Ibrokhim SHOKIROV

141816060


GAZİ ÜNİVERSİTESİ
Gazi
1926

MİLLİ ARAMA MOTORU UYGULAMASI: ARATURKA

Ibrokhim SHOKIROV

LİSANS TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

GAZİ ÜNİVERSİTESİ

TEKNOLOJİ FAKÜLTESİ

ARALIK 2017

Ibrokhim SHOKIROV tarafından hazırlanan "MİLLİ ARAMA MOTORU UYGULAMASI: ARATURKA" adlı bu tezin Lisans tezi olarak uygun olduğunu onaylarım.

Doç. Dr. Necaattin BARIŞÇI .....

Bu çalışma, jürimiz tarafından oy birliği ile Bilgisayar Mühendisliği Bölümünde Lisans tezi olarak kabul edilmiştir.

Doç. Dr. Aydın ÇETİN .....

Doç. Dr. Necaattin BARIŞÇI .....

Yrd. Doç. Dr. Hüseyin POLAT .....

Tez Savunma Tarihi : ...../...../.....

Bu tez, Gazi Üniversitesi Teknoloji Fakültesi Bilgisayar Mühendisliği'nce onaylanmıştır.

.....

Prof. Dr. Recep DEMİRCİ

Bilgisayar Mühendisliği Bölüm Başkanı

ETİK BEYAN

Gazi Üniversitesi Teknoloji Fakültesi Tez Yazım Kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- Tez içinde sunduğum bilgi ve dokümanları akademik kurallar etik çerçevesinde elde ettiğimi,
- Tüm bilgi, belge, değerlendirme ve sonuçları bilimsel etik ve ahlak kurallarına uygun olarak sunduğumu,
- Tez çalışmamda özgün verilerim dışında kalan ve tezde yararlanılan eserlerin tümüne uygun atıfta bulunarak kaynak gösterdiğimi,
- Kullanılan verilerde herhangi bir değişiklik yapmadığımı,
- Bu tezde sunduğum çalışmanın özgün olduğunu ve başka bir yerde sunmadığımı

Beyan ederim.

Ibrokhim SHOKIROV

MİLLİ ARAMA MOTORU UYGULAMASI: ARATURKA

(Lisans Tezi)

Ibrokhim SHOKIROV

GAZİ ÜNİVERSİTESİ

TEKNOLOJİ FAKÜLTESİ

Aralık 2017

ÖZET

Teknolojinin ve bilimin gelişmesi sayesinde sahip olunan bilginin sayısı ve içeriği artmıştır. Bu sebeple ulaşılmak istenen bilginin diğerlerinin içerisinden ayırt edilmesi, doğru bilgiye hızlı ve kolay yoldan sahip olunması ihtiyacı ortaya çıkmıştır. Günümüzde bu ihtiyacı karşılamak için akla gelen ve başvurulan ilk yol internet üzerinden araştırma yapmaktır. Bu yapılan araştırmayı kolay hale getirmek için arama motorları oluşturulmuştur.

Günümüzde birçok sayıda arama motoru bulunmaktadır. Bunlardan en çok bilinen ve tercih edilenleri Google, Yandex gibi Amerika ve Rusya kökenli arama motorlarıdır. Fakat günümüzde sahip olunan en değerli şeylerden biri olarak görülen bilginin korunması ve saklanması gerekmektedir. Bu arama motorları gibi yabancı kökenli arama motorları kullanılarak arama bilgileri yurt dışına çıkmış olacaktır. Oysa günümüzdeki durum göz önünde bulundurulduğunda sahip olunan bilgi ve birikim ile milli sistemlerin geliştirilmesi ve kullanılmasının günden güne daha çok önem kazandığı görülmektedir.

Türkiye'nin sahip olduğu birkaç tane milli arama motoru bulunmaktadır. Fakat bu arama motorları Google üzerinden arama yapıp sonuçları kullanıcıya göstermektedir. Arama Google üzerinden gerçekleştirildiği için arama bilgileri yurt içerisinde kalmamaktadır.

Bu nedenlerden dolayı yapılan çalışmada AraTurka isimli milli arama motoru geliştirilmesi amaçlanmıştır.

Anahtar Kelimeler : Arama Motoru, Milli Arama Motoru, Türkçe Arama Motoru, Veri Madenciliği, Metin Madenciliği, RAKE Algoritması

Sayfa Adedi : 70

Tez Yöneticisi : Doç. Dr. Necaattin BARIŞÇI

NATIONAL SEARCH ENGINE APPLICATION: ARATURKA

(BSc. Thesis)

Ibrokhim SHOKIROV

GAZİ UNIVERSITY

FACULTY OF TECHNOLOGY

December 2017

ABSTRACT

Thanks to the development of technology and science, the number and content of knowledge have increased. For this reason, the need to distinguish the information to be accessed from others, the need to have the right information quickly and easily has emerged. Nowadays, the first thing that comes to mind and needs to be addressed is to do research on the internet. Search engines have been created to make this research easier.

Today, there are many search engines. The most well known and preferred ones are American and Russian based search engines like Google, Yandex. But it is necessary to protect and hide the information that is regarded as one of the most precious things today. By using foreign origin search engines such as these search engines, search information will be exported abroad. However, considering the present situation, it is seen that the development and use of national systems and knowledge and know-how gained more importance from day to day.

Turkey has owned several national search engines. But these search engines search through Google and show the results to the user. The search information is not available in the country because the search is done via Google.

For these reasons, it is aimed to develop a national search engine named AraTurka.

Key Words : Search Engine, National Search Engine, Turkish Search Engine, Data Mining, Text Mining, RAKE Algorithm

Page Number : 70

Adviser : Assoc. Prof. Dr. Necaattin BARIŞÇI

TEŞEKKÜR

Bu çalışmada danışmanım olarak tezin yazılmasında yol gösteren sayın hocam **Doç. Dr. Necaattin BARIŞÇI**'ya, her konuda yardımını ve desteğini esirgemeyen sayın hocalarım **Arş. Gör. Esra Söğüt** ve **Arş. Gör. Akın EKER**'e, maddi manevi desteğini hiç esirgemeyen **anne** ve **babama**, tezi yazarken beni yalnız bırakmayan ve Türkçe konusunda sürekli destekleyen **Seda Nur ALTUN**, **Umutcan ATA**, **Simge Yıldız ÇETİN**, **Umut Küzeyfe TURKOĞLU**, **Ahmetcan BİNGÖL** ve **Ömer Ayberk ŞENCAN**'a teşekkür ederim.

İÇİNDEKİLER Sayfa

ŞEKİLLERİN LİSTESİ

ŞEKİLLER Sayfa

KISALTMALAR

Bu çalışmada kullanılmış bazı kısaltmalar açıklamaları ile birlikte aşağıda sunulmuştur.

Kısaltma	**Açıklama**
HTML	Hypertext Markup Language
SMGL	Standard Generalized Markup Language
CSS	Cascading Style Sheet
ASP	Active Server Page
MSSQL	Microsoft Structured Query Language
CIL	Common Intermediate Language
IIS	Internet Information Services
MVC	Model View Controller
RAKE	Rapid Automatic Keyword Extraction

1. GİRİŞ

İnternetin hızlı gelişimiyle birlikte web sitelerinin sayısı ve aranılan bilginin içeriği çoğalmaktadır. Bu sebeple istenilen içeriklere ulaşmak gittikçe zorlaşmaktadır. Buna çözüm olarak ihtiyaç duyulan bilgileri içeren web sitelerini daha kolay bulmayı sağlayan arama motorları oluşturulmuştur. Arama motorları yardımıyla istenilen içerik çok hızlı bir şekilde bulunabilir.

Bu sistemin faydalarını gören insanlar diğer ülkelerden de kullanıcı kitlesine sahip olmak için sistemlerini çoklu dil destekli hale getirmiştir. Çoklu dil destekli arama motorlarından en bilindik olanı ise Google'dır. 4 Eylül 1998 yılında çalışmaya başlayan bu sistem şuan elliden daha fazla dili desteklemektedir.

Google'a benzer olarak milli arama motorları da geliştirilmiştir. Örneğin Baidu Çin ve Japonya halkı için, Maktoob Arap halkı için geliştirilmiştir. Buna benzer olarak birçok ülke kendi dilinde kendi halkına özel arama motorları geliştirmiştir. Bu sebeplerden dolayı yapılan tez çalışmasında Türk halkına ve kültürüne hizmet etmesi için bir arama motoru geliştirilmesi amaçlanmıştır.

Türk halkı için geliştirilmiş birkaç arama motoruna bakılırsa bu sistemlerin aslında sadece arayüzden oluşturuldukları görülmektedir. Yani bu sistemler aslında aranan sorguyu başka arama motorlara gönderip oradan dönen cevabı süsledikten sonra kullanıcıya göstermektedirler. Bu sistemler diğer arama motorlarının bir teması olarak adlandırılabilirler.

Bu problemin çözümü tabiki de vardır. Eğer arama motorların çalışmasına yakından bakılırsa iki alt sistemden oluşturuldukları görülür. Bu iki alt sistem Robot ve İstemci olarak adlandırılabilir. Robot tüm siteleri gezip onların içeriğini kaydedip saklar. Aynı zamanda sitelerin içerisindeki linkler kaydedilerek kaydedilen linkler de gezilir. Bunun dışında eğer sistemin gezilecek bir linki yoksa sistem tarafından IP tahmin etme yöntemini kullanılarak yeni bir site aranır ya da sistem yöneticisiden link istenir ve girilen linke gidilir. İstemci ise burda kullanıcı ile ilgilenmektedir. Yani kullanıcının aradığı sorguyu kayıtlı bilgiler arasında arar ve sonuçlarını kullanıcıya gösterir.

Tez kapsamında üzerinde çalışılan ve "AraTurka" ismi verilen arama motoru da içerisinde diğer arama motorları gibi iki adet alt sistem barındırmaktadır. Alt sistemlerden biri olan Robot çalıştırıldığında 1 saat içinde yaklaşık olarak 500 farklı alan adı ve 5.000 link taranıp

120.000 anahtar kelime toplayabilir. Buna bakarak bu sistem bir sene içerisinde yine yaklaşık olarak 4.392.000 alan adı, 43.920.000 link ve 1.051.200.000 anahtar kelime toplamış olacaktır. Tabi ki bu verileri saklamak için büyük çaplı sunucular gerekmektedir. Bu sistemin sonraki aşamaları olacaktır.

2. LİTERATÜR ÇALIŞMALARI

İnternet kullanan herkes günde en az bir kere arama motoru kullanmaktadır. "Internet Live Stats" tarafından yayınlanan dünyada internet kullanımı istatistiklerine göre yaklaşık 3,5 milyar internet kullanıcısı ve 1,15 milyardan fazla web sitesi mevcuttur. Bu sitelere ait 50 trilyondan fazla web sayfası olduğu tahmin edilmektedir. Her biri farklı bir siteye ait olan bu sayfalar, internetteki tüm içerikleri oluşturmaktadır.

Bir gazetenin internet sitesinde yayınlanan haberden tutun da en çok kullanılan sosyal medya sitelerinden birisi olan Facebook'ta oluşturulan bir profile kadar her adres (URL) bir web sayfasıdır. Trilyonlarca web sayfasında bir markanın tanıtım yazısından, bir ürünün sipariş edilebileceği sayfaya kadar neredeyse her içerik bulunabilmektedir.

İnternette bu kadar çok web sayfası olmasının avantajları olduğu gibi dezavantajları da vardır. İhtiyaç duyduğunuz herhangi bir şeyi içeren web sayfalarının sayısının fazla olması çeşitlilik sağlarken, aynı zamanda hangisinin işinize en çok yarayacağını bulmak zaman maliyetini de beraberinde getirmektedir.

İşte internet teknolojisinin son kullanıcıya en çok ulaşılabilir olduğu yıllarda artan web sitesi sayısı insanlara çeşitlilik sunduğu gibi beraberinde "Peki benim ihtiyacımı karşılayabilecek en iyi web sayfasını nasıl bulabilirim?" veya "Acaba bulduğum web sayfalarından daha iyi bir web sayfası olabilir mi?" gibi soruları da beraberinde getirmiştir.

Tüm bu sorular insanların aradıkları sayfaları kolayca bulabilecekleri bir araç ihtiyacını oluşturmuştur. Bu ihtiyacı karşılamak için ise başta Google, Yandex, Yahoo! Ve Bing olmak üzere çeşitli arama motorları geliştirilmiştir. Özellikle dünyanın en çok kullanılan arama motoru olan Google bu ihtiyacı neredeyse tamamen karşılayabilmektedir.

Arama motoru, internet kullanıcılarının ihtiyaç duyduğu herhangi bir bilgiyi içeren en iyi web sayfalarını arayabildiği sistemdir. İhtiyaç duyulan şey bir markanın web sitesi, bir kişinin sosyal medya sitesindeki profili, yerel bir haber, bir ürün veya hizmet ya da akademik makale olabilmektedir [1].

2.1. Arama Motorların Tarihçesi

Arama motorların tarihçesine bakarsak, ilk arama motoru 1990 yılında bir üniversite öğrencisi olan Alan Emtage tarafından Archie adıyla kurulmuştur. İngilizce "archive" kelimesinden türetilmiştir. Bu arama motoru insanların aradıkları dosyaları bulmaya

çalışmaktaydı. Popüler olmaya başladığında Minnesota Üniversitesi'nden Mark P. McCahill, "www.archie.com"un karşısına 1991'de Veronica (**V**ery **E**asy **R**odent-**O**riented **N**et-wide **I**ndex to **C**omputerized **A**rchives)'yı çıkarmıştır. Çok geçmeden aynı amaçla Jughead (**J**onzy's **U**niversal **G**opher **H**ierarchy **E**xcavation **A**nd **D**isplay) de kuruldu. İkisi de dosya aktarım iletişim kuralı (FTP) çerçevesinde çalışmıştır.

Haziran 1993'te Massachusetts Teknoloji Enstitüsü'nden Matthew Gray tarafından, bir indeks adı oluşturmak için kullanılan ve "Wandex" adıyla anılan ilk internet botunu üretilmiştir. Sonra Kasım 1993'te kurulan ve internet botu olmayan Aliweb arama motoru, web sitelerinin bilgilerini kullanarak oluşturulmuş ilk arama motoru olmaktadır.

