modelden isteyerek, bilgileri teyit etmesi için kullanıcıya geri gönderir.

Bu tez çalışmasında Visual Studio.NET MVC tasarım kalıbı kullanılmıştır. ASP.NET MVC, ASP.NET Web uygulama iskelet sisteminin bir parçasıdır. Bu sistem iki farklı programlama modelini içerir. Birtanesi, ASP.NET web uygulamaları, diğeri ise ASP.NET Web formlarıdır. MVC uygulaması, aşağıdaki şemada gösterildiği gibi 3 farklı özellik kullanılarak tasarlanır ve uygulanır.



Şekil 4-2 Model, Görünüm, Denetleyici (Model-View-Controller) mimarisi

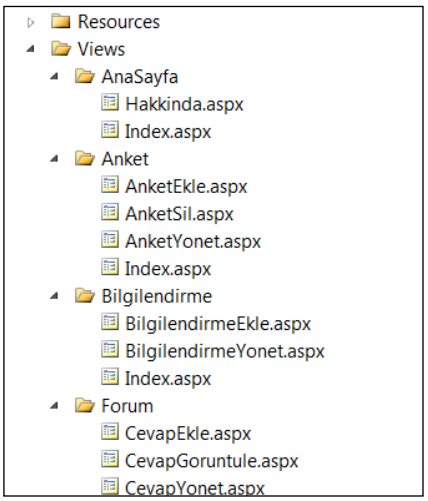
4.1.Model

Model, veritabanı işlemlerinin yapılabilmesi için gerekli olan tablo, saklı yordam ve fonksiyon gibi işlevleri nesne olarak tanımlayıp daha sonrasında denetleyici katmanında gerekli işlemleri yaparak kullanılabilmesine olanak tanıyan katmandır [2]. Model, bir uygulama için veriyle ilgili çekirdek bilgileri içermektedir. Veri doğrulama kuralları ve ayrıca veriye erişim ile veriyi bir araya toplama mantığını barındırır. Model, aşağıdaki paketleri içerir.

1. Etki odaklı tasarım (Domain Driven Design, DDD)
2. Test odaklı geliştirme (TOG)
3. Depo kalıp (Repository Pattern)
4. Servis kalıp (Service Pattern)
5. Şartname kalıp (Specification Pattern)
6. LINQ-To-SQL
7. ADO.NET Entity Framework
8. NHiberante
9. Veri Tabloları
10. Özel iş katmanı mantık kodları

4.2.Görünüm (View)

Görünüm, uygulamanın kullanıcıya sunulduğu arayüzleri içerir. Görünüm katmanı içerisinde, Asp.NET HTML formları kullanılmaktadır. ASP.NET MVC içerisinde varsayılan olarak gelen motor WebForm görünüm motorudur. Bu motor, klasör hiyerarşisinin oluşturulması görevini ve aspx ile ascx dosyalarının tarayıcıya gönderilirken HTML‘e dönüştürülme görevini üstlenir. WebForm görünüm motoru, Framework 1.0‘dan bu güne, standart görünüm motoru olarak kullanılmaktadır. Ancak bu motor, geliştirilerek yalnızca kontrol tabanlı(control-based) dönüştürme yapabiliyorken, dahili kod taraflı(inline-codebased) dönüştürme yetenekleri de eklenmiştir. Şekil 4-3’de varsayılan görünüm motoru için oluşturulmuş klasör hiyerarşisi görülmektedir. İlerleyen bölümlerde, aşağıdaki ekran çıktısında gösterilen klasörlerdeki dosyaların, denetleyici katmanıyla nasıl eşleştirildiği ve denetleyici tarafından nasıl tarandığı da anlatılacaktır.



Şekil 4-3 Klasör hiyerarşisi

Görünüm motoru, aspx ve ascx dosyalarını çevirmek için neredeyse aynı metodları uygular. Bu nedenle aspx ve ascx dosyaları aynı metodla doğrudan HTML'ye çevrilebilirler. Fakat denetleyici tarafından çevrilecek dosyaların aspx'mi yoksa ascx'mi olduğunu anlayabilmesi için bazı gereksinimler vardır. Standart ASP.NET MVC görünüm motoru hangi dosyayı çevireceğine karar verebilmek için aşağıda gösterilen şekilde bir listeyi en alttan başlayarak en üste kadar tarar.

- ~/Views/{controller}/{action}.aspx
- ~/Views/{controller}/{action}.ascx
- ~/Views/Shared/{action}.aspx
- ~/Views/Shared/{action}.ascx

Bu listenin anlamı kısaca şöyle özetlenebilir;

- Denetleyici öncelikle normal klasörleri tek tek kontrol eder. (Shared klasörleri sona bırakır)
- Aspx ve sayfa (page) dosyaları ascx dosyalarından önce kontrol edilir
- Gözatma listesi, view sayfalarında seçilmiş olan ana şablon (master template) dosyalarını'da işler. Fakat ana şablonların çevrilme işlemleri View sayfası ve denetleyiciye göre biraz daha farklıdır. Aşağıda ana sayfaların (master page) kontrol edilme sırası görülmektedir.

- ~/Views/{controller}/{master_name}.master

- ~/Views/Shared/{master_name}.master

4.2.1. ViewMasterPage, ViewPage ve ViewUserControl

ASP.NET MVC de kullanılmakta olan ve yukarıdaki başlıkta bildirilen üç nesne, Asp.NET web form metodlarından da bilinmektedir [2].

- ViewMasterPage ve MasterPage, Page nesnesine şablon tanımlamak için kullanılan nesnelerdir.
- ViewPage görüntülenmesi istenen ana içeriği barındıran nesnedir.
- ViewUserControl, UserControl, modüler tasarım, alt içerikler, diğer sayfalar tarafından ortak kullanılabilen nesneleri barındırır.

MVC'deki bu nesneler aslında web form karşıtlarından miras edilmişler(inheritance) yani türetilmişlerdir, çünkü onların derleyici tarafından inşa edilmesi anında çalıştırılmasına dayanırlar ki bu ASP.NET çekirdeği tarafından, Internet Informatin Server (IIS) gibi sunucularda içerik(content) iletmek için kullanılan yollardan biridir.

Bu nesnelerin tümünde görmeye alışık olduğumuz arayüzler (User, Context, Request, Response, IsPostBack v.b.) hala MVC'nin Page, User Control ve Masterpage nesnelerinde mevcuttur.

4.2.2. ViewData ve Model

ViewData özelliği, veriyi saklamak, model ve denetleyiciden görünüm nesnesine aktarmak için kullanılır. Bu özellik, sözlük(Dictionary) nesnesi olarak da kullanılabilir;

```
<%= ViewData["text"] %>
```

Veya bir model nesnesi olarak da kullanılabilir;

```
<%= ViewData.Model.MusteriID %>
```

Bu şekilde kullanım ancak genelleyici(generic) metodlar kullanılarak herhangi bir nesneden türetildiğinde (inheritance) uygulunabilir;

Örneğin, Musteri tipindeki ViewPage kullanımı şu şekilde örneklenebilir;

```
public partial class MusteriDuzenle : ViewPage<Musteri>
```

veya,

```
<%@ Page Inherits="System.Web.Mvc.ViewUser<Musteri>" %>
```

Bu çok yönlü bir kolleksiyondur ve hem görünümde hemde denetleyicide kullanılabilir. Her nesne, bu ControllerContext'den türetilerek ViewContext'e gönderilebilir.

4.2.3. TempData

TempData özelliği, ViewData özelliğine benzese de her türlü veriyi ek istekler için oturumda taşıyabilen bir özelliktir [17]. TempData nesnesi ilk bakışta önemsiz gibi görünse de bu özellik sayesinde, uygulama içindeki veriler ASP.NET web formda bulunan ViewState kullanımı gibi farklı istekler için saklanabilir. Bir başka dikkat çekici özelliği ise yönlendirme işlemleri için kullanılabilmesidir.
Şöyle bir senaryo örnek olarak gösterilebilir;

Sisteme kullanıcı girişi yapmamış bir kullanıcının ziyaretçi olarak giriş yaptığı varsayılsın. Sistem bu kullanıcıyı, kullanıcı girişi yapmasını sağlamak için giriş sayfasına yönlendirsin ve sistem, kullanıcıya içeriği görmeden önce sisteme giriş yapması gerektiği uyarısını göstersin fakat bu mesaj yalnızca kullanıcı doğrudan giriş sayfasına gitmediği durumlarda geçerli olsun.