Aralık 1994'te web sayfalarını bulmak amacıyla kendi içerisinde dizin oluşturan ve tasarlanan sorgu programına arayüz ve bir web formu olarak kullanılabilen JumpStation arama motoru oluşturulmuştur. Bir ilk olan "tam metin"(full text) tarayıcı arama motoru olan WebCrawler, 1994 yılında görücüye çıkarılmıştır. Önceki arama motorlarının aksine; herhangi bir web sayfasını, her kelimesi için herhangi bir kullanıcının aramasına izin verilmekteydi. Ayrıca yine 1994 yılında Carnegie Mellon Üniversitesi'nden Dr. Michael Mauldini tarafından üretilip satışa çıkan Lycos, büyük bir ticari çaba olmuştur. Kısa bir süre sonra; Magellan, Excite, Infoseek, Inktomi, Northern Light ve AltaVista dahil pek çok arama motoru internet ortamında görücüye çıkarılmıştır ve popülerlik için birbiriyle yarışmıştır. Ancak bunların içinde bulunan, David Filo ve Jerry Yang'ın kurduğu Yahoo!, insanların ilgisini diğerlerinden çok çekerek web sayfaları bulmanın en popüler yolu olarak kullanılan arama motorları arasında yer almıştır.

Arama motorlarına 1990'ların sonlarına kadar büyük çapta bir yatırım yapılmamıştır. Ancak o yıllardan itibaren büyük şirketler ortaya çıkan bu yeni arama motorlarından kazanç elde etmeye başlamıştır. 1998 yılında Google'ı kuran Larry Page ve Sergey Brin, PageRank adlı teknolojilerini satmak istediler ancak alıcı çıkmamıştır. İnternet ağındaki her sayfayı puanlayan bu sistem; o sayfaya ne kadar çok link verildiyse ve link veren yerlerin puanı ne kadar çoksa, söz konusu sayfaya da o kadar çok puan verme mantığına dayanmaktaydı. Google'ın sahipleri bu teknolojiyi satamayınca büyüme kararı almış ve 35 milyon dolar yatırım kredisi de alınca 1999'da Google Search'ü kurmuşlardır. 2000'li yıllarda öne çıkan Google arama motoru'nun ardından kısa sürede gelişip 2000 yılında Google Araç Çubuğu çıkaran ekip, 2004 yılında kütüphanelerdeki binlerce kitabı Google Book Search adıyla aramaya açmıştır ve 1 GB kapasite ile Google Mail yani Gmail hizmetini başlatmışlardır.

Google 2007'nin sonları itibarıyla, en popüler web arama motoru olarak dünya çapında tanınmaya başlamıştır [2].

2.2. Anlamsal Arama Yöntemi

2008 yılında ortaya çıkmaya başlayan anlamsal arama motorları, dil üzerine yoğunlaşmıştır. Mesela Hakia bunlardan biridir. Anlamsal aramada ön sıralara çıkmak için, anahtar kelimeleri yalnızca doğru yerlere yazmanız yeterli olmamaktadır. İnsanların ne aramak isteyeceğini de iyi bilmeniz gerekmektedir. Kullanıcı arama yaptığında, bir soruya cevap bulmak ister. Anahtar kelime teknolojisini etkili kullanan bir kullanıcı ise, sorunun içinde geçen anahtar kelimeleri aratmakla yetinmek zorunda kalmaktadır. Anlamsal aramada, işler değişecektir. Yalnızca anahtar kelimeleri aratmak yerine, soruyu yazıp, gerçek anlamına uygun cevaplar elde etmek mümkün olmaktadır.

Mesela, insanlar "yoga" kelimesini aratarak neyi arıyor olabilir?

- Yoga nedir?
- Değişik yoga çeşitleri
- Değişik yoga pozisyonları nasıl yapılır
- Yoga kıyafetleri nasıl seçilir
- Yoga egzersizleri videoları

Olasılıkların sonu yoktur. Bu yüzden bunu diğer insanlar en basit hangi şekilde sorar, diye düşünürseniz, anlamsal arama motorunun yükü hafiflemiş olacaktır. Böylece dilin en doğal halini kullanmak en doğru sonuçları verecektir [3].

Anlamsal arama yöntemini kullanan arama motorlarının en bilindikleri aşağıda belirtilmiştir;

- Ask.com
- Bing
- Hakia
- Lexxe

2.3. Benzer Çalışmalar/Uygulamalar

Günümüzde artık bir çok arama motoru mevcuttur ve her biri farklı kategorilerde arama yapmaktadır. Aşağıda bu kategorilerin listesi verilmiştir;

- Genel
- Muhasebecilik
- İş ve Ticaret
- Eğitim
- Gıda ve Yemek Tarifleri
- Haber
- İnsan
- Tıbbi

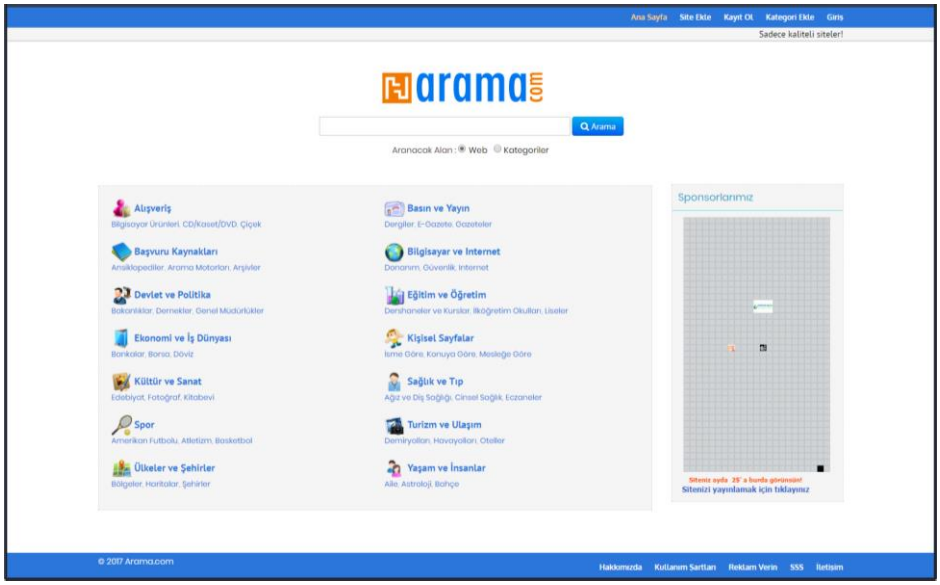
Aynı zamanda çoğu ülkelerde milli arama motorları geliştirildi ki bu arama motorları o ülkede yaşayanlar ve ülkenin diline hakim olan insanlar tarafından kullanılmaktadır. Bu arama motorlarından bir kaç tanesi aşağıda belirtilmiştir;

- Rediff (Hindistan)
- Rambler (Rusya)
- Walla! (İsrail)
- Naver (Kore)
- Maktoob (Arap Ülkeler)
- Goo (Japonya)

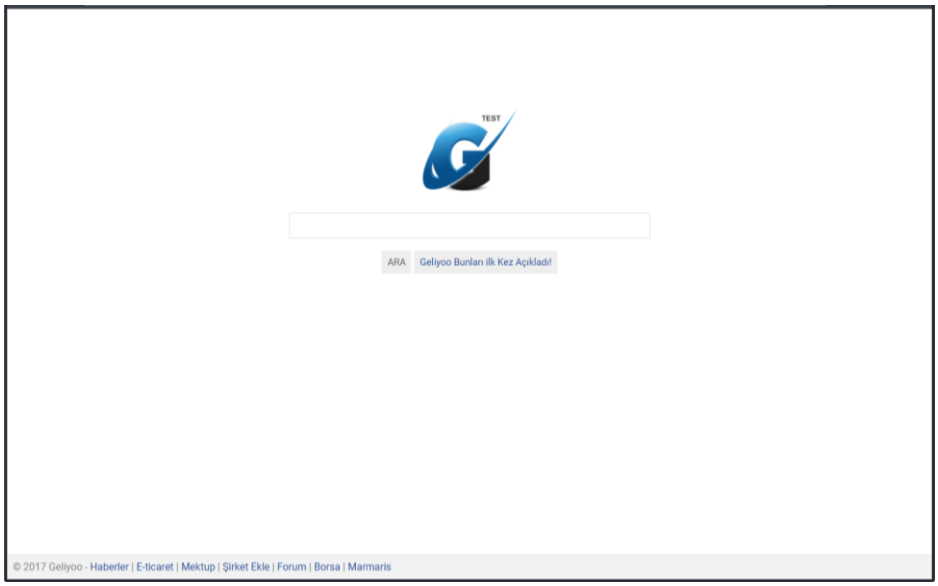
Türk arama motorlarına bakarsak, karşımıza sayılı bir miktarda site çıkacak ki bunların çoğu ya başka arama motorlarını kullanarak çalışmaktalar ya da artık kullanım dışı bırakılmıştır. Kısacası bu arama motorlarına diğer arama motorlarının birer teması diyebiliriz. Bu arama motorlarında bir kaç tanesi aşağıda listelenmiştir;

- Ara.com.tr – Kullanım dışı
- Attabot.com – Kullanım dışı
- Ara.ma – Kullanım dışı

Başka arama motorlarına bağlı olmayan iki adet Türk arama motoru bulunmaktadır. Bu arama motorları ise arama.com ve Geliyoo olarak adlandırılmıştır. Bu iki arama motoru kendi robotlarına sahipler ve bu robotlar sayesinde kendileri için büyük miktarda siteleri endekslemişlerdir. Bu arama motorların arayüzleri de Şekil 2.1 ve Şekil 2.2'de verilmiştir.



Şekil 2.1. Arama.com arama motorunun ana sayfası



Şekil 2.2. Geliyoo.com arama motorunun anasayfası

3. MATERYAL VE METOD

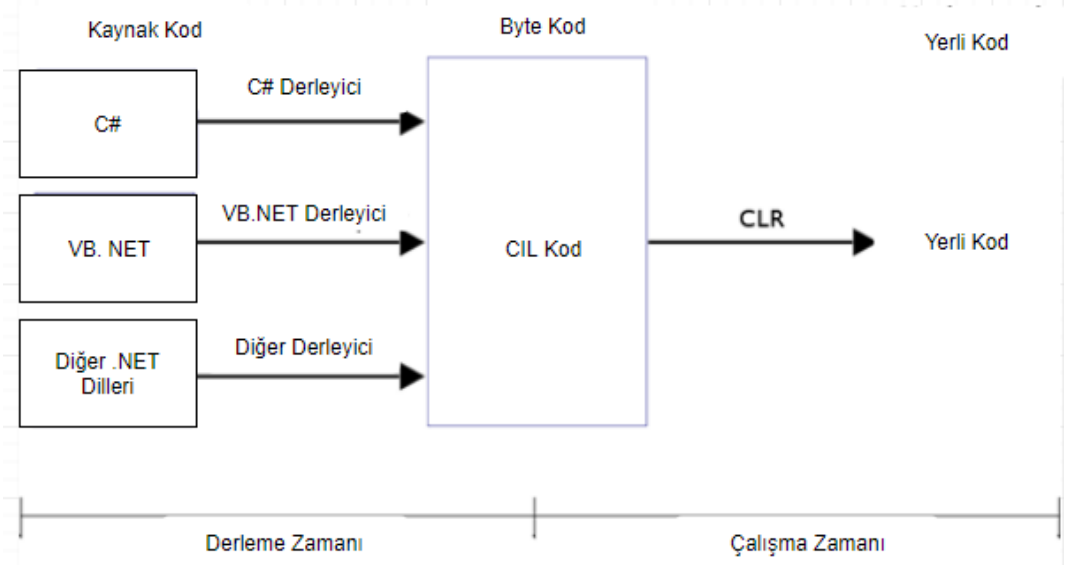
"AraTurka" uygulamasını geliştirmek için bir kaç teknolojinin bir arada çalışması gerekmektedir. Bunun dışında uygulamanın düzgün çalışması için iki alt uygulama olarak hazırlanması gerekmektedir. Bu iki alt uygulamada kullanılan teknolojilerin sürümü birbirine uyumlu olmalıdır. Yapılan çalışmada sözü geçen teknolojilerin mümkün olduğu kadar son sürümleri kullanılmaya çalışılmıştır. Bu teknolojiler aşağıda anlatılmıştır.

3.1. .NET Framework

.NET Framework, Microsoft tarafından geliştirilen, açık İnternet protokolleri ve standartları üzerine kurulmuş bir "uygulama" geliştirme platformudur. Daha önce Sun Microsystems tarafından geliştirilmiş olan Java platformuna önemli benzerlikler göstermektedir.

Buradaki uygulama kavramının kapsamı çok geniştir. Bir masaüstü uygulamasından bir web tarayıcı uygulamasına kadar her şey bu platform içinde düşünülmüştür ve desteklenmiştir. Bu uygulamaların birbirleriyle ve geliştirildiği ortam farketmeksizin dünyadaki tüm uygulamalarla iletişimi için kolayca web servisleri oluşturulmasına imkân verilmiştir. Bu platform, işletim sisteminden ve donanımdan daha üst seviyede taşınabilir olarak tasarlanmıştır.

.NET tabanlı diller genellikle birbirine benzer bir şekilde derlenip çalışmaktadırlar. Şekil 3.1'de .Net tabanlı dillerin derleme ve çalışma aşamaları gösterilmiştir.



Şekil 3.1. .Net tabanlı dillerin derlenmesi ve çalışması

.Net mimarisi, ortak bir yürütme ortamı, ortak bir değişken tür sistemi, ve devingen bağlantılı kütüphanelerden oluşur. .Net kütüphanesi eski visual basic için tasarlanmış Application Programming Interface (API) lerin sınıflanmış halidir. API'ler sınıflandırılmamış olduğundan programcılar için bir kâbus halini almaktaydı. .Net kütüphanesi programın işletim sistemi ile kolayca uyum içinde çalışmasını sağlamıştır [4].

3.2. ASP

Active Server Pages (Etkin Sunucu Sayfaları) kısaca ASP, Microsoft'un ilk dinamik web sayfaları üretmek için geliştirdiği sunucu taraflı betik motorudur. Klasik ASP ya da ASP Klasik olarak da bilinir. Bir ASP dosyasının içinde, özel nesneler ve VBS, JS, SQL kodları bulunur, bu sayfalar istemci tarafından istendiğinde sunucu öncelikle ASP içindeki kodları icra ederek, istemciye göndereceği bilgiyi oluşturur ve gönderir. Gönderilen bilgi genellikle HTML (Hypertext Markup Language) ya da SGML (Standard Generalized Markup Language) şeklindedir. Fakat sadece bunlarla sınırlı değildir, aynı şekilde bir grafik dosyası da oluşturulup, istemciye gönderilebilinir. ASP sayfaları HTML kodlarının içine <% ve %> ASP taglarıyla gömülü şekilde oluşturulduğu halde bir kez sunucu tarafından yorumlandığında saf HTML olarak döner. Kaynak kodlara bakıldığında ASP kodları görülmez. Bu kodlamacıların kaynaklarını saklamalarını kolaylaştırır. ASP'nin ortaya çıkış nedenlerinden birisi de CGI dillerinin Oturum (Session) ve Uygulamaların (Application) başından sonuna kadar izlenmesinin yetersiz oluşundandır.

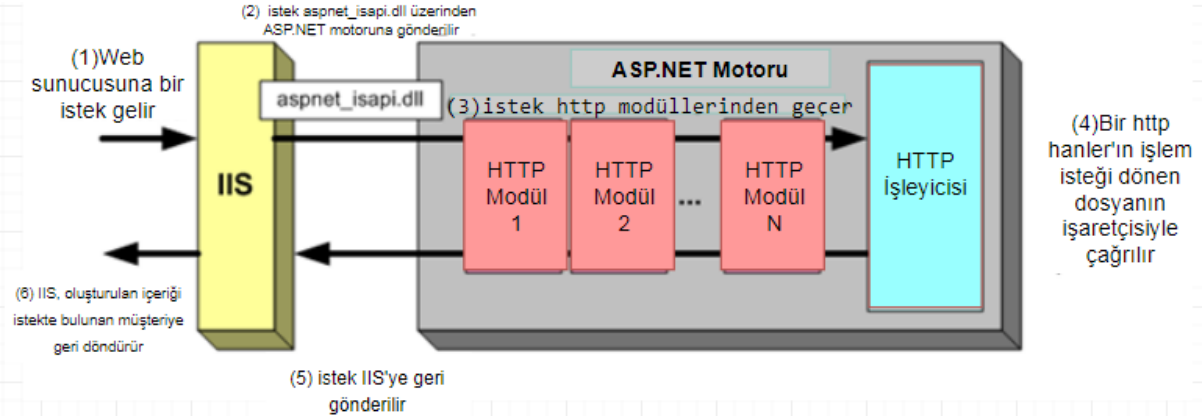
3.2.1. ASP.NET

ASP.NET, Microsoft tarafından geliştirilmiş bir web uygulama gelişimi teknolojisidir. Özdevinimli (dinamik) web sayfaları, web uygulamaları ve XML tabanlı web hizmetleri geliştirilmesine olanak sağlar. Aynı işletme tarafından geliştirilen .Net çatısı'nın (framework) parçası ve artık işletmece desteklenmeyen ASP teknolojisinin devamını teşkil etmiştir [5].

Her ne kadar isim benzerliği olsa da ASP.NET, ASP'ye oranla çok ciddi bir değişim geçirmiştir. ASP.NET kodu ortak dil çalışma zamanı (ingilizce - common language runtime) altyapısına dayalı çalışır, diğer bir deyişle, yazılımcılar .Net çatısı tarafından desteklenen tüm dilleri ASP.NET uygulamaları geliştirmek için kullanabilirler. Yani, Java teknolojisinde

olduğu gibi, yazılımcı tarafından yazılan kod, çalıştırılmadan önce sanal bir yazılım katmanı tarafından ortak bir dile çevirilmektedir.

ASP.NET uygulamaların çalışabilmeleri için Internet Information Service (IIS) gerekmektedir. ASP.NET ve IIS arasındaki iş birliği Şekil 3.2'de gösterilmiştir.



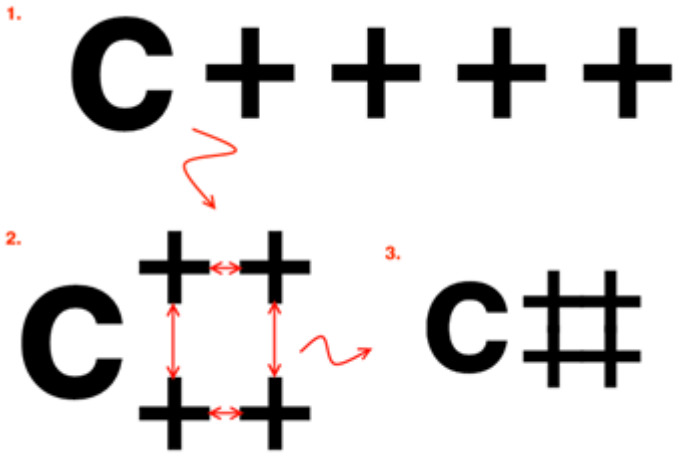
Şekil 3.2. ASP.NET ve IIS çalışması

ASP.NET belgeleri ASPX yapısını kullanırlar. ASPX belgesi, özdevinimsiz (statik) HTML veya XHTML tabanlı, web sayfasında belirecek olan içeriği ayrı tutar. Kullanılan dile göre, sunucu tarafından çalıştırılacak kodu içeren belge Sayfam.aspx.cs (C#) veya Sayfam.aspx.vb (VB.NET) olarak adlandırılabilir. Bu yaklaşım ile, yazılımcı kodunu yordamsal programlama ilkelerinden çok, oluşabilecek olaylara gereken tepkiyi verebilecek biçimde geliştirir; örneğin, bir sayfa yüklenince, bir düğmeye basılınca, vb. ASP.NET'in kullandığı diğer belge türleri arasında ascx, asmx, ashx, master, sitemap, skin ve config sayılabilir.

3.3. C#

C# Microsoft tarafından geliştirilmiş olan bir programlama dilidir. C++ ve Java dillerine oldukça benzer, ancak C#'ın bu dillere benzerliği yanında farkları da vardır. Örneğin C#, C++'dan farklı olarak %100 nesne yönelim tekniğine sahiptir. Java'dan farklı olarak ise C#'ta gösterici (pointer) kullanılabilir. Böylelikle eski yazılım bileşenleriyle uyumlu bir şekilde çalışılabilir. Microsoft'un geliştirmiş olduğu bu programlama dili yeni nesil programlama dilidir. C#, .NET Teknolojisi için geliştirilmiş dillerden biridir. Microsoft tarafından geliştirilmiş olsa da ECMA ve ISO standartları altına alınmıştır [6].

C programlama dilinde bir tam sayı değişkeni 1 atırmak için ++ son eki kullanılır. C++ dili adını, C diliyle Nesneye Yönelimli Programlama yapabilmek için eklentiler (C With Classes) almıştır. Benzer şekilde C++ diline yeni eklentiler yapılarak ((C++)++) bir adım daha ileriye götürülmüş ve tamamen nesneye yönelik tasarlanmış C# dilinin isimlendirilmesinde, + karakterlerinin birbirlerine yakınlaşmış hali (Şekil 3.3'te gösterildiği gibi) ve bir melodi anahtarı olan C# Major kullanılmıştır.



Şekil 3.3 C# isminin yapılışı

Bu dilin tasarlanmasına Pascal, Delphi derleyicileri ve J++ programlama dilinin tasarımlarıyla bilinen Anders Hejlsberg liderlik etmiştir.

Birçok alanda Java'yı kendisine örnek alır ve C# da java gibi C ve C++ kod sözdizimine benzer bir kod yapısındadır. .NET kütüphanelerini kullanmak amacıyla yazılan programların çalıştığı bilgisayarlarda uyumlu bir kütüphanenin ve yorumlayıcının bulunması gereklidir. Bu, Microsoft'un .Net Framework'u olabileceği gibi ECMA standartlarına uygun herhangi bir kütüphane ve yorumlayıcı da olabilir. Yaygın diğer kütüphanelere örnek olarak Portable.Net ve Mono verilebilir.

Özellikle nesne yönelimli programlama kavramının gelişmesine katkıda bulunan en aktif programlama dillerinden biridir .NET platformunun anadili olduğu bazı kesimler tarafından kabul görse de bazıları bunun doğru olmadığını savunur.

C#, .NET orta seviyeli programlama dillerindendir. Yani hem makine diline hem de insan algısına eşit seviyededir. Buradaki orta ifadesi dilin gücünü değil makine dili ile günlük

konuşma diline olan mesafesini göstermektedir. Örneğin; Visual Basic .NET (VB.NET) yüksek seviyeli bir dildir dersek bu, dilin insanların günlük yaşantılarında konuşma biçimine yakın şekilde yazıldığını ifade etmektedir. Dolayısıyla VB.NET, C#.NET'ten daha güçlü bir dildir diyemeyiz. Programın çalışması istenen bilgisayarlarda framework kurulu olması gerekmektedir. Windows Vista, 7, 8, 8.1 ve 10'da .NET Framework otomatik olarak kurulu gelmektedir.

3.3.1. Tasarım Hedefleri

ECMA tarafından C# dilinin tasarım hedefleri şöyle sıralanır:

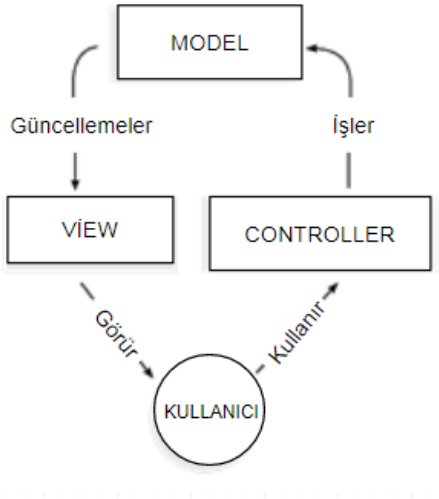
- C# basit, modern, genel-amaçlı, nesneye yönelik programlama dili olarak tasarlanmıştır.
- Çünkü yazılımın sağlamlılığı, güvenirliliği ve programcıların üretkenliliği önemlidir. C# yazılım dili, güçlü tipleme kontrolü (strong type checking), dizin sınırlar kontrolü (array bounds checking), tanımlanmamış değişkenlerin kullanım tespiti, (source code portability), ve otomatik artık veri toplama gibi özelliklerine sahiptir.
- Programcı portatifliği özellikle C ve C++ dilleri ile tecrübesi olanlar için çok önemlidir.
- Uluslararası hale koymak için verilen destek çok önemlidir.
- C# programlama dili sunucu ve gömülü sistemler için tasarlanmıştır. Bununla birlikte C# programlama dili en basit işlevselli fonksiyondan işletim sistemini kullanan en teferruatlısına kadar kapsamaktadır.
- C# uygulamaları hafıza ve işlemci gereksinimleri ile tutumlu olmak uzere tasarlanmıştır. Buna rağmen C# programlama dili performans açısından C veya assembly dili ile rekabet etmek için tasarlanmamıştır [6].

3.4. MVC

Model-view-controller (MVC), yazılım mühendisliğinde kullanılan bir "mimari desen" dir. Bu desen Trygue Reenskaug tarafından ilk olarak tanımlanan bir desendir. Daha sonra Smalltalk üzerine yapılan araştırmalar Xerox araştırma laboratuarlarında devam etmiştir [7].

Birçok bilgisayar sisteminin amacı, bir veri deposundaki veriyi almak ve kullanıcıya göstermektir. Kullanıcı veri değişikliği yaptıktan sonra, sistem bu güncellemeleri veri deposunda saklar. Çünkü en önemli bilgi akışı, veri depolamak ve kullanıcı arayüzü arasındadır. Eğer kodlama miktarını azaltmak ve uygulama performansını artırmak

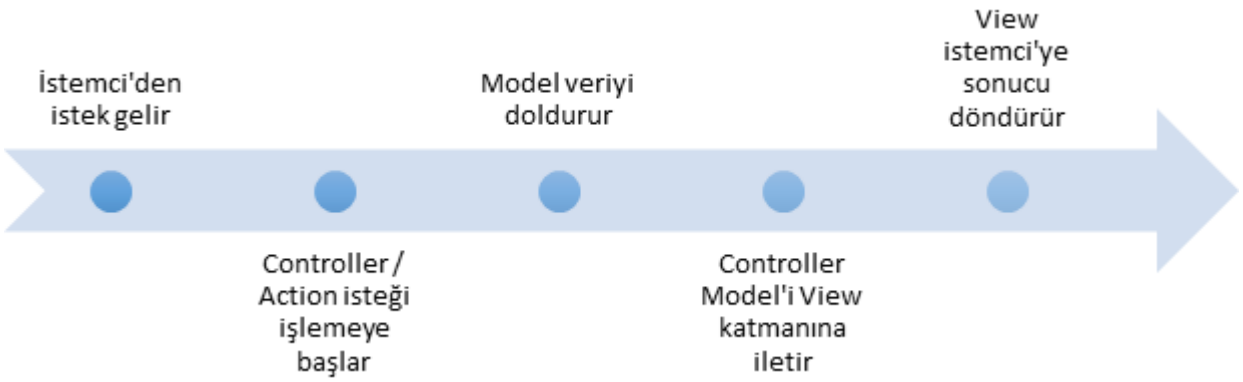
istiyorsanız bu ikisi arasındaki bağı iyi kurmalısınız. Bu işlemler Şekil 3.4'te gösterilmiştir, ayrıca bu mimarinin yaşam dögüsü Şekil 3.5'te gösterildiği gibidir.



Şekil 3.4. MVC Mimari Deseni

Ancak, görünüşte doğal olan bu yaklaşımda bazı önemli sorunlar var. Bir sorun, kullanıcı arabiriminin sıklıkla veri depolama sisteminde değişiklik yapma eğiliminde olmasıdır. Diğer bir sorun ise, iş uygulamalarının veri ve kullanıcı arayüzü arasındaki iş mantığını birleştirmek eğiliminde olmasıdır, bu veri iletimin ötesine geçmektedir.