Öncelikle bu tip bir senaryoyu gerçekleştirebilmek için; kullanıcının giriş sayfasına farklı bir sayfadan yönlendirilip yönlendirilmediğini tespit etmek gerekiyor, bu tespit işlemi için bir yönlendirici sayfa adında değişken bir değerin varlığını kontrol etmek çözüm olabilir fakat bu durum kontrol edilse bile kullanıcı sisteme giriş yaptıktan sonra hangi sayfaya yönlendirileceğini bulmak birhayli zordur. TempData işte bu durum için hazır bulunmaktadır. Aşağıdaki kod satırları bu durum için ihtiyaç duyulan mesajı taşıyabilir;

```
<% if (TempData[“Mesaj”] != null) { %>
<div class=”mesaj”><%= TempData[“Mesaj”] %></div>
<% } %>
```

Bu kod ayrıca ana şablon dosyasına yazılabilir ve kullanıcıya tüm sayfalar için yalnızca üç satır kodla aynı mesaj iletilebilir.

4.2.4. HTML ve AJAX Metodları

HTML ve Ajax metodları Link ve Textbox gibi nesneler oluşturmak için kullanılırlar. ASP.NET MVC’de HTML yapısı tamamen değişmiştir [17]. Örneğin, MusteriAdi adında bir TextBox oluşturmak için aşağıdaki kodu yazmak yeterlidir.

```
<%= Html.TextBox(“MusteriAdi”) %>
```

Bu kod, aşağıdaki html kodunu üretecektir;

```
<input type=”text” name=”MusteriAdi” id=”MusteriAdi” value=”“ />
```

Bu nesneye denetleyiciden veri göndermek için şu kodu yazmak yeterlidir;

```
ViewData[“MusteriAdi”]
```

HTML formlarında mevcut olan tüm girdilerin ASP.NET MVC içinde karşılıkları bulunmaktadır. Ek olarak Ajax uyulaması gibi bazı eklentiler de MVC içinde mevcuttur.

Bu özellikleri kullanmak için tek yapılması gereken MVC iskelet sistemi içinde web sayfası oluşturmaktır. Yazılımcı kendi metodlarını oluşturmak veya mevcut metodları geliştirmek isterse aşağıdaki satırı yazması yeterlidir;

```
public static string MyCustomControl (this HtmlHelper
html, string name)
```

This kelimesi, eklenecek olan custom metodun, HtmlHelper'ına ekleneceğini ifade eder. HtmlHelper, viewPage'lerde Html olarak temsil edilir.

4.3.Denetleyici (Controller)

Denetleyici, kontrol akış mantığını içermektedir [1]. Model ve görünüm katmanı ile koordinasyonu ve verinin bu iki katman arasındaki akışını sağlayarak, uygulamanın yürütülme görevini üstlenir. Uygulama mantığı açısından, model katmanını manipüle eder, kullanıcı etkileşimlerini yönetir ve tarayıcıda gösterilecek olan görünüm nesnesini seçer ve görevlendirir. System.Web.Mvc.Icontroller arayüzünden (interface) türetilen bir sınıftır.

Bu 3 farklı katman(MVC), uygulamanın inşası esnasında çeviklik, hız, esneklik ve kolay bakım(debug) özellikleri sağlar. Örneğin, görünüm katmanını ayırarak çekirdek İMK hiç bir değişiklik yapılmasına gerek kalmadan uygulamanın kullanıcı arayüzünde değişiklik yapılabilmesini mümkün kılar. Ayrıca bu ayrım, uygulama geliştiren yazılımcıların farklı rollerde çalışabilmesine de imkân tanır. Örneğin bir yazılımcı, görünüm katmanı kısmında çalışırken diğeri, denetleyici katmanını geliştirebilir.

ASP.NET MVC, klasik web form metoduna ek olarak aşağıdaki yetenekleri sağlamaktadır;

- HTML üzerinde tam kontrol ve hakimiyet sağlar
- Zengin AJAX ve Jquery entegrasyonu içerir
- Uygulamaya, Arama motoru uyumlu (SEO-friendly) URL yetenekleri kazandırır
- TOG özelliği içerir.

4.3.1. URL Rotaları (URL Routes)

Denetleyicinin önemli bir parçası da URL'lerin tanımlandığı rotalardır(Routes). Rotalar, denetleyici listesine, hangi denetleyicinin tetikleneceğini ve denetleyiciler içindeki hangi aksiyonun çalıştırılacağını söylerler. Varsayılan rotaya bir göz atılacak olursa;

```
routes.MapRoute(
 "Default",
```

```
"{controller}/{action}/{id}",
new { controller = "Home", action = "Index", id = "" }
);
```

Bu rota tanımında, URL aşağıdaki gibi oluşturulabilir;

```
{controller}/{action}/{id}
```

Aşağıda; Rota tanımının URL'leri nasıl ayırdığını gösteren tablo verilmiştir;

Tablo 4-1 URL rota metodlarının aldığı parametreler

URL	Denetleyici	Aksiyon	ID
/	Home	Index	
/Home	Home	Index	
/Home/About	Home	About	
/Account	Account	Index	
/Account/User/1	Account	User	1

Tablo 4-1'de görüldüğü gibi bazı bölümlerde URL'lerin tanımlanmadığı, hatta ilk satırda hiç bir parametrenin tanımlanmadığı görülmektedir. Bu durum, URL'nin her bölümü için bir varsayılan değer tanımlaması yapıldığı içindir. Eğer değerlerden herhangi birisi eksik girilirse varsayılan rota değerleri işletilir. Varsayılan rota, önceki örnekte gösterildiği gibi, şu kodla tanımlanır;

```
new { controller = "Home", action = "Index", id = "" }
```

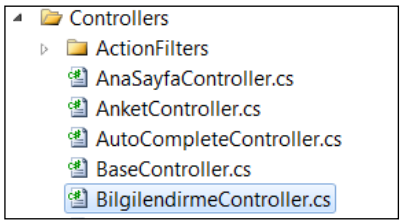
Asp.NET MVC'de, her rota tanımı için en az 2 bölüm değeri gereklidir. Bu değerler "Controller" yani denetleyici ve "Action" yani aksiyondur. Çünkü bu değerler denetleyici listesinin doğru denetleyici nesnesini ve denetleyici içindeki doğru aksiyonu bulabilmesi için kullanılırlar. Diğer tüm bölümler isteğe bağlı olarak aksiyon metodlar için adreslenebilir veya kullanılabilirler.

4.3.2. Denetleyici Fabrikası (Controller Factory)

ASP.NET MVC iskelet sisteminde "DefaultControllerFactory" adındaki denetleyici fabrikası kullanılır. Aşağıda bildirilen kriterler varsayılan olarak mevcut denetleyicilerin aranılmasında kullanılırlar;

- Web Assembly'nin isim uzayı ile Controller sona eklenirler. Teknodestek.Web.Controllers
- İsim uzayının içindeki nesnenin biçimi şu şekilde olmalıdır {controller_adi}Controller(that is, AnaController, HesapController)
- Nesneler Icontroller arayüzeyinden türetilmelidir.

Şekil 4-4'de denetleyicilerin ekran çıktısı verilmiştir.



Şekil 4-4 Denetleyici klasörü

Tüm bu kriterler yeni bir denetleyici eklemenin zor olacakmış gibi algılanmasına neden olabilir ama aslında tek yapılması gereken Controller klasörüne yeni bir kod dosyası eklemektir ve kod dosyasının denetleyici nesnesinin System.Web.Mvcnamespace'den türetildiğine emin olmaktır.

4.3.3. Aksiyonlar (Actions)

Aksiyonlar önceki konularda bahsedildiği gibi, görüntülenecek olan görünüm dosyasının URL'sine neyin bağlanacağını belirlemek için kullanılırlar. Anlaşılması oldukça kolaydır. Aşağıdaki kodda, genel yapıyı anlayabilmek için aksiyon metodda kullanılabilen farklı kısımların bir araya getirilmiş hali gösterilmiştir. Diğer kısımların (Section) görevi anlatılacaktır.

```
[AcceptVerbs(HttpVerbs.Post)]
[OutputCache(Duration = 600)]
```

```
public ActionResult MusteriGetir (string adi, string
email)
{
// MusteriGetir aksiyonunu çalıştırmak için yazılan kod
return View()
}
```

4.3.4. Metodlar (Methods)

Bir aksiyon metodunun [17] diğer standart .NET metodlarından hiç bir farkı yoktur. Aksiyon metodlar da parametre alırlar ve bir değer döndürürler. Tek farkı denetleyici sınıfının içinde olmalarıdır. Yukarıda verilmiş olan örnek kod incelendiğinde metod yapısının, diğer standart .NET metodlarıyla aynı olduğu görülmektedir.

4.3.5. Sonuçlar (Results)

Sonucun diğer adı, standart metoddaki "Return" koduyla dönen değerdir. Burdaki sonucun tek farkı ise ActionResult [2] tipinde türetilmesi yani miras edilmesidir. Önceki kodda sonuç tipi ActionResult'tır fakat aslında dönen nesne ViewResult diye adlandırılan ActionResult'tan türetilmiştir. ResultView, denetleyicideki View() diye adlandırılan bir korunmuş metoddan (Protected Method) türetilir.

4.3.6. Filtreler (Filters)

Filtreler [2], aksiyon metodlara nitelik kazandırmak amacıyla kullanılırlar. Biri aksiyonlar diğeri de sonuçlar için kullanılan toplam iki adet filtre çeşidi mevcuttur. Aksiyon filtreler, aksiyon metodlarla ilgilidir ve 2 tane olayları vardır. Bir tanesi aksiyon, çalışmadan hemen önce, diğeri de aksiyon çalıştıktan sonra tetiklenir. Sonuç fitreler ise HTTP yanıtlara aittir ve aynı şekilde 2 olayları vardır. Biri, yanıt tarayıcıya gönderilmeden önce, diğeri de tarayıcıya gönderildikten sonra çalışır. Bu tezde filtrelerden daha fazla bahsedilmeyecektir fakat uygulama içinde yetki(Authorization), önbellekleme(Caching) ve RESTFUL servis sonuçlar için filtreler kullanılmıştır. Yukarıda verilen örnek kodda; Çıkış önbellek (OutputCache) için filtre kullanılmıştır.

4.3.7. Seçiciler (Selectors)

Seçiciler [2] aksiyon metoda nitelik veya nitelikler belirlemek için kullanılırlar. Hem filtreler hem de seçiciler nitelik eklerler. Sıklıkla bu ikisi arasındaki farkı anlamak zor gibi görünse de aralarında çok önemli bir fark vardır. Seçiciler denetleyici fabrikasının, hangi aksiyon metodunun çağrılacağını belirlerken çağrılırlar. Yani aksiyon metodun çalışma aşamasıyla herhangi bir ilgileri yoktur. Yukarıda verilen örnek kod, AcceptVerbs niteliğini belirlemek için kullanılmıştır.

4.4.MVC'mi, Klasik Web Form mu ?

Microsoft forumlarında bu günlerde, Asp.net web form ve Asp.net MVC hakkında bir tartışma konusu mevcut [15]. Bir çok insan MVC'nin klasik web formların yerini alacağını düşünüyor. Diğerleri de MVC'nin klasik web formlarının yerini almayacağını idda ediyor.

Bu gerçekten olacak mı bilinmez ama MVC nin geliştirilme amacı, web formlarını kaldırmak yerine alternatif bir uygulama geliştirme metodu inşa etmektir. Bu durumda iki yaklaşımın da devam edeceğini söylemek yanlış olmaz. Farklı metodlar diğer platformlar için de geçerlidir. Özellikle Java platformu bu duruma örnek gösterilebilir. Bu araştırmada işlenen konulardan birtanesi de, yıllardır yalnızca web form metodunu destekleyen Microsoft, neden MVC kalıbını geliştirme gereği duyduğudur. Bu iki yaklaşım için Visual Studio.NET platformunun avantajları ve dezavantajları nelerdir?

Visual Studio .NET web form un problemleri şu şekilde sıralanabilir,

Microsoft, Windows form model tasarımının aynısını web ortamında da gerçekleştirilebilmesini mümkün kılan bir uygulama geliştirme platformu tasarlamaya çalıştı. Bu nedenle .NET platformu bir çok Windows form geliştiricisinin ilgisini çekmeyi başardı. Özellikle Visual Basic 6.0 (VB 6.0) geliştiricileri bu kitlenin çoğunluğunu oluşturuyorlardı. Öyle ki, VB 6.0 da uygulama geliştiren bir çok programcı, hiç bir HTTP veya WEB bilgisine bile gerek duymadan mevcut uygulamalarını ASP.NET ortamına taşıyabildiler. Windows form geliştirme metodunu taklit edebilmek için web form yapısına olay-yöneticisi(event-driven) yaklaşımı, ViewState ve Postback özellikleri eklendi. Bu metodlar yazılım geliştiricilere tanıtıldı. ASP.NET'in web form metodolijisi, doğal web teknolojisinde bazı kırılımlar ortaya çıkardı. Özellikle ViewState ve Postback, web ortamında geliştirilen uygulamalar için bir çok problemi ve karmaşayı da beraberinde getirdi.

Bir çok web sayfasının yüzlerce kilobyte'lık ViewState verisi içermesi, web sayfalarının ve uygulamalarının performansını önemli ölçüde etkiliyordu. Uygulama geliştiriciler, sunucu tarafından üretilen HTML formları ve standart etiketler (tag) içermeyen sunucu kontrolleri üzerinde yeteri kadar kontrol sahibi olamıyorlardı. Web formlarla ilgili bir diğer önemli sorunlardan birisi de javaScript entegrasyonuydu. HTML formlarındaki nesnelerin isimlerinin dönüştürülme işlemi esnasında değişmesi, JavaScript kodları tarafından kullanılmasında önemli sorunlar oluşturuyordu. Web form yaşam döngüsünde geliştirilmiş bir sayfa hem çok karmaşıktı, hem veri girişi hemde verinin sunulması görevi için tek bir sınıf kullanılıyordu. Birim testi (Unit testing) işlemi ise neredeyse imkansız bir işti. Oysa günümüzde, birim testi işlemi, modern bir uygulama teknolojisi için son derece önemlidir. Özellikle "Agile Software Development" metodolojileri açısından bakıldığında vazgeçilmezdir demek yanlış olmayacaktır.

Web teknolojisi herhangi bir ülkeye ait olmadığı için olaylar, PostBack ve ViewState iyi bir yöntem değildir. Bu nedenler yüzünden çoğu ASP.NET Web form geliştiricisi, Web tarayıcı uyumsuzluğu gibi bir çok farklı problemle yüzleşmek durumunda kalıyor.

Örneğin, objectDataSource nesnesi kullanıldığında Web formda bulunan hiç bir Grid nesnesi kültür (culture) özelliği kullanamaz. Varsayılan olarak tarih biçimini "Eng" standartında tarih biçimi olarak kabul eder ve bu değer değiştirilemez.