Kullanıcı arayüzü mantığı özellikle web tabanlı uygulamalarda daha sık iş mantığını değiştirmek eğilimindedir. Bir örnekle bunu açıklayacak olursak, yeni bir kullanıcı arayüzüne sayfalar eklenebilir veya elde olan sayfaların düzenlemelerinde karışıklıklar olabilir. Sonuçta uygulamayı yeniden dağıtmaya gerek kalmadan, istediğiniz zaman kullanıcı arayüzünü değiştirek, Web tabanlı ince istemci uygulamasının (Web-based thin-client application) avantajlarından birisidir. Eğer sunum kodu ve iş mantığı tek bir nesnede birleştirilirse, kullanıcı arayüzü veriyi değiştirdiğinde, iş mantığı da her zaman içerdiği nesneyi değiştirmek zorundadır. Kullanıcı arayüzünün yaptığı en ufak bir değişiklikte bile hatalara bakmak ve bütün iş mantığını yeniden kontrol etmek gerekir [8].



Şekil 3.5. MVC Yaşam Döngüsü

3.4.1. Model

MVC dünyasında model uygulama verisinin veya durumunun saklandığı yerdir, genellikle veritabanı veya xml/json dosyası formatındadır. Model, veri katmanını (database, xml, json dosyası, vb.) uygulamadan izole eder, böylece diğer katmanlarda veri katmanının neresi olduğunun bilinmesine gerek kalmaz. Model katmanı sıklıkla Entity Framework, Nhibernate, LLBLGen, vb. gibi araçlar kullanılarak oluşturulur [9].

3.4.2. View

View, istemcinin gördüğü arayüzü içeren katmandır, genellikle Model katmanındaki verinin kullanılması ile oluşturulur. View katmanının Model ve Controller katmanlarından ayrılması ile arayüz değişikliklerinin uygulamanın diğer katmanlarını değiştirmeye gerek kalmadan yapılabilmesi sağlanmıştır. View katmanında HTML5 ve CSS3 gibi son versiyon teknolojiler kullanmak mümkündür. HTML5 ve CSS3 ile masaüstü ve mobil tarayıcılarda çalışabilen uygulamalar geliştirmek çok kolaylaşmıştır. Hatta Windows Store uygulamaları geliştirmek için HTML5 ve CSS3 teknolojilerinden yararlanılabilir [9].

3.4.3. Controller

Controller, istemciden gelen isteği işlemek, Model ve View katmanları arasında köprü olmak gibi görevleri yerine getirir. Controller içerisinde bir veya daha fazla Action olabilir, genellikle her Action bir web sayfası üretmek için kullanılır [9].

3.4.4. Routing

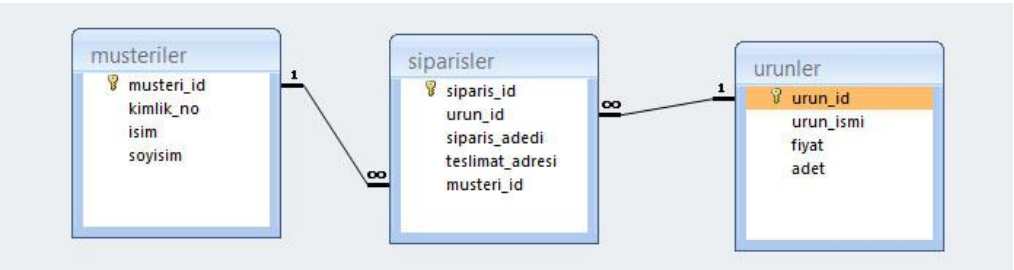
MVC'nin diğer bir önemli yapıtaşı Routing mekanizmasıdır. Routing, istemci'nin uygulamaya yaptığı isteği uygun Controller ve Action'a yönlendiren yapıdır. İstemci, isteği

uygulamanın belli bir adresine gönderir, routing mekanizması sayesinde ilgili adres için en uygun Controller ve içerisindeki Action tespit edilir ve çalıştırılır [9].

3.5. Veri Tabanı

Veri tabanları birbirleriyle ilişkili bilgilerin depolandığı alanlardır. Bilgi artışıyla birlikte bilgisayarda bilgi depolama ve bilgiye erişim konularında yeni yöntemlere ihtiyaç duyulmuştur. Veri tabanları; büyük miktardaki bilgileri depolamada geleneksel yöntem olan "dosya-işlem sistemine" alternatif olarak geliştirilmiştir. Telefonlarımızdaki kişi rehberi günlük hayatımızda çok basit bir şekilde kullandığımız veri tabanı örneği olarak kabul edilebilir. Bunların dışında internet sitelerindeki üyelik sistemleri, akademik dergilerin ve üniversitelerin tez yönetim sistemleri de veri tabanı kullanımına örnektir. Veri tabanları sayesinde bilgilere ulaşır ve onları düzenleyebiliriz. Veri tabanları genellikle bireysel olarak satın alınamayacak kadar yüksek meblağlara sahip olmasına karşın; ücretsiz kullanıma açılan akademik veri tabanları da bulunmaktadır. Akademik veri tabanları aracılığıyla bazen bibliyografik bilgi bazen de tam metinlere erişmek mümkündür. Veri tabanları, veri tabanı yönetim sistemleri aracılığıyla oluşturulur ve yönetilir. Bu sistemlere; Microsoft Access, MySQL, IBM DB2, Informix, Interbase, Microsoft SQL Server, PostgreSQL, Oracle ve Sysbase örnek olarak verilebilir.

Veri tabanında asıl önemli kavram, kayıt yığını ya da bilgi parçalarının tanımlanmasıdır. Bu tanıma şema adı verilir. Şema, veri tabanında kullanılacak bilgi tanımlarının nasıl modelleneceğini gösterir. Bu modele veri modeli (data model), yapılan işleme de veri modelleme denir. En yaygın olanı ilişkisel modeldir (relational model). Bu modelde veriler tablolarda saklanır. Tablolarda bulunan satırlar (row) kayıtların kendisini, sütunlar (column) ise bu kayıtları oluşturan bilgi parçalarının ne türden olduklarını belirtir. Şekil 3.6'da bu modelin örneği verilmiştir. Başka modeller (sistem modeli ya da ağ modeli gibi) daha belirgin ilişkiler kurarlar [10].



Şekil 3.6. İlişkisel Veri Tabanı Modelin Örneği

3.5.1. Microsoft SQL Server

Microsoft SQL Server (MSSQL), Microsoft tarafından geliştirilmiş bir veritabanı yönetim sistemidir. Bu sistem 24 Nisan 1989 tarihinde kullanılmaya başlanmıştır. Bu veritabanı yönetim sistemi ilişkisel veritabanı modelini kullanmaktadır. Genellikle diğer Microsoft yazılımları bu veritabanı yönetim sistemini otomatik olarak desteklemektedir. C#, .NET ve ASP MSSQL destekli Microsoft yazılımlarına örnek olarak verilebilir. Aynı zamanda Microsoft tarafından geliştirilmiş Azure sunucularda otomatik kurulu gelmektedir.

Microsoft, en az 12 farklı Microsoft SQL Server sürümünü piyasaya sürmüştür. Bu sürümler farklı izleyicileri, küçük tek makineli uygulamalardan çok sayıda eşzamanlı kullanıcıya sahip büyük İnternet'e dönük uygulamalara kadar değişen iş yüklerini hedeflemektedir.

3.6. LINQ

Türkçe karşılığı “Dil ile Bütünleştirilmiş Sorgu” olan LINQ (Language Integrated Query), .NET 3.5 ile yazılım hayatımıza girmiştir [11]. .NET nesnelerini sorgulamayı sağlayan bir mimaridir [12]. Linq teknolojisi, platform farklılıklarından kaynaklanan sorunların kaldırılması için geliştirilmiştir [13].

Bu teknoloji veri erişimini kolaylaştırmak, veri sorgulama yeteneklerine sahip bir araç olması sebebiyle geliştiriciler tarafından tercih edilmektedir [14].

3.6.1. Linq’de Kullanılan Sınıflar

LinQ teknolojisinde kullanılan sınıflar şu şekildedir:

- System.Query: Sorguların gerçekleştirilmesi için gerekli olan sınıfları içerir.
- System.Xml.Xlinq: Xml dokümanlarının sorgulanabilmesi için gerekli olan sınıfları içerir
- System.Linq : Linq e ait sınıf ve arabirimler bu sınıf altında bulunur.
- System.Data.Linq: Sql tablolarının sorgulanması için gerekli olan sınıfları içerir

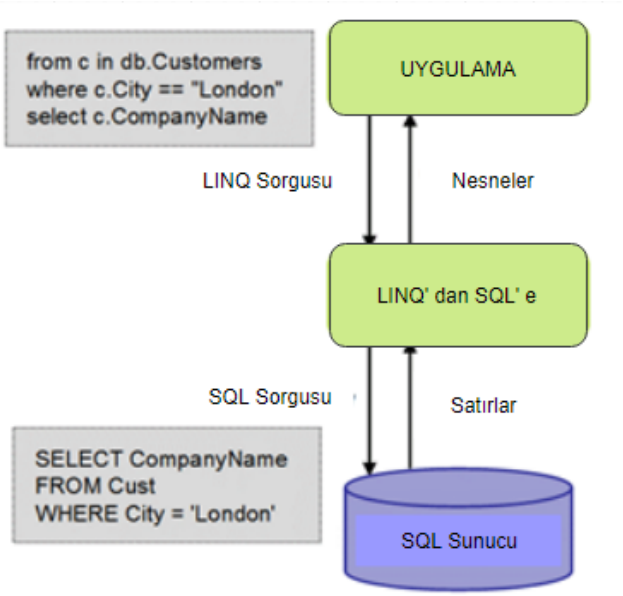
3.6.2. Linq Çeşitleri

Linq; linq veritabanları, ado.net, xml ve bellekte bulunan veriler için bize özel sağlayıcılar sunar. Bunlar şu şekilde sıralanabilir:

Linq to Object: Koleksiyonları sorgulama yarar. Fakat sadece Ienumerable<T> ara birimini destekleyen koleksiyonlar sorgulanır. System.Linq.Enumarable'a ait fonksiyonlar kullanılır [15].

Linq to XML (Xlinq): Xml belgelerini sorgulamak için kullanılır. Bunun için System.Xml.Linq kütüphanesi kullanılır [15].

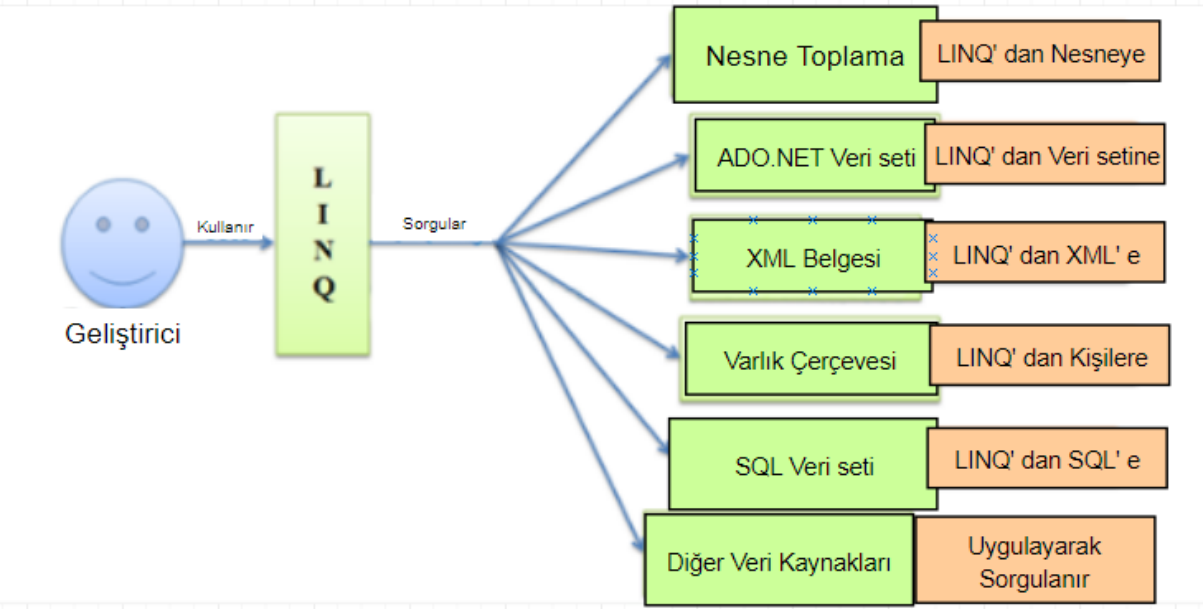
Linq to SQL (Dlinq): İlişkisel veri tabanlarının sorgulanmasını sağlar. Linq ile sql arasındaki iletişimi DBML(Database Markup Language-Veritabanı İşaretleme Dili) sağlar [15]. Bu sağlayıcının çalışma mantığı Şekil 3.7'de verilmiştir.



Şekil 3.7. LINQ to SQL çalışma yapısı

Linq to Dataset: Ado.Net'e ait Dataset nesnelerinin ilişkisel veritabanı gibi sorgulanmasını sağlar [15].

Şekil 3.8'de Linq çeşitleri ve dönüş tipleri gösterilmiştir.



Şekil 3.8. LINQ çeşitleri ve işlevleri

3.7. XPath

XPath, XSL Dönüşümleri (XSLT) ve XPointer (XPointer) arasında paylaşılan işlevsellik için ortak bir sözdizimi ve anlamsallık sağlama çabasının bir sonucudur. XPath'ın ana amacı, bir XML (XML) belgenin parçalarını birbirlerine adreslemektir. Bu birincil amacı desteklemek üzere, dizgeler, sayılar ve mantıksal ifadeler için temel oluşumlar sağlar. XPath, tanım-yeri (URI) başvurularında ve XML öznitelik değerlerinde XPath kullanımını kolaylaştırmak için XML-dışı, kısa ve özlü bir sözdizimi kullanır. XPath, bir XML belgenin yüzeysel sözdiziminden ziyade soyut, mantıksal yapısı üzerinde işlem yapar. XPath, bir XML belgenin orunsal yapısı boyunca gezinmek için URL'lerdeki gibi bir yol gösterimi kullanmasından dolayı bu adı almıştır.