4.5.ASP.NET MVC Yöntemi

ASP.NET MVC, ASP.NET platformunun içerdiği hiç bir güç ve esneklikten ödün vermeden Web Formlarının içerdiği mevcut karmaşanın ve problemlerin yok edilmesini veya en aza indirilmesini amaçlamıştır [15]. MVC tasarım kalıbında Web form olayları yerine, Denetleyici Aksiyon (Controller Action) getirilmiştir. MVC kalıbının başlıca avantajlarından birtanesi, her katmanın kendi alanlarına göre ayrılmış olmasıdır. Birim testi kolaylığı eklenmiştir ve URL, HTML üzerinde çok daha fazla kontrol olanağı eklenmiştir. MVC kalıbı, Viewstate, Postback, sunucu kontrolleri ve sunucu taraflı formlar kullanmaz. MVC model, dosya adı uzantılı dosyalar (web formlarda kullanılan) yerine Temsili Konum Transferi (TKT) tabanlı URL'ler kullanır. Bu yapı, arama motorları tarafından indekslenebilir bir yapı sağlar. Arama motoru uyumluluğu (Search engine optimization, SEO) yeteneği denen bu

yöntem web uygulamaları için vazgeçilmez bir yetenektir. Mevcut MVC yapısı, .NET Framework 3.5 tarafından desteklenmektedir.

4.5.1. MVC Tasarım Kalıbının Avantajları

1. İlgili katmanlar, kendi alanlarına göre tamamen ayrılır .
2. Yorumlanmış HTML üzerinde tam kontrol sağlar.
3. TOG mümkün kılar
4. SEO ve TKT uyumlu URL sağlar
5. JavaScript iskelet sistemiyle tam entegrasyon sağlar.
6. NVelocity, Brail, Nhaml gibi üçüncü parti görünüm motorlarını destekler
7. ViewState ve PostBack olay kullanmaz.
8. "Stateless Nature of Web" kurallarına uyar
9. Genişletilebilir ve açık kodlu bir sistemdir.
10. Web 2.0 uygulamaları için ideal platformdur.

4.5.2. Web Form Model Avantajları

1. Hızlı uygulama geliştirme sağlar.
2. Şirket hedefinde yapılacak olan geliştirmeler (Line of Business) için kolay geliştirme olanakları sağlar.
3. Zengin kontrol nesneleri sunar.
4. Windows form geliştirme metodolojisine çok yakın bir metodddur.

4.5.3. En İyi Yaklaşım Hangisi

Bu seçim her yazılımcıya göre farklılık gösterebilir [15]. Eğer HTML üzerinde daha fazla hakimiyet, test odaklı geliştirme isteniyorsa, web standartlarına bağlılık, erişilebilirlik veya SEO tabanlı URL ler isteniyorsa, ozaman MVC kalıp daha doğru bir seçim olacaktır. Eğer çok daha zengin kontrol galerisi ve durum tabanlı olay-yönetimli web geliştirme aracı isteniyorsa ozaman Web Form model seçilebilir.

5. TAM METİN ARAMA

Bu bölümde tam metin arama metodları [9] ve Fonetik algoritmalar [10] hakkında inceleme yapıldıktan sonra Sql Server tam metin arama [5] metodlarına yer verilecek ve uygulamalı olarak anlatılacaktır.

5.1.Tam Metin Arama Tanımı

Son yıllarda kişisel bilgisayar kullanımı giderek artmakta ve her geçen gün artmaya da devam etmektedir. E-posta gibi elektronik dökümanların kullanımı da aynı oranda artmaktadır [9]. Netice olarak bir çok insan bir çok elektronik dökümanı farklı amaçlar için araştırmakta ve toplamaktadırlar. Günümüz teknolojisi gereği dünya genelinde dökümanlar artık bilgisayar ortamında yazılmakta ve saklanmaktadır. Bu teknolojiye dayalı bilgi ortamı, devasa çoğunlukta birikmiş olan dökümanların verimli olarak yönetilmesi konusunda çalışma yapmayı gerekli kılmıştır. Tam metin arama(Full Text Search), elektronik ortamda yazılmış sayısız dökümanlar içerisinde, mümkün olan en kısa sürede bilgi aramak için geliştirilmiş bir arama tekniğidir. Günümüzde tam metin arama, özellikle internet ortamında çok farklı servislerce kullanılmakadır. Her geçen gün artan döküman sayısı ve bu dökümanları kullanan insanların artması, bu arama tekniğinin daha hızlı ve daha verimli olma gerekliliğini de tetiklemektedir.

5.1.1. Çevrilmiş Dosya Kullanılarak Geliştirilmiş Statik Tam Metin Arama

Dünya üzerinde en çok kullanılan metod çevrilmiş dosya kullanılarak oluşturulan statik tam metin arama metodu olarak bilinmektedir [9]. Statik tam metin arama metodunun indeksleme tekniği bu bölümde anlatılacaktır. Elektronik ortamda bulunan bir dizi döküman olduğu varsayılsın. Her döküman, kendi içerisinde bulunan anahtar kelimeler ve niteliklerine göre oluşturulmuş bir liste tarafından tayin edilmektedir.

5.1.2. İndeksleme Algoritması

İndeksleme algoritması şu şekildedir;

Adım 1: Karakterin, metin içerisindeki konumu belirlenir.

Tablo 5-1 Karakterler ve döküman içerisindeki konumları

Karakter	Döküman İçindeki Yeri
v	1
e	2
r	3
i	4
y	5
i	6
ç	7
e	8
v	9
i	10
r	11
g	12
e	13
r	14
i	15
g	16
e	17
t	18
i	19
r	20

Tablo 5-2 Birleştirilmiş karakterler ve döküman içerisindeki konum bilgileri.

Karakter	Döküman İçindeki Yeri
v	1,9
e	2,8,13,17
r	3,11,14,20
i	4,6,10,15,19
y	5
ç	7
g	12,16
t	18

Tablo 5-3 Karakterlere ve döküman içerisindeki eş konum bilgilerine göre sıralanmış hali

Karakter	Döküman İçindeki Yeri
ç	7
e	2,8,13,17
g	12,16
i	4,6,10,15,19
r	3,11,14,20
t	18
v	1,9
y	5

Adım 2: Aynı karakterler birleştirilir ve döküman içerisindeki konumları yan yana yazılır.

Adım 3: Karakterlerin yerlerini bildiren liste indeks dosyası olarak kaydedilir. Genellikle tam metin arama, geniş ölçekli dökümanlarda arama yapmak için kullanıldığından oluşan indeks dosyası da büyük olacaktır. Bu nedenle, verilerin karakter konum bilgisi data grubunu içeren liste istenildiğinde hızlıca karşılaştırma işlemi yapılabilmesi için artan tipinde sıralanır.

Örnek 1: "veriyi çevir geri getir" adında bir döküman olduğu varsayılsın. 5-1 ve 5-3 Tabloları adım 1, adım 2 ve adım 3 için daha açıklayıcı anlaşılmasını sağlayacaktır.

5.1.3. Bulup Getirme Algoritması

Tablo 5-4 Çevrilmiş dosya içerisindeki konum bilgisi

İndeks	Karakter	Döküman İçindeki Yeri
1	g	12,16
2	e	2,8,13,17
3	r	3,11,14,20
4	i	4,6,10,15,19

Tablo 5-5 Çevrilmiş dosya içerisindeki konum bilgisi

İndeks	Karakter	Döküman İçindeki Yeri
1	g	12,16
2	e	2,8,13,17
3	r	3,11,14,20
4	i	4,6,10,15,19

Tablo5- 5. Girilen anahtar değerin sırasıyla döküman içerisindeki konum bilgisi

Çevrilmiş dosya içindeki bulup getirme algoritması aşağıda gösterilmiştir;

Adım 1: Girilen anahtar değer, karakterlere ayrılır. Geri kelimesi g, e, r, i olarak ayrılmıştır.

Adım 2: Karakterlerin çevrilmiş dosya içerisindeki yerleri Tablo 5-4 'de gösterildiği gibi bulunur.

Adım 3: Girilen anahtar değerin her karakteri, çevrilmiş dosya içerisindeki çokluklarına göre Tablo 5-5'deki gibi sıralanır ve tekrar indekslenir.

Adım 4: Anahtar kelimenin karakter sırasına dayalı olarak pozisyonların sırasını getirir. Tablo 5-6'da gösterilmiştir.