Adresleme için kullanımına ek olarak, eşleştirme işlemlerinde (bir düğümün bir örüntüyle eşleşip eşleşmediğinin sınanması gibi) kullanılabilen doğal bir altkümeye sahip olacak şekilde tasarlanmıştır. XPath'ın bu kullanımı XSLTde açıklanmıştır.

XPath bir XML belgeyi düğümlerden oluşan bir ağaç olarak modeller. Eleman düğümleri, öznitelik düğümleri ve metin düğümleri gibi farklı düğüm türleri vardır. XPath her düğüm türü için bir dizgesel değeri hesaplayacak bir yöntem tanımlar. Bazı düğüm türlerinin ayrıca adları da mevcuttur. XPath, XML İsim-alanlarını (XML Adları) tamamen destekler. Bu

bakımdan, bir düğüm adı, olası bir isim-alanı tanım-yeri ile bir yerel parçadan oluşan bir çift olarak modellenir; buna genişletilmiş isim adı verilir. Veri modeli Veri Modeli bölümünde ayrıntılı olarak açıklanmıştır.

XPath'ta başat sözdizimsel oluşum ifadedir. Bir ifade, İfade sözdizimi tanımıyla eşleşir. Bir ifade, aşağıdaki dört temel türden biri olarak bir nesneyle sonuçlanmak üzere değerlendirilir:

- düğüm-kümesi (yinelenmemiş düğümlerden oluşan sırasız düğüm kolleksiyonu)
- mantıksal ifade (doğru veya yanlış)
- sayı (bir gerçel sayı)
- dizge (UCS karakterleri dizisi)

İfade, bağlamla ilişkili olarak değerlendirilir. XSLT ve XPointer, kendileri tarafından kullanılan XPath ifadeleri için bağlamın nasıl saptanacağını kendileri belirtirler. Bağlam şunlardan oluşur:

- bir düğüm (bağlamsal düğüm)
- sıfırdan farklı ve pozitif bir tamsayı çifti (bağlamsal sıra ve bağlamsal boyut)
- bir değişken bağıntıları kümesi
- bir işlev kütüphanesi
- ifadenin etki alanı içindeki isim-alanı bildirimleri kümesi

Bağlamsal konum daima bağlamsal boyuttan küçük veya ona eşittir.

Değişken bağıntıları, değişken isimleri değişken değerlerine eşlenerek oluşur. Bir değişkenin değeri bir nesne olup, bu nesne, bir ifade için olası türlerden biri olabileceği gibi burada belirtilmemiş ilave türlerden biri de olabilir.

İşlev kütüphanesi işlev isimleri işlevlere eşlenerek oluşur. Her işlev sıfır veya daha fazla argüman alır ve tek bir sonuçla döner. Bu belirtim, tüm XPath gerçeklenimlerinin desteklemesi gereken temel bir işlev kütüphanesi tanımlar (bkz, Temel İşlev Kütüphanesi). Temel işlev kütüphanesindeki bir işlev için argümanlar ve sonuçlar dört temel türde olabilir. XSLT ve XPointer, ek işlevler tanımlayarak XPath'ı genişletir; bu işlevlerin bazıları dört temel türde işlem yapar; bazıları da XSLT ve XPointer tarafından tanımlanmış veri türleri üzerinde işlem yapar.

İsim-alanı bildirimleri, önekler isim-alanı tanım-yerlerine eşlenerek oluşur.

Bir alt ifadeyi değerlendirmekte kullanılan değişken bağıntıları, işlev kütüphanesi ve isim-alanı bildirimleri, daima bu ifadeyi içeren ifadeyi değerlendirmekte kullanılanlarla aynıdır. Bir alt ifadeyi değerlendirmekte kullanılan bağlamsal düğüm, bağlamsal konum ve bağlamsal boyut bazan bu ifadeyi içeren ifadeyi değerlendirmekte kullanılanlardan farklıdır. Bazı ifade çeşitleri bağlamsal düğümü değiştirir; bağlamsal konumu ve bağlamsal boyutu sadece dayanaklar değiştirir. Bir ifade çeşidinin değerlendirimi açıklanırken, daima doğrudan doğruya, alt ifadelerin değerlendirimi için bağlamsal düğüm, bağlamsal konum ve bağlamsal boyutun değişmesine göre durumlanır; eğer bağlamsal düğüm, bağlamsal konum ve bağlamsal boyut hakkında hiçbir şey söylenemiyorsa, ifade çeşidinin alt ifadelerinin değerlendirimi için bunlar değişmeden kalır.

XPath ifadeleri çoğunlukla XML özniteliklerinde görülür. Bu bölümde belirtilen dilbilgisi öznitelik değerine XML normalleştirmesinden sonra uygulanır. Örneğin, dilbilgisi < karakterini kullanıyorsa, bunun XML belgede < olarak gözükmemesi, XML 1.0 kurallarına göre öncelenmesi gerekir (< olarak verilmesi gerekir). İfadelerin içinde, dizgesel sabitler, öznitelik değerleri için yapıldığı gibi, tek veya çift tırnak içine alınırlar. XML işlemcinin ifadedeki tırnak karakterlerini öznitelik değerini sonlandıran tırnak karakteri olarak algılamaması için ifade içindeki tırnak karakterleri karakter gönderimleri olarak girilmelidir (“ veya ‘ gibi). Başka bir yöntem de, XML özniteliğin değeri çift tırnak içine alınmışsa, ifade içinde tek tırnak kullanmak veya değer tek tırnak içine alınmışsa, ifade içinde çift tırnak kullanmaktır.

Önemli ifade çeşitlerinden biri konumsal yoldur. Bir konumsal yol bağlamsal düğüme göreli olarak bir düğüm kümesini seçer. İfadenin konumsal yol olduğu durumda değerlendirme sonucu, konumsal yol tarafından seçilen düğümleri içeren düğüm kümesidir. Konumsal yollar düğüm kümelerini süzmekte kullanılan ifadeleri dönüşümlü olarak içerebilir. Konumsal yollar KonumsalYol sözdizimi tanımıyla eşleşirler.

Sözdizimi için kullanılan gösterim (XML)'de kullanılan BNF gösterimi ile aynıdır (tanım terimlerinin baş harflerinde büyük harf kullanılması dışında).

İfadeler çözümlenirken, önce karakter dizgelerine bölünür, bunlar dizgecikler halinde çözümlendikten sonra da dizgecikler çözümlenir. Dizgecikler arasında boşluk karakterleri özgürce kullanılabilir. Dizgeciklere ayırma işlemi Sözdizimsel Yapı bölümünde açıklanmıştır.

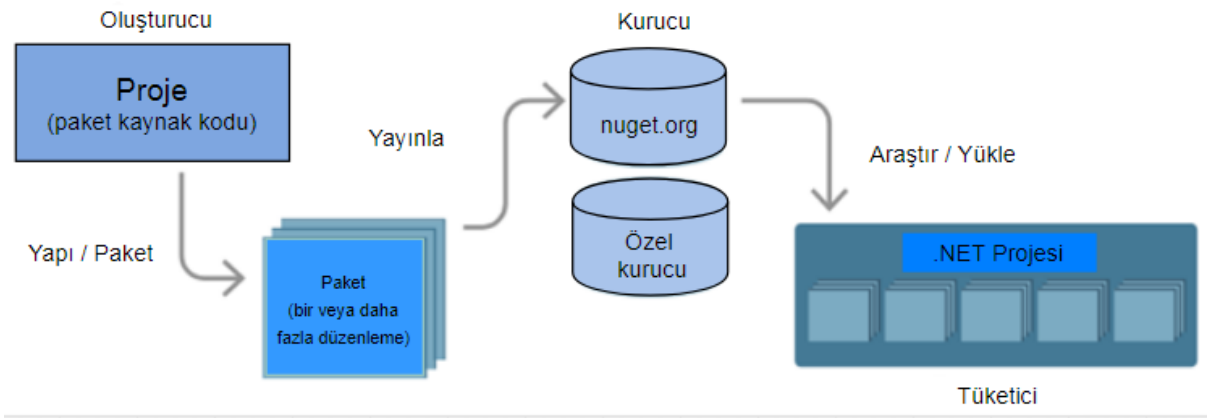
Ek olarak, XML 1.0 ve XML Adları 1.0 gönderimleri daima kolaylık olsun diyedir. Bununla birlikte, bir gerçeklenim (XML) ve (XML Adları) veya (XML 1.1) ve (XML Adları 1.1) sözdizimsel belirtimlerini desteklemeyi tercih edebilir. Bu bakımdan, uluslararası tanım-yeri (IRI) ayrıca destekleniyor olsa da tanım-yeri (URI) başvuruları ayrıca kullanılmıştır. Bazı durumlarda XML 1.0 ve XML 1.1 tanımları tamamen aynı olabilir [16].

3.8. NuGet

Herhangi bir modern geliştirme platformu için vazgeçilmez bir araç, geliştiricilerin yararlı kod kitaplıklarını yaratması, paylaşması ve tüketmesi için kullanılan bir mekanizmadır. Bu tür kütüphaneler, genellikle "paketler" olarak adlandırılır, çünkü derlenen kodları (DLL'ler gibi) bu kütüphaneleri tüketen projelerde ihtiyaç duyulabilecek diğer içerikler içerebilirler.

.NET için kod paylaşma mekanizması, .NET paketlerinin nasıl oluşturulduğu, barındırıldığı ve tüketildiğini tanımlayan NuGet'tir ve bu rollerin her biri için araçlar sağlar. Bu sistemin çalışma mantığı Şekil 3.9'da gösterilmektedir.

Basitçe ifade etmek gerekirse, bir NuGet paketi, derlenmiş kodu (DLL) içeren diğer .nupkg uzantılı tek bir ZIP dosyası ve bu kodla ilgili diğer dosyalar ve paketin sürüm numarası gibi bilgileri içeren açıklayıcı bir manifestir. Kütüphane geliştiricileri paket dosyaları oluşturur ve bir ana makineye yayınlar. Paket tüketicileri bu paketleri alır, bunları projelerine ekler ve sonra bu kitaplığın işlevselliğini kendi proje kodunda arar. NuGet kendisi tüm ara ayrıntıları işler [17].



Şekil 3.9. NuGet'in çalışma şekli

3.9. HTML Agility Pack

HTML Agility Pack (HAP) C# için HTML ayrıştırıcı paketidir. Bu paket sayesinde bir HTML kod üzerinde gezinme, değiştirme gibi işlemler daha kolay bir şekilde yapılmaktadır. HAP aldığı HTML kaynak kodunu düğümler (nodes) haline getirip daha sonra bunlar arasındaki bağlantıları oluşturarak bir düğümden diğer düğüme geçme işlemini daha kolay bir hale getirmiş olacaktır. Burdaki her düğüm ise HTML kodundaki etiketler (tags), etiketlerin birer parametreleri (attributes) olabilir. HAP ile HTML içinde düğüm sorgulama yaparken, bu sorgulamayı XPath, düzenli ifade (Regular Expression) gibi teknolojiler kullanılarak yapılmaktadır [18].

Bu paketi projenize eklemek için, Visual Studio'da paket yönetim konsolunda (package manager console) Install-Package HtmlAgilityPack yazarak paketin son sürümünü elde etmiş olacaksınız.

3.10. Veri Madenciliği

Basit bir tanım yapmak gerekirse veri madenciliği, büyük ölçekli veriler arasından bilgiye ulaşma, bilgiyi madenleme işidir. Ya da bir anlamda büyük veri yığınları içerisinden gelecekle ilgili tahminde bulunabilmemizi sağlayabilecek bağıntıların bilgisayar programı kullanarak aranmasıdır. Veri madenciliği deyimi yanlış kullanılan bir deyim olabileceğinden buna eş değer başka kullanımlar da literatüre geçmiştir. Veritabanlarında bilgi madenciliği (İng. knowledge mining from databases), bilgi çıkarımı (İng. knowledge extraction), veri ve örüntü analizi (İng. data/pattern analysis), veri arkeolojisi gibi.

Bunların arasındaki en yaygın kullanım Veritabanlarında Bilgi Keşfi (İng. VBK - Knowledge Discovery From Databases - KDD)'dir. Alternatif olarak veri madenciliği aslında bilgi keşfi sürecinin bir parçası şeklinde kabul görmektedir. Bu adımlar:

1) Veri temizleme (gürültülü ve tutarsız verileri çıkarmak)
2) Veri bütünleştirme (birçok veri kaynağını birleştirebilmek)
3) Veri seçme (yapılacak olan analizle ilgili olan verileri belirlemek)
4) Veri dönüşümü (verinin veri madenciliği tekniğinden kullanılabilecek hale dönüşümünü gerçekleştirmek)
5) Veri madenciliği (veri örüntülerini yakalayabilmek için akıllı metotları uygulamak)
6) Örüntü değerlendirme (bâzı ölçümlere göre elde edilmiş bilgiyi temsil eden ilginç örüntüleri tanımlamak)

7) Bilgi sunumu (mâdenciliği yapılmış olan elde edilmiş bilginin kullanıcıya sunumunu gerçekleştirmek).

Veri madenciliği adımı, kullanıcı ve bilgi tabanıyla etkileşim halindedir. İlginç örüntüler kullanıcıya gösterilir, ve bunun ötesinde istenirse bilgi tabanına da kaydedilebilir. Buna göre, veri madenciliği işlemi, gizli kalmış örüntüler bulunana kadar devam eder.

Bir veri madenciliği sistemi, aşağıdaki temel bileşenlere sahiptir:

1) Veritabanı, veri ambarı ve diğer depolama teknikleri
2) Veritabanı ya da Veri Ambarı Sunucusu
3) Bilgi Tabanı
4) Veri Madenciliği Motoru
5) Örüntü Değerlendirme
6) Kullanıcı Arayüzü

Veri madenciliği, eldeki verilerden üstü kapalı, çok net olmayan, önceden bilinmeyen ancak potansiyel olarak kullanışlı bilginin çıkarılmasıdır. Bu da; kümeleme, veri özetleme, değişikliklerin analizi, sapmaların tespiti gibi belirli sayıda teknik yaklaşımları içerir.