Tablo 5-6 Anahtar kelimenin karakterlerinin sıralı listesi

İndeks	Karakter	Döküman İçindeki Yeri
1	g	12,16
2	e	2,8,13,17
3	r	3,11,14,20
4	i	4,6,10,15,19

Adım 5: "geri" olarak girilmiş olan anahtar kelimesinin " veriyi çevir geri getir" çevrilmiş dosyasının içerisindeki pozisyonu, g karakterinden itibaren hesaplanır. Sonuç olarak "geri" anahtar değerinin pozisyonu 12 'den, pozisyon 15'e kadardır.

5.2.Tam Metin Arama Uygulama Örneği

Geliştirilen uygulamada SQL Server 2008 tam metin arama metodları kullanılmıştır [5]. SQL Server, tam metin arama işlemleri için çevrilmiş indeks dosyası kullanmaktadır. Bu metod özetle veritabanı içerisindeki aranabilir kelimeleri ayırarak konum bilgileriyle birlikte farklı bir dosyada indeksler.

Tablo 5-7 Örnek bir cümlenin çevrilmiş indeksi

Kelime	Döküman ID	Pozisyonu
Now	1	1
is	1	2
the	1	3
time	1	4
for	1	5
all	1	6
good	1	7
men	1	8
.	.	.
.	.	.
.	.	.

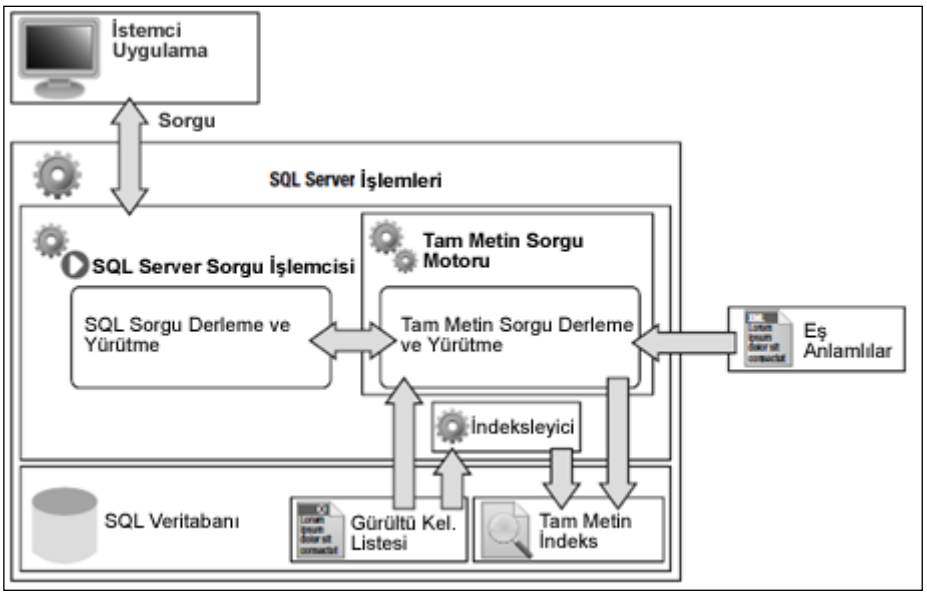
Dosyanın içerisinde, indekslenmiş kelimelere göre 2 tane alan bulunmaktadır. Document ID alanında kelimenin rastlandığı dökümanın ID numarası, Pozisyonu alanında ise kelimenin döküman içerisindeki pozisyonu bulunmaktadır. SQL Server, indeksleme işlemi esnasında öncelikle "stopwords" olarak tanımlanmış olan kelimeleri elemektedir ve indeks dosyasına dahil etmemektedir. Örneğin ingilizce içerikli dökümanlar için the, and, of gibi karakterler elenir. Bu liste düzenlenebilir bir

listedir ve her dile uyarlanabilir durumdadır. İstenirse projeye özel liste oluşturulabilmektedir. Aşağıdaki tabloda aynı cümlenin stopwords filtreleme işlemi uygulanmış hali bulunmaktadır.

Tablo 5-8 Örnek cümle için gürültü kelimeler silindikten sonraki indeks

Kelime	Döküman ID	Pozisyonu
time	1	4
good	1	7
men	1	8
aid	1	13
party	1	16
{End Of File}	1	17

Şekil 5-1'de Sql Server tam metin arama mimarisi gösterilmiştir.



Şekil 5-1 Tam metin arama mimarisi

Şekil 5-1 de gösterilen bileşenler şu şekildedirler;

- **İstemci uygulama :** Tam metin arama sorgularının oluşumundan ve bu sorguların, SQL Server sorgu işlemcisine gönderilmesinden sorumludur. Sorgu dizilimi kontrolü ve doğrulaması bu etapta yapılmaktadır.

- **SQL Server işlemleri:** Bu modül, SQL sorgularını derleyen ve çalıştıran sorgu işlemcisini ve tam metin sorgularını çalıştıran tam metin motorunu içermektedir.

- **SQL Server sorgu işlemcisi:** SQL Server sorgu işlemcisi, SQL sorgularının doğrulamasını yapan, sorguları derleyen, sorgu planlarını oluşturan elemanları barındıran işlemcilerden oluşur.

- **Tam metin sorgu motoru:** Herhangi bir SQL Server tam metin arama talebi geldiğinde bu talep doğrudan tam metin sorgu motoruna iletilir. Bu motor, gelen sorgu isteğini inceler doğrulamasını yapar, sorguyu gerçekleştirmek için tam metin indeks dosyasına danışır ve aldığı cevaba göre istemciye sonucu döndürür.

- **İndeksleyici:** Bu modül, dökümanları inceleyerek indeks dosyasını oluşturur.

- **Tam metin indeks:** Veritabanında bulunan dökümanların, indeksleyici modülü tarafından oluşturulmuş indeks dosyalarını içermektedir. Tam metin motoru, bu listeye gelen istekler doğrultusunda danışır.

- **Gürültü Kelime Listesi:** Sorgularda aranmasına gerek olmayan kelimeleri içeren dosyaları barındırır. Bu dosyalarda bulunan kelimeler, gürültü olarak kabul edilir ve indeksleyici tarafından değerlendirilerek indeksleme işlemi esnasında göz ardı edilirler.

- **Eş anlamlılar:** Bu bölüm eş anlamlı kelimeler sözlüğünü içeren XML dosyalarını barındırır. Sorgu isteği esnasında, kullanıcıların aradığı kelimelerle bu sözlük eşleştirilir ve eşleşen kelime bulunması durumunda bu kelimeler de sorgulanırlar. Örneğin, kullanıcı “kırılmak” kelimesini arattığında bu listeye tanımlanması durumunda kırılmak kelimesiyle eş anlamlı olan, “parçalanmak” ve “parçalara ayrılmak” olarak da aynı anda sorgulanması sağlanabilir.

5.2.1. SQL Server Tam Metin Arama Metodları

SQL Server 2008, temel olarak CONTAINS ve FREETEXT olmak üzere iki türlü arama fonksiyonu içermektedir [5].

- CONTAINS arama türü cümle arama, yakınlık arama, tam arama ve ileri düzey arama yeteneklerini içermektedir.
- FREETEXT ise, bulanık arama metodları ve temel arama metodlarına sahiptir.

FREETEXT ve FREETEXTTABLE

Arama metodları içeren uygulamalar geliştiren veritabanı yöneticilerinin yaptıkları hataların başında, yetersiz arama metodları geliştirmeleri gelmektedir. Arama sonuçlarında alakasız verilen gelme olasılığının yüksek olması da bu durumla bağlantılıdır. Bunun en büyük nedenlerinden birisi içeriğin doğru olarak indekslenmemesi veya sorguların başarısız ve itinasız oluşturulmasından kaynaklanmaktadır.

Genellikle sorgular çok katı eşleşme yapmaya çalıştığı için kullanıcıların arama sonuçlarının boş dönme olasılığı çok yüksektir. Örneğin araba sözcüğü aranırken, araç, otomobil, arabam gibi kelimeler de eşzamanlı olarak sistemde aranmalıdırlar. Hatta ingilizce aramalarda fiillerin zamanlara göre, isimlerin çoğul veya tekil olma durumlarına göre arama yapılabilmelidir.