Başka bir deyişle, veri madenciliği, verilerin içerisindeki desenlerin, ilişkilerin, değişimlerin, düzensizliklerin, kuralların ve istatistiksel olarak önemli olan yapıların yarı otomatik olarak keşfedilmesidir.

Temel olarak veri madenciliği, veri setleri arasındaki desenlerin ya da düzenin, verinin analizi ve yazılım tekniklerinin kullanılmasıyla ilgilidir. Veriler arasındaki ilişkiyi, kuralları ve özellikleri belirlemekten bilgisayar sorumludur. Amaç, daha önceden fark edilmemiş veri desenlerini tespit edebilmektir.

Veri madenciliğini istatistiksel bir yöntemler serisi olarak görmek mümkün olabilir. Ancak veri madenciliği, geleneksel istatistikten birkaç yönde farklılık gösterir. Veri madenciliğinde amaç, kolaylıkla mantıksal kurallara ya da görsel sunumlara çevrilebilecek nitel modellerin çıkarılmasıdır. Bu bağlamda, veri madenciliği insan merkezlidir ve bazen insan – bilgisayar arayüzü birleştirilir.

Veri madenciliği sahası, istatistik, makine bilgisi, veritabanları ve yüksek performanslı işlem gibi temelleri de içerir.

Veri madenciliği konusunda bahsi geçen geniş verideki geniş kelimesi, tek bir iş istasyonunun belleğine sığamayacak kadar büyük veri kümelerini ifade etmektedir. Yüksek hacimli veri ise, tek bir iş istasyonundaki ya da bir grup iş istasyonundaki disklere sığamayacak kadar fazla veri anlamındadır. Dağıtık veri ise, farklı coğrafi konumlarda bulunan verileri anlatır [19].

3.10.1. Metin Madenciliği

Metin madenciliği çalışmaları metni veri kaynağı olarak kabul eden veri madenciliği (data mining) çalışmasıdır. Diğer bir tanımla metin üzerinden yapısallaştırılmış veri elde etmeyi amaçlar. Metin madenciliği, metinlerin sınıflandırılması, bölütlenmesi (clustering), metinlerden konu çıkarılması (concept/entity extraction), metinler için sınıf taneciklerinin üretilmesi (production of granular taxonomy), metinlerde görüş analizi yapılması (sentimental analysis), metin özetlerinin çıkarılması (document summarization) ve metinin özü ile ilgili ilişki modellemesi (entity relationship modelling) gibi çalışmaları hedefler.

Yukarıdaki hedeflere ulaşılması için metin madenciliği çalışmaları kapsamında enformasyon getirimi (information retrieval), hece analizi (lexical analysis), kelime frekans dağılımı (Word requency distribution), örüntü tanıma (pattern recognition), etiketleme (tagging), enformasyon çıkarımı (information extraction), veri madenciliği (data mining) ve görselleştirme (visualization) gibi yöntemleri kullanmaktadır.

Metin madenciliği çalışmaları, metin kaynaklı literatürdeki diğer bir çalışma alanı olan doğal dil işleme (natural language processing, NLP) çalışmaları ile çoğu zaman beraber yürütülmektedir. Doğal dil işleme çalışmaları daha çok yapay zeka altındaki dil bilim bilgisine dayalı çalışmaları kapsamaktadır. Metin madenciliği çalışmaları ise daha çok istatistiksel olarak metin üzerinden sonuçlara ulaşmayı hedefler. Metin madenciliği çalışmaları sırasında çoğu zaman doğal dil işleme kullanılarak özellik çıkarımı da yapılmaktadır [20].

3.11. RAKE

Rapid Automatic Keyword Extraction (RAKE) Türkçesi ise Hızlı Otomatik Anahtar Kelime Çıkarma algoritması, kelime kullanım sıklığı ve metindeki diğer kelimelerle olan ortak oluşunu analiz ederek, metin gövdesindeki anahtar cümleleri belirlemeye çalışan bir alan bağımsız anahtar kelime çıkarma algoritmasıdır.

3.11.1. Anahtar Kelime

Anahtar kelime, bir yazı metni içerisindeki konuyu en net şekilde ifade edebilen bir ya da birden fazla kelimeye verilen addır. Öneminin daha iyi anlaşılması için metnin odak noktası ya da kalbidir denilebilir.

İnternet kullanıcıları Google, Yandex, Yahoo vb. arama motorlarında aradıkları konu ile ilgili içeriklere ulaşabilmek için en çok bilinen birkaç kelimeyi (anahtar kelime) kullanarak aramalarını gerçekleştirmektedir. Bu sayede arama motorlarında içerik ile ilgili arama yapan birçok kişiye ulaşılması mümkün olmaktadır.

3.11.2. RAKE Algoritmasının Çalışması

Yukarıda açıklandığı üzere RAKE algoritması bir metinden anahtar kelime veya ifade çıkarılmasına olanak sağlamaktadır. Peki bu algoritma bu işlemi nasıl yapmaktadır?

İlk olarak RAKE algoritması işlenmesi gereken metni cümlelere ayırıp aday ifadeler üretmektedir. Burada, çeşitli noktalama işaretleri cümle sınırları olarak ele alınacaktır. Bu işlem çoğu durumlar için geçerlidir. Ancak bu sınırlar ifadenin bir parçası olduğu durumlar (örneğin .Net ve Dr. Who) için geçerli olmamaktadır. Engellenecek kelimeler (stopwords) dosyasında listelenen tüm kelimeler, ifade sınırları olarak ele alınmaktadır. Bu, bu metinde 'uyumluluk', 'sistemler', 'doğrusal kısıtlar', 'doğal sayılar' ve 'ölçütler' gibi bir veya daha fazla engellenmeyecek (non-stopwords) kelimeden oluşan adayların oluşturulmasına yardımcı olur. Adayların çoğu geçerli cümleler olacak ancak engellenecek kelime ifadenin bir parçası olduğu durumlarda çalışmamaktadır. Örneğin, 'new' RAKE'in engellenecek listesinde bulunan kelimelerden biridir. Bundan dolayı ne ‘New York’ ne de ‘New Zealand’ kelimeleri anahtar kelime olarak kabul edilmemektedir.

Daha sonra RAKE her adayın özelliklerini hesaplamaktadır. Bu da her kelimenin puan toplamına eşit olmaktadır. Kelimeler kullanım sıklığına ve bulundukları aday ifadenin tipik uzunluğuna göre puanlandırılmaktadır. Burada dikkate alınması gereken husus, adayların herhangi bir şekilde normalleştirilmemesidir. Bundan dolayı neredeyse aynı görünen anahtar kelimelere sahip olabiliriz. Buna örnek olarak ‘küçük ölçekli üretim’ ve ‘küçük ölçekli üreticiler’ veya ‘yağsız süt tozu’ ve ‘yağsızlaştırılmış süt tozu’ ifadeleri verilebilir. İdeal olarak, bir anahtar kelime çıkarma algoritması, anahtar kelimeleri normalleştirmenin köküne inme yolunu ve diğer yollarını uygulamalıdır.

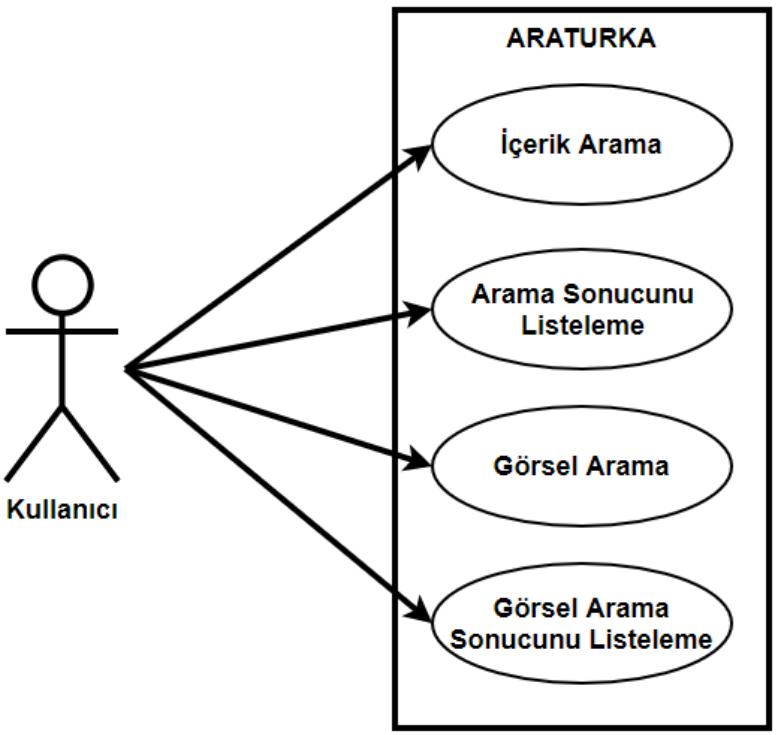
Son olarak anahtar kelime adayları RAKE puanlarına dayanarak sıralanacaktır. Sonra sıralanmış bu listeden ister ilk 5 ister belli bir puan üzerinde olanları anahtar kelime olarak seçilebilmektedir [21].

4. DENEYSEL ÇALIŞMALAR

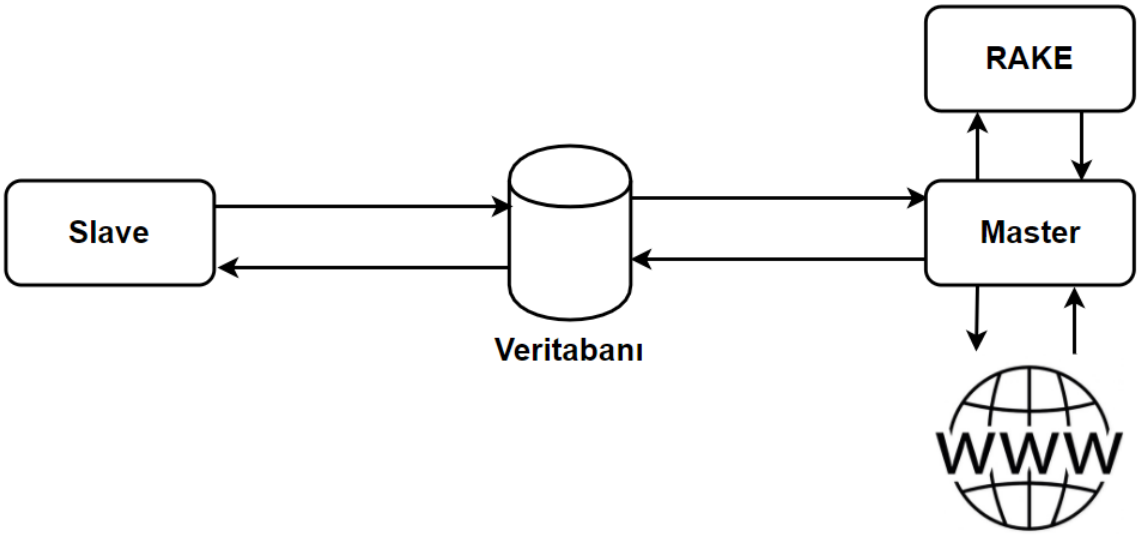
"AraTurka" uygulamasının geliştirilmesi birçok adımdan oluşmaktadır. Bu adımlar aşağıda adım adım açık bir şekilde anlatılmıştır;

4.1. Planlama

Başlangıçta araştırmalarla birlikte aynı zamanda uygulamanın hangi platformda ve nasıl bir şekilde çalışacağı, use case diyagramı (Şekil 4.1) planlanmıştır. İlk olarak sistemin iki alt uygulamadan (Master ve Slave) oluşturulmasına karar verilmiştir. Bunların aralarındaki ilişki tasarlanmıştır. Tasarlanan bu ilişki Şekil 4.2'de gösterildiği gibi olmaktadır. Sonra hangi yazılım dilini kullanılacağı karar verilmiştir. Verilen kararda Master uygulaması için C#, Slave uygulaması için ise ASP.NET teknolojileri seçilmiştir. Bunun nedeni aynı zamanda LINQ, HAP ve diğer paketlerin bu teknolojilerde daha kolay bulunabileceğidir. Bunun dışında başlangıçta Master sisteminin bir WinForms veya WPF tabanlı bir uygulama olmasını düşünülüyordu ama bu sonra da değişmiştir ve Master uygulamasını konsol tabanlı uygulama şeklinde yapılacağını karar vermiştir.



Şekil 4.1. Use Case Diyagramı



Şekil 4.2. Master Slave ilişkisi

4.2. Sunucuya İstek Gönderme

Planlamalar bittikten sonra uygulamanın artık bir websitesine istek göndermesini sağlamak gerekmekteydi. Bu işlemi gerçekleştirmek için C#'ın System.Net kütüphanesindeki WebRequest kullanılmıştır. WebRequest önceden belli olan bir URL'e istek gönderir. Bu durumda ilk olarak klavyeden bir url girilecektir. Girilen url WebRequest'e verilecektir ve WebRequest isteği oluşturup gönderecektir.

4.3. Sunucudan Gelen Yanıtı Denetleme

İsteği gönderdikten sonra, elde ettiğimiz sonuçları nelerden oluşuyor diye gelen yanıtın denetlenmesi gerekmektedir. Yanıt paketi, paket başlığı ve paket verisi olarak iki bölümden oluşmaktadır.

4.4. Yanıt Paketin Başlıklarını Okuma

Paket başlığı kısmında yanıtın durum kodunu kontrol ederek isteğin sunucuya ulaşıp ulaşamadığı, sunucu yanıt verip vermediği gibi bilgiler elde edinebilmektedir. Eğer durum kodu 200 yani OK ise paketin içerik tipi kontrol edilecektir. Bu kontrol sonucu eğer text/html metinini içeriyorsa o zaman sunucu yanıt olarak bir web sayfası göndermiştir. Eğer kontrol sonucu image metnini içeriyorsa o zaman sunucu bir resim döndürmüştür. Bu kontrolun sonucu sadece bunlar olmamaktadır ancak bu uygulamayı ilgilendiren bunlardır.