FREETEXT sorgularının bir diğer özelliği de, eş anlamlılar sözlüğünü otomatik olarak uygulaması ve sorgu içeriğini genişletmesidir. Kelimeleri tekil veya çoğul, cinsiyet farklılığı, durum, fiil formlarına göre arayabilmektedir. Örneğin, inligizce aramalar için jog kelimesi aşağıdaki formlarıyla birlikte aranır;

- Jog
- Jogging
- Jogged

Yine ingilizce aramalar için, book kelimesi aşağıdaki formlarıyla araştırılır;
Hem isim hemde fiil halleri dahil edilecektir.

- Books
- Booked

- Booking
- Book

FREETEXT sorgularına ayrıca eş anlamlılar sözlüğü uygulanarak özel olarak sorguların genişletilmesi sağlanır. Yukarıdaki örneklerden devam edecek olunursa Jog kelimesi için ayrıca “run” ve “sprint” kelimelerinin de aranması sağlanabilir.

5.2.2. Tam Metin Arama Uygulaması İçin SQL Kodu

Bu tez kapsamında geliştirilen web uygulamasında makale içeriği arama fonksiyonu için vurgulu tam metin arama metodu kullanılmıştır. Aşağıda saklı yordam kodları gösterilmektedir.

FREETEXT saklı yordam kodu;

```
ALTER PROCEDURE [dbo].[sp_MakaleAra]
  @AranacakText nvarchar(100),
  @Style nvarchar(200)
AS
BEGIN
  CREATE TABLE #uyusan_dokumanlar
  (
    dok_id bigint NOT NULL PRIMARY KEY
  );

  INSERT INTO #uyusan_dokumanlar
  (
    dok_id
  )
  SELECT DISTINCT
    makaleID
  FROM  TeknoDestek.Makaleler
  WHERE FREETEXT
  (
    (Icerik,Baslik),
   @AranacakText ,
    LANGUAGE N'Turkish'
  );

  DECLARE @db_id int = DB_ID(),
```

```
    @table_id int = OBJECT_ID(N'teknodestek.Makaleler'),
    @column_id int =
    (
      SELECT
        column_id
      FROM sys.columns
      WHERE object_id = OBJECT_ID(N'teknodestek.Makaleler')
        AND name = N'Icerik'
    );

  SELECT
    s.makaleID,
    c.Baslik,
    MIN
    (
      N'...' + SUBSTRING
      (
        REPLACE
          (
            c.Icerik,
            s.Display_Term,
            N'<span  style="'  +  @Style  +  '">'  +  s.Display_Term  +
'</span>'
          ),
        s.Pos - 512,
        s.Length + 1024
      ) + N'...'
    ) AS Icerik
  FROM
    (
      SELECT DISTINCT
        c.makaleID,
        w.Display_Term,
        PATINDEX
          (
            N'%[^a-z]' + w.Display_Term + N'[^a-z]%',
            c.Icerik
          ) AS Pos,
        LEN(w.Display_Term) AS Length
      FROM sys.dm_fts_index_keywords_by_document
        (
```

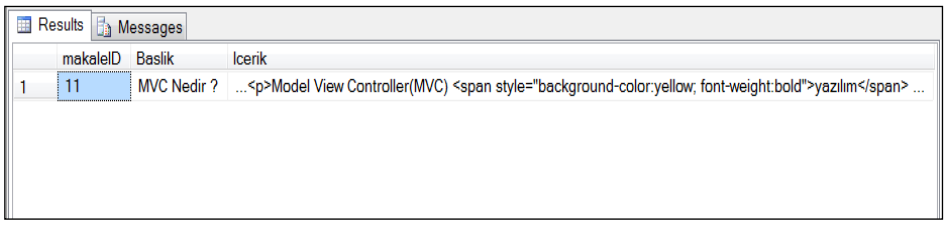
```
            @db_id,
            @table_id
          ) w
        INNER JOIN TeknoDestek.Makaleler c
          ON w.document_id = c.makaleID
        WHERE w.column_id = @column_id
          AND EXISTS
            (
              SELECT 1
              FROM #uyusan_dokumanlar m
              WHERE m.dok_id = w.document_id
            )
          AND EXISTS
            (
              SELECT 1
              FROM sys.dm_fts_parser
                (
                  N'FORMSOF(FREETEXT, "' + @AranacakText + N'")',
                  1055,
                  0,
                  1
                ) p
              WHERE p.Display_Term = w.Display_Term
            )
    ) s
    INNER JOIN TeknoDestek.Makaleler c
    ON s.MakaleID = c.MakaleID
  GROUP BY
    s.MakaleID,c.Baslik

  DROP TABLE #uyusan_dokumanlar;

END;
```

Bu prosedür, iki adet parametre almaktadır. Birtanesi aranacak değer diğeri ise bulunan kelimenin belirtileceği HTML stilidir. Aşağıdaki kodda "yazılım" kelimesi birinci parametre olarak gönderilmiş hali verilmiştir.

```
EXECUTE sp_MakaleAra N'yazılım',
N'background-color:yellow; font-weight:bold';
```



Şekil 5-2 sp_MakaleAra sorgusunun sonucu

Yukarıda, FREETEXT saklı yordam kodu olarak gösterilmiş olan kod, makaleler tablosunun baslik ve icerik alanlarına tam metin arama metodunu uygulamaktadır. Eşleşen makalelerin sonuçları ID bilgileri ile geçici bir tabloda listelenmektedir.

```
CREATE TABLE #uyusan_dokumanlar
  (
    dok_id bigint NOT NULL PRIMARY KEY
  );

  INSERT INTO #uyusan_dokumanlar
  (
    dok_id
  )
  SELECT DISTINCT
    makaleID
  FROM  TeknoDestek.Makaleler
  WHERE FREETEXT
  (
    (Icerik,Baslik),
   @AranacakText ,
    LANGUAGE N'Turkish'
     );
```

Prosedür daha sonra yürürlükte olan veritabanının ID bilgisini, makaleler tablosunun ID bilgisini ve içerik kolonunun ID bilgisini getirir. Tüm bu bilgiler daha sonra "dm_fts_index_keywords_by_document" fonksiyonu tarafından kullanılacaktır.

```
  DECLARE @db_id int = DB_ID(),
    @table_id int = OBJECT_ID(N'teknodestek.Makaleler'),
    @column_id int =
    (
      SELECT
```

```
      column_id
    FROM sys.columns
    WHERE object_id = OBJECT_ID(N'teknodestek.Makaleler')
      AND name = N'Icerik'
  );
```

Sorgunun son parçası ise uyuşan kayıtları ve içerikten uyuşan kısmın 1KB lık parçasını getirmektedir. SELECT cümlesi uyuşan kayıtların başlık bilgisini, satır ID numarasını ve vurgulu olarak içerik bilgisinin 1KB'lık uyuşan bölümünü getirir. Uyuşan kelime > HTML etiketi arasında belirtilmiştir.

```
  SELECT
    s.makaleID,
    c.Baslik,
    MIN
    (
      N'...' + SUBSTRING
      (
        REPLACE
          (
            c.Icerik,
            s.Display_Term,
            N'<span style="' + @Style + '">' + s.Display_Term +
'</span>'
          ),
        s.Pos - 512,
        s.Length + 1024
      ) + N'...'
    ) AS Icerik
```

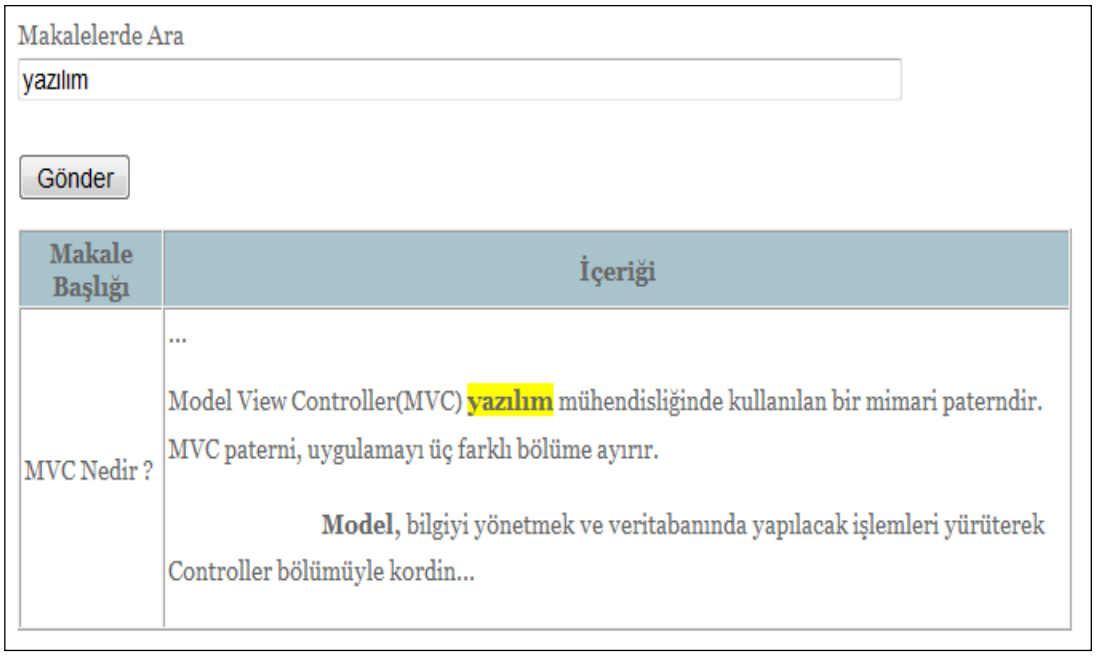
FROM cümlesi sys.dm_fts_index_keywords_by_document fonksiyonunu kullanarak uyuşan kelimelerin tam metin indeksten doğrudan getirilmesini sağlar. Eşleşen kelimenin, etkilenen kelimeler ve eş anlamlılar sözlüğüne istinaden bulunup bulunmadığını kontrol etmek için Sys.dm_fts_parser fonksiyonu kullanılır. PATINDEX fonksiyonu, içerik kolonunda ki metin için "inflectional" aramasına göre eşleşen kelimenin yerini belirler. Son olarak makalenin başlığını ve içeriğini görüntüleyebilmek için gerekli join işlemleri uygulanır.

```
FROM
    (
      SELECT DISTINCT
        c.makaleID,
        w.Display_Term,
        PATINDEX
          (
            N'%[^a-z]' + w.Display_Term + N'[^a-z]%',
            c.Icerik
          ) AS Pos,
        LEN(w.Display_Term) AS Length
      FROM sys.dm_fts_index_keywords_by_document
        (
          @db_id,
          @table_id
        ) w
      INNER JOIN TeknoDestek.Makaleler c
        ON w.document_id = c.makaleID
      WHERE w.column_id = @column_id
        AND EXISTS
          (
            SELECT 1
            FROM #uyusan_dokumanlar m
            WHERE m.dok_id = w.document_id
          )
        AND EXISTS
          (
            SELECT 1
            FROM sys.dm_fts_parser
              (
                N'FORMSOF(FREETEXT, "' + @AranacakText + N'")',
                1055,
                0,
                1
              ) p
            WHERE p.Display_Term = w.Display_Term
          )
    ) s
    INNER JOIN TeknoDestek.Makaleler c
    ON s.MakaleID = c.MakaleID
  GROUP BY
```

```
    s.MakaleID,c.Baslik
```

Prosedürü çağıran C# kodu ve ekran çıktısı aşağıda gösterilmiştir.

```
public ActionResult MakaleAra(string aranacakText)
        {
      aranacakText = aranacakText ?? "?";
      var ViewData =
      MakaleSorgulari.GetirMakaleAramaSonuc(aranacakText.Replace("\"
", ""));
       return View(ViewData);
       }
```



Şekil 5-3 Arama sonucunun ekran çıktısı

5.2.3. Otomatik Tamamlama (Auto Complete)

Kullanıcılara aramaları hakkında kolaylık sağlamak adına makale kayıtları başlıklarını gruplandırarak listeleyen bir otomatik tamamlama uygulaması eklenmiştir [2]. Uygulamanın sorgusu ve ekran çıktısı aşağıda verilmiştir

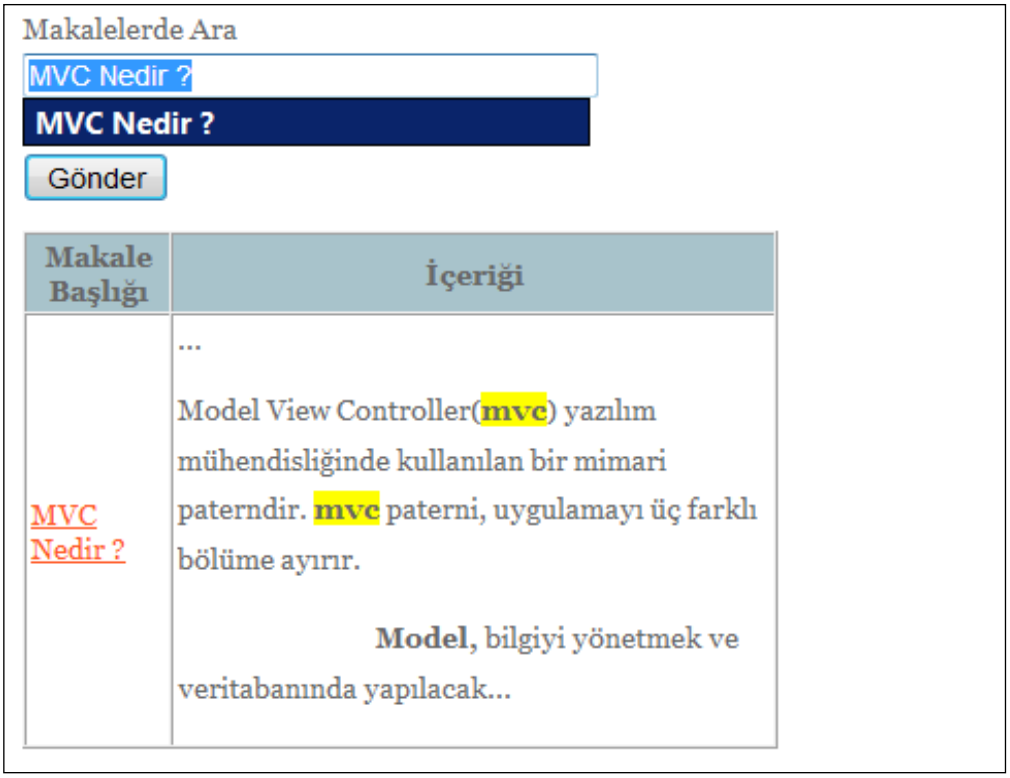
```
public static string[] MakaleAutoComplete(string makaleBaslik)
        {
       myTeknoDestekDataContext dc = new myTeknoDestekDataContext();
```

```
        var myMakale = from p in dc.Makalelers
                       where SqlMethods.Like(p.Baslik, "%" +
                       makaleBaslik + "%")
                           group p by p.Baslik into c
                           select c;

        List<string> sList = new List<string>();

        foreach (var m in myMakale)
        {
            sList.Add(m.Select(p=>p.Baslik).FirstOrDefault());
        }

        return sList.ToArray();
    }
```



Şekil 5-4 Otomatik tamamlama ekran çıktısı

Şekil 5-4'de gösterilen örnekte, kullanıcı MVC yazdığında çalışan SQL sorgusu sonucu görünen otomatik tamamlama penceresi görülmektedir.

5.3.Fonetik Algoritmalar

Web ortamında yapılan aramaların büyük çoğunluğu varlıkların veya bir nesnenin ismini yazarak yapılmaktadır [10]. Veritabanı yöneticileri sistem geliştirirken isim aramaları üzerine yoğunlaşmalı ve kullanıcıların isimleri yanlış yazabilme olasılıklarını göz önünde bulundurarak kullanıcıya öneriler sunabilmelidirler.