4.5. Yanıtın Kaynak Kodunu Okuma

Paketin başlıklarını okuduktan sonra sıra paketin verisine gelecektir. Paketin içeriği html ise tüm bu html kodları bir string'e alınacaktır. Başlangıç için sistemin doğru çalışıp çalışmadığını görmek için bu html kodları doğrudan konsola yazdırılır.

4.6. Veritabanını Oluşturma

İstek gönderip yanıtını alabildikten sonra bu yanıtları kaydetmek için bir veritabanı oluşturulmuştur.

Veritabanında WebSiteler tablosunda her erişilen site kaydedilmektedir.

Veritabanındaki diğer tablo ise WebSayfalar tablosudur. WebSayfalar tablosunda ise taranması gereken ve taranılan tüm sayfalar kaydedilmiştir. Bu tabloda bir web sayfasıyla ilgili sayfa yolu (path), sorgusu (query), uzantısı (uzanti), başlığı (baslik), açıklaması (aciklama), içindeki metinler (icerik), istek sonucundan dönen yanıtın içerik tipi (content_type), bu yanıtın durum kodu (response_code) ve sayfanın taranıp taranmadığına dair bilgi (scanned), ayrıca her sayfayla ilgili website_id bilgisi tutulacaktır. Bu website_id sayesinde linkin ait olduğu siteye kolayca erişilebilir.

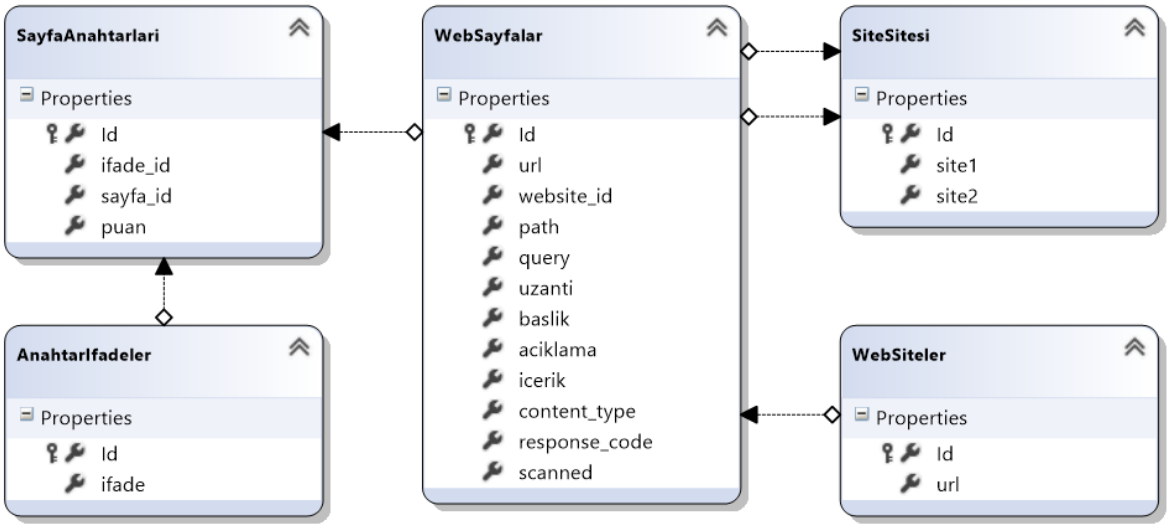
AnahtarIfadeler tablosunda ise bulunan tüm anahtar kelimeler kaydedilmektedir.

SayfaAnahtarlari tablosu da hangi sayfanın hangi anahtar kelimelere sahip olduğu bilgisinin kaydını tutmaktadır, ayrıca bu RAKE algoritmasının bu kelimeye vermiş olduğu puanını da

kaydedilmiştir. Kısacası AnahtarIfadeler ve WebSayfalar tabloları arasındaki bağlantıyı oluşturmaktadır.

SiteSitesi tablosu da yeni bulunan sayfaların hangi sayfada bulunduklarını kaydedilmektedir. Bu tabloda yeni bulunan sayfa site2 ve bulunduğu sayfa ise site1 olarak kaydedilmektedir.

Bu tabloların bağlantıları ve içerikleri Şekil 4.3'teki şemada gösterilmiştir.



Şekil 4.3. Veritabanı ER diyagramı

4.7. Kaynak Koddan Meta Etiketler Çıkarma

Kaynak koddan meta etiketlerini çıkarma işlemi ise HAP kullanmadan önce metin işlemleri ile yapılmaktaydı. Bu işlem önce meta etiketleri bulup ("<meta" ve ">" arasındakiler) sonra bunların arasındaki anahtar kelime (keyword) ve hakkında (description) olanlarını seçip en son onların içeriğini (content) okuyup kaydederek gerçekleştirmektedir.

HAP paketini kullandıktan sonra bu işlem daha da kolaylaşmıştır. HAP ile meta etiketleri seçmek "//meta" XPath'ini kullanarak gerçekleştirmektedir. Özellikle keyword veya description meta etiketini çıkarmak ise "//meta[@name='keyword']" ve "//meta[@name='description']" ile yapılmaktadır.

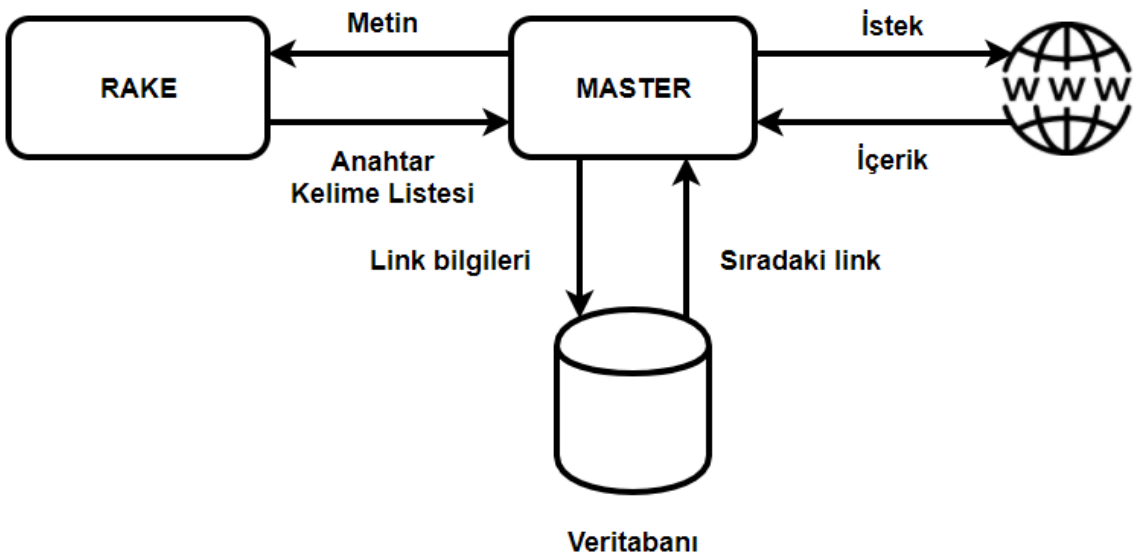
Bu iki etiketin içeriği kelime kelime ayrılıp sonra her kelimesi bir anahtar kelime olarak veritabanına kaydedilmektedir. Açıklama meta etiketi aynı zamanda linkin açıklaması olarak da kaydedilmektedir.

4.8. Kaynak Koddan İçerik Çıkarma

Kaynak kodlardan içerik çıkarma ise HAP ile yapılmıştır. Bu işlemin XPath'i ise "//body//text()[not(ancestor::style)][not(ancestor::script)][not(ancestor::button)][not(ancestor::select)][not(ancestor::ul)][not(ancestor::ol)]" olmaktadır. Burada bu XPath sayesinde tüm style, script, button, select, ul ve ol içinde olmayan metinler seçilecektir. Seçildikten sonra bu metinler birleştirilip veritabanına kaydedilir. Aynı zamanda sonraki başlık altında anlatılacak olan anahtar kelime çıkarma algoritmasını kullanılarak bu metinden anahtar kelimeler çıkarılacaktır ve veritabanına yazılacaktır. Bu işlem ise sonraki başlık altında anlatılmıştır.

4.9. Metinden Anahtar Kelime Çıkarma

Metinden anahtar kelime çıkarma ise RAKE algoritmasına göre yapılmaktadır. RAKE algoritmasının C# dilinde uygulaması kullanılarak websitesinin içeriğinden anahtar kelime listesi oluşturulmaktadır. Sonra oluşturulmuş olan bu listeden ilk 20'si anahtar kelime olarak veritabanına kaydedilecektir. Bu işlem Şekil 4.4'te gösterilmiştir.

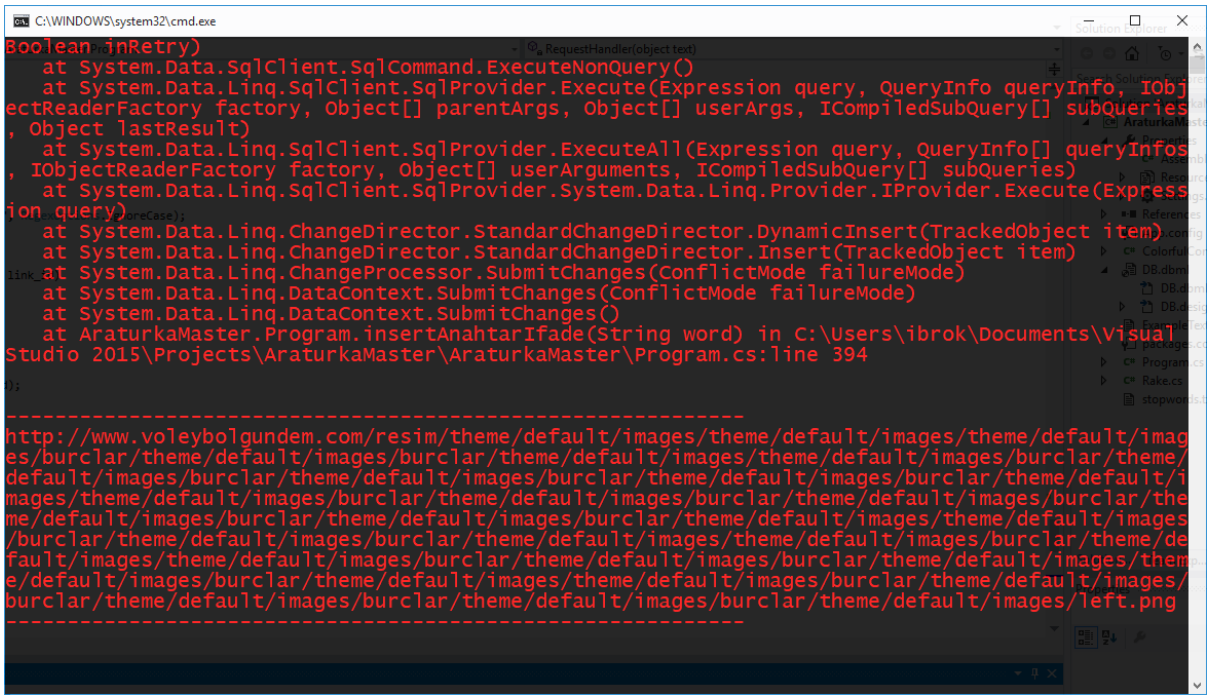


Şekil 4.4. Master'in Çalışma Mantığı

4.10. Hata Yakalama Sistemini Oluşturma

Uygulamanın çalışma esnasında birçok hata ile karşılaştırılmıştır (Şekil 4.5). Bu nedenle bu hataları daha iyi anlamak için bir hata yakalama sistemi oluşturulmuştur. Bu sistem

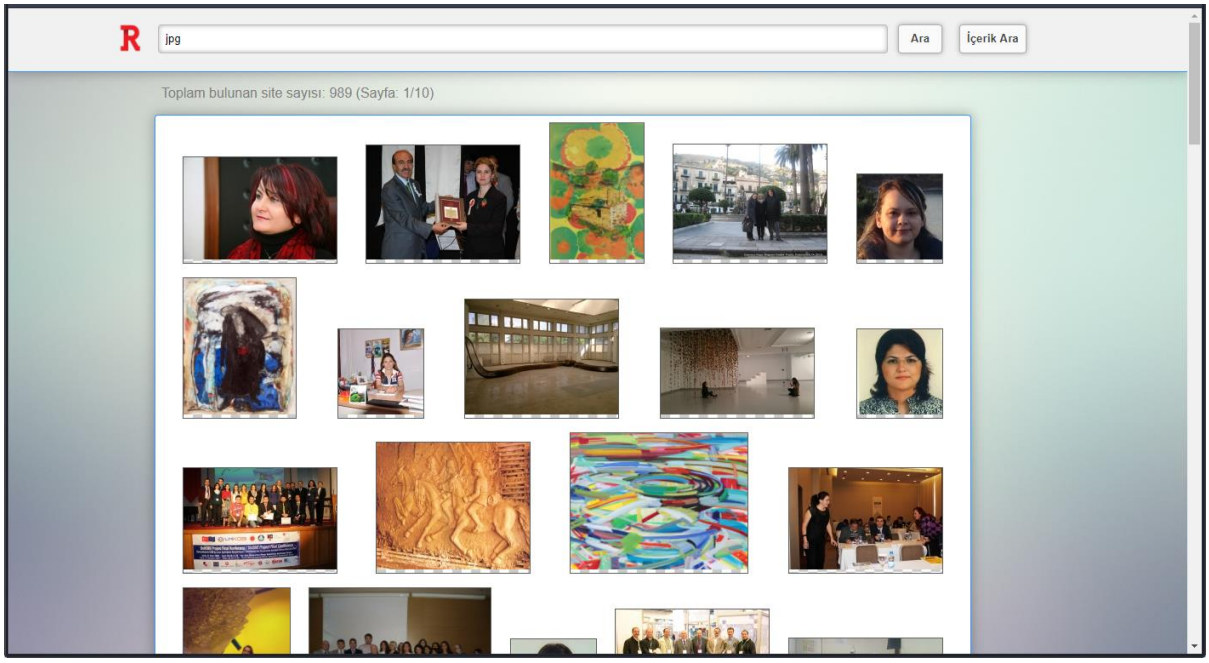
yakaladığı her hata için sırada olan linki, hata mesajını ve satırını göstermektedir. Sistemde normal olarak hata verildiğinde uygulamayı durdurmadan sonraki linkle devam edilmesi gerçekleştirilmiştir, ayrıca bu ayar programın başında kullanıcı tarafından değiştirilebilecektir.