Fonetik algoritma, bu önerileri sunmak amacıyla kullanılabilen, kelimeleri telafuzlarına göre indeksleyen algoritmalara verilen isimdir. R. C. Russell tarafından ilk kez 1890 yılında Soundex adı verilen ilk algoritma tasarlanmıştır. Soundex, günümüzde yaygın olarak kullanılan çoğu Fonetik algoritmanın temelini oluşturmaktadır. Bu algoritma eğitim ve arama gibi alanlarda sayısız varyasyonlarda kullanılmıştır ve halen kullanılmaktadır.

5.3.1. SOUNDEX Algoritması

Soundex algoritması, bir ismi ilki bir harf, diğer üçü ise rakamlardan oluşan dört karakterli olacak şekilde kodlar; harf olarak kelimenin ilk harfi, sayı olarak da geri kalan harflerin belirli bir kurala göre numaralandırılmasıyla ortaya çıkan sonuç alınır [12]. Benzer telaffuzlu sözcükler ise aynı numaralarla kodlanır; örneğin, benzer telaffuzlu **B**, **F**, **P** ve **V** harfleri **1** ile kodlanır. Sesli harfler kodlama işlemini etkiler, ancak bu sesli harf kelimenin başında bulunmuyorsa ortaya çıkan sonucu asla doğrudan etkilemez.

Tam algoritma aşağıdaki gibidir:

1. Karakter katarının ilk harfi yakalanır.
2. Eğer ilk harf "a, e, h, i, o, u, w, y" harflerinden herhangi biri değilse, bu harfler metinden silinir.
3. Sırasıyla tüm harflere aşağıdaki numaralandırma yapılır;
    - b, f, p, v = 1
    - c, g, j, k, q, s, x, z = 2
    - d, t = 3
    - l = 4
    - m, n = 5
    - r = 6

4. Eğer numaralandırmada aynı numarayı almış iki ya da daha fazla harf yan yanaysa (ilk işlemden önce) ya da bu harflerin arasında h veya w harfi varsa (sadece Amerikan sayımında geçerli) aynı olanlar atlanır.
5. İlk dört karakter sonuç olarak döndürülür. Eğer sonuç dört karakterden az çıkıyorsa, dört karakter tamamlanması için sonuna sıfırlar eklenir.

SQL Server 2008 içerisinde Soundex algoritma fonksiyonunu barındırmaktadır. Bu algoritma "Distance" parametresiyle birlikte kullanılabilmektedir. Bu parametre 0(en kötü eşleşme) ile 4(en iyi eşleşme) arasında değerler alabilir. Aşağıda DIFFERENCE ve SOUNDEX fonksiyonları, dbo.soyadlar tablosundan "Soundex koduna göre **SARIYAKA** soyadına yakın soyadlar listesini getirmiştir.

```
SELECT
id,
Soyad,
SOUNDEX(soyad) AS Soyad_Soundex,
DIFFERENCE(soyad, N'SARIYAKA') AS Soundex_Difference
FROM dbo.soyadlar
WHERE DIFFERENCE(soyad, N'SARIYAKA') >= 3;
```

Results | Messages

	id	Soyad	Soyad_Soundex	Soundex_Difference
1	15	SARIKAYA	S620	4
2	16	SARIYAKA	S620	4
3	17	LARIYAKA	L620	3
4	18	SARIKALA	S624	3

Şekil 5-5 Soundex isim karşılaştırma tablosu

Bu tez kapsamında geliştirilmiş olan projede Soundex algoritması kullanılarak, kullanıcılara makale başlıkları için öneriler sunulmaktadır.

6. SONUÇ

Geliştirilmiş olan web uygulaması sayesinde yazılım, sistem uzmanlarının veya benzerlik gösteren farklı alanlarda çalışan teknik insanların bilgilerini, tecrübelerini ve kendi pratik çözümlerini diğer meslektaşlarıyla paylaşmalarına ve ayrıca birbirlerinin problemlerini çözmelerine ortam sağlayacak bir platform oluşturulmuştur.

Uygulama geliştirme sürecinde dikkat edilen en önemli husus, mevcut teknolojiler içerisinde günümüz ihtiyaçları ve teknolojileri içinde mimari bir kılavuz oluşturulmaya çalışılmış olmasıdır. Sadelik ve içerik temizliği öncelikli olarak hedeflenmiştir. Sistem güvenliği, bakımı ve geliştirilebilir olmasına dikkat edilmiş bu nedenle uygulama katmanlara ayrılmıştır.

Geliştirilen uygulama, kullanıcılardan gelecek olan talep ve ihtiyaçlar doğrultusunda geliştirilebilir ve daha esnek hale getirilebilir. İyileştirme açısından öneriler şunlardır;

- Forum bölümü için hiyerarşik sınırsız kategori oluşturulabilir.
- Arma bölümüne forum için kullanıcılara farklı kategorilerde arama yapma olanağı sağlanabilir.
- Video paylaşımı için modifikasyon yapılabilir.
- İçeriğin farklı dillerde oluşturulmasına olanak sağlanıp uluslararası bir platform haline dönüştürülebilir.

KAYNAKLAR

1. Thakur, V.(2008) , *Asp.NET 3.5 Application Architecture and Design*, Packt Publishing Ltd.

2. Jeffrey, P., BEN S. ve Jimmy B. (2010), *Asp.NET MVC in Action* , Manning Publications Co.

3. Patrick, L. and Sarah, H. (2009) ,*WEB STYLE GUIDE*. 3nd edition**,** Yale University Press, Book URL: http://www.webstyleguide.com/wsg3/index.html

4. Jennifer, N. R. (2007), *Learning Web Design A Beginner's Guide to (X)HTML, Style Sheets, and Web Graphic*. Third Edition, O'Reilly Media, Inc.

5. Michael, C ve Hilary, C. (2009), *Pro Full-Text Search in SQL Server 2008*, Second Edition, Springer-Verlag New York Inc.

6. Eric, E., Stephan, S., Rand, F. and Jessie, S. (2010), *The Art of SEO*, First Edition, O'Reilly Media, Inc.

7. Jerri, L. (2009), *Search Engine Optimization*, Second Edition, Wiley Publishing Inc.

8. Jonathan, C. ve Karl, S. (2009), *Learning Jquery 1.3*, First Published, Packt Publishing Ltd.

9. El-Sayed, A., Elmarhomy, G. ve Jun-ichi, A. (2004), *A Fast Dynamic Full-Text Search Method Using Efficient Block Management Structure Masao Fuketa*, University of Tokushima

10. Chakkrit, S. ve Michael, B. (2009), *Novel Phonetic Name Matching Algorithm with a Statistical Ontology for Analysing Names Given in Accordance with Thai Astrology*, Naresuan University, Phitsanulok, Thailand

11. UML Use Case Diagrams, Engineering Notebook C++ Report Nov-Dec 98

Internet WWW-page,

URL: http://www.objectmentor.com/resources/articles/usecases.pdf

12. Method of Soundex and compares it to another algorithms which today used and can bring more reliable hit quality and higher rate. Internet WWW-page, URL: http://www.sound-ex.com/index.html

13. Scott Guthrie, 2007, Corporate Vice President in the Microsoft Developer Division, ASP.NET MVC Framework Internet WWW-page, URL: http://weblogs.asp.net/scottgu/archive/2007/10/14/asp-net-mvc-framework.aspx
URL : http://weblogs.asp.net/scottgu/archive/2007/07/11/linq-to-sql-part-4-updating-our-database.aspx

14. Search Engine Optimization, Internet WWW-page, URL: http://www.sitemaps.org

15. Comparing Web Forms And ASP.NET MVC, Dino Esposito URL: http://msdn.microsoft.com/en-us/magazine/dd942833.aspx

16. Prof. Jim Whitehead (2006), *SQL Injection Attacks*,WWW-page, URL:http://www.soe.ucsc.edu/classes/cmps183/Spring06/lectures/SQL%20Injection%20Attacks.pdf

17. Maarten B. (2009), *ASP.NET MVC 1.0 Quickly*, First Edition, Packt Publishing Inc.