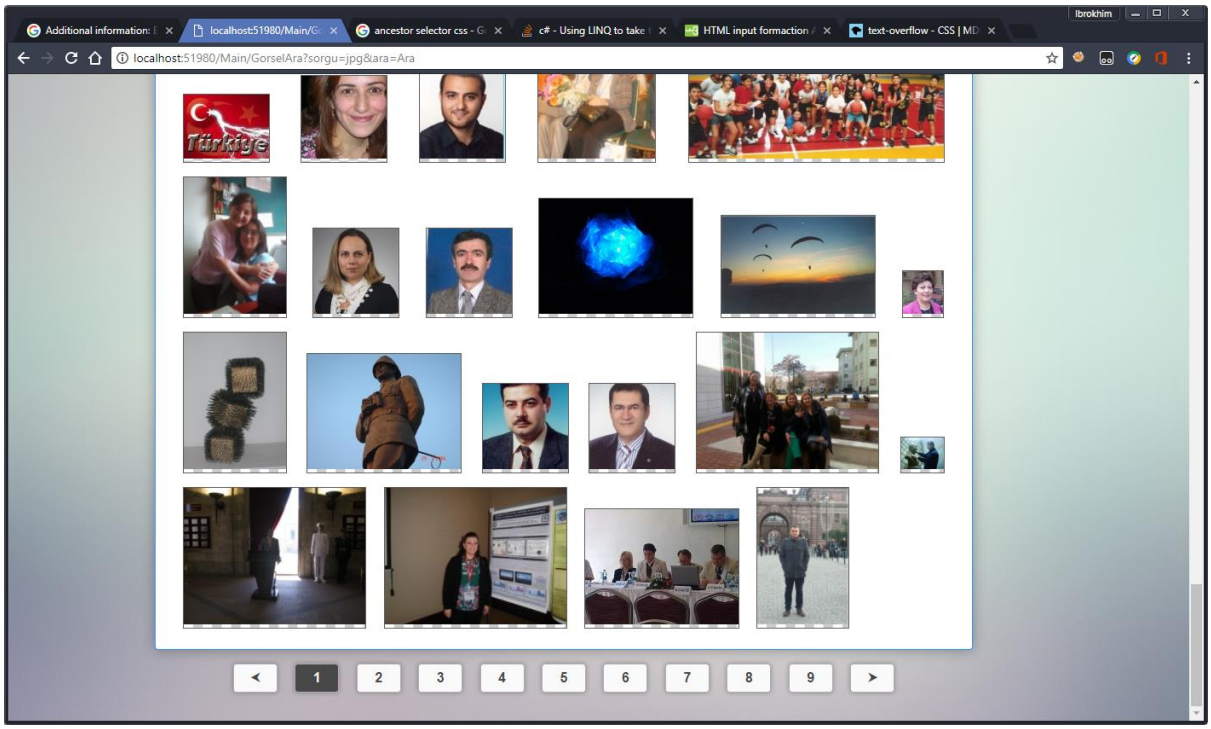
Şekil 4.5. AraTurka - Hata yakalama

4.11. Web Sitenin Çalışma Mantığı

Uygulamanın Slave sistemi bir MVC uygulamasıdır. Bu sistemde C#, ASP.NET ve LINQ teknolojileri kullanılmıştır. Aynı zamanda bir web uygulaması olduğu için html, css ve javascript gibi işaret ve betik dilleri de kullanılmıştır. Sistem kullanıcı dostu bir tasarımla kullanıcıya sunulmuştur. Kullanıcı aramak istediğini klavyeden giriş yaptıktan sonra enter veya ara tuşuna tıklayarak bu girişi controller'e gönderecektir. Controller bu girişi kelimelere ayırıp bu kelimelerin en çok bulunduğu siteleri listelenecektir (Şekil 4.8). Listede bulunan her sitenin linki, başlığı, eğer varsa açıklaması yoksa içeriğinin ilk 320 karakteri kullanıcıya sunulacaktır. Slave sistemin çalışma mantığı Şekil 4.9'da gösterilmiştir. Kullanıcı bunların arasında istediğini seçerek o siteye yönlendirilecektir. Aynı zamanda ana sayfadan kullanıcı görsel arama butonunu seçerek görselleri başlıklarına göre (img etiketin alt özelliği) veya linkine göre arayabilmektedir.

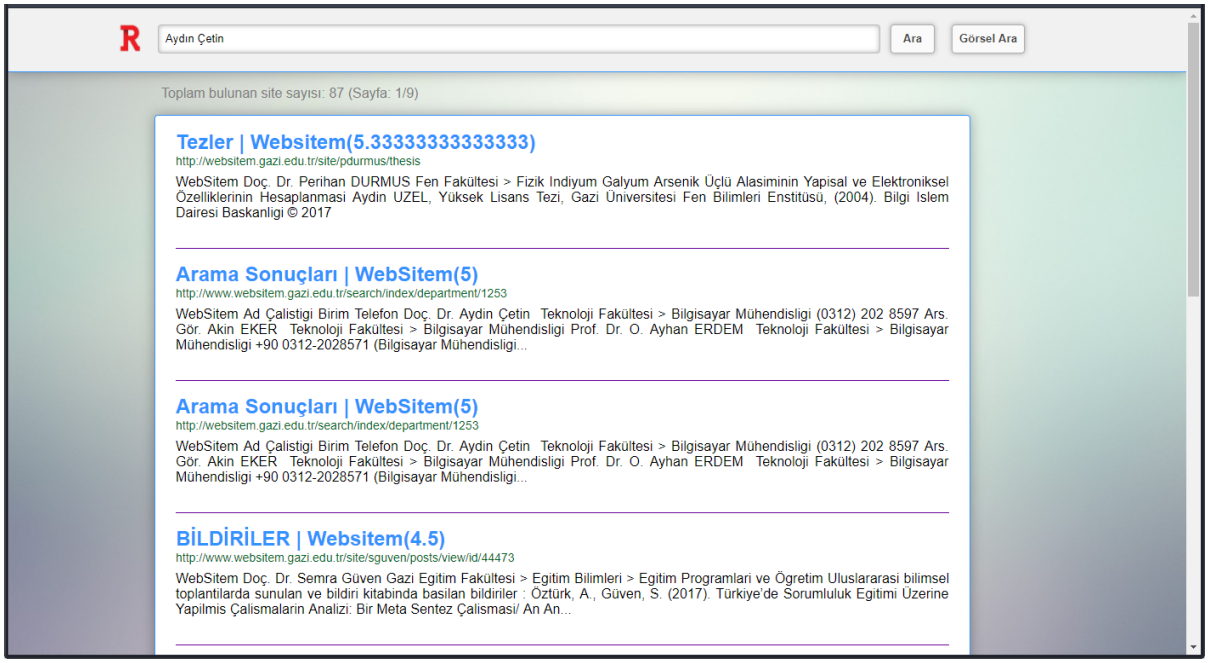


Şekil 4.6. AraTurka - Görsel Arama

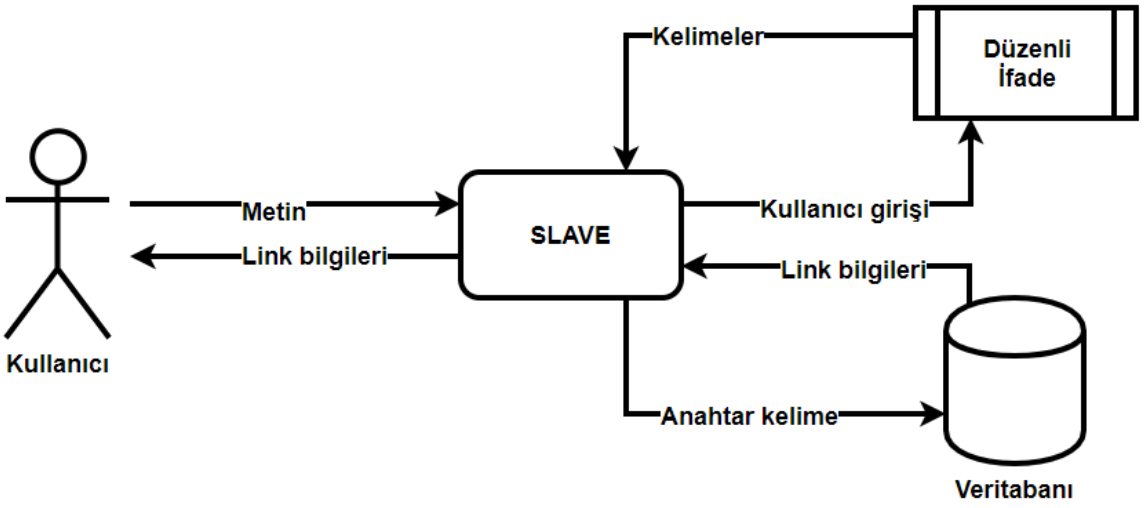


Şekil 4.7. AraTurka - Sonuc sayfalama

Aradıktan sonra kullanıcıya bulunan fotoğraflar listelenmektedir (Şekil 4.6). Eğer sonuçlar fazla miktarda ise her sayfada 100 görsel gösterilecek şekilde sayfalara bölünmektedir (Şekil 4.7). Listedeki herhangi bir fotoğraf seçildiğinde o fotoğrafın başlığı ve linki kullanıcıya göstermektedir.



Şekil 4.8. AraTurka - İçerik arama



Şekil 4.9. Slave'in kullanıcı diyagramı

5. SONUÇ VE ÖNERİ

Internet Live Stats tarafından yayınlanan dünyada internet kullanımı istatistiklerine göre yaklaşık 3,5 milyar internet kullanıcısı ve 1,15 milyardan fazla web sitesi mevcuttur. Bu sitelere ait 50 trilyondan fazla web sayfası olduğu tahmin edilmektedir. Bu bilgiye göre dünya nüfusunun büyük bir kısmının kullanıcı olduğu internet ortamında arama motorlarının yeri, insanları ulaşmak istedikleri bilgiye, doğru ve hızlı bir şekilde ulaşabilmesini sağlayan yön göstericilerine örnek gösterilebilinir. Bu yön göstericilerin doğru ve hızlı çalışması kuşkusuz en çok kullanıcının işine yarayacak olup insanların ihtiyaçlarını giderme konusunda büyük önem taşımaktadır. İnternette bu kadar çok web sayfası olmasının avantajları olduğu gibi dezavantajları da vardır. İhtiyaç duyduğunuz herhangi bir şeyi içeren web sayfalarının sayısının fazla olması çeşitlilik sağlarken, aynı zamanda hangisinin işinize en çok yarayacağını bulmak zaman maliyetini de beraberinde getirmektedir. Arama motorlarının asıl amacı kullanıcı ile web sayfalarında bir köprü oluşturup kullanıcıya doğru bilgiyi hızlı bir şekilde iletmek olduğundan internetin kullanıldığı her zaman insanın sürekli aktif olarak kullanabilmesi gereken sürekli yenilenmesi ve geliştirilmesi gereken bir ortamdır.

Bu nedenlerden dolayı AraTurka sisteminin geliştirilmesine gerek duyulmuştur. Bu sistem farklı algoritmaları kullanarak aramak istediğiniz bilgiyi karşınıza çıkarmaktadır. Diğer arama motorlardan birinci farkı milli olmasıdır ki milletimizin arama bilgileri yurt içinde kalmasını sağlar. Bir diğer farkı ise bu arama motorunun diğer milli arama motorları gibi sadece sitenin anahtar kelimelerini kotrol etmemektedir. AraTurka sayfaların anahtar kelimeleri dışında aynı zamanda kendisi sitenin içeriğinden anahtar kelime çıkartabilmektedir. Böylece kullanıcının aradığı içeriğin doğruluğu daha yüksek olmaktadır.

Sistemin şuan çalışıyor olmasına rağmen iyileştirme ve geliştirmesi gereken yanları vardır. Sistemin ilk çalışmaya başlatıldıktan sonra yeterince içerik topladıktan sonra bu içerikleri en çok arananlar sırasına göre puanlandırılır. Aynı zamanda görsel işleme algoritmalardan birini kullanarak görsel ve video arama yeteniği kazandırılır. İlerdeki zamanlarda ise alan adı sunuculardan (DNS) sürekli site listesini alıp bu sitelerin içeriklerini de taramış olacak. Bu da sistemin güncel kalmasını sağlar.

Sistemin aynı zamanda kod konusunda geliştirmesi kolay bir şekilde yapılmaktadır. Eklemek istediğimiz fonksiyonların tümünü bir sınıf içinde yazıp sonra bu sınıfı projeye

ekleyip ana fonksiyon içinde çağırmak yeterlidir. Sistemin tüm işlemleri için ayrı bir fonksiyon oluşturulduğu için bu fonksiyonları bulup iyileştirmek kolay olacaktır.

Veri madenciliğinin ilerlemesiyle büyük veri yığınları içerisinden gelecekle ilgili tahminde bulunabilmemizi sağlayabilecek bağıntıların bilgisayar programı kullanarak aranması işlemi daha kolay ve hızlı bir şekilde olacağı için gün giderek arama motorlarından beklenenden daha yüksek performans göstermesi ve ihtiyacı karşılayabilmesi durumu kaçınılmazdır. Veri madenciliği, büyük ölçekli veriler arasından bilgiye ulaşma, bilgiyi madenleme işi olduğu için derinine çalışan arama motorlarının geleceğe daha kalıcı adımlarla ilerlemesi ve özellikle bilgi işleme ve aktarma hızı konusunda sürekli kendini yenileyen bir yapı haline gelmesi beklenmektedir.

6. KAYNAKLAR

[1] [Çevrimiçi]. Available: http://www.motionb.com/blog/arama-motoru-nedir. [Erişildi: 2 Kasım 2017].

[2] [Çevrimiçi]. Available: https://en.wikipedia.org/wiki/Web_search_engine. [Erişildi: 3 Kasım 2017].

[3] [Çevrimiçi]. Available: http://www.teknolojioku.com/internet/googleda-anlamsal-arama-donemi-5a28f2d618e540630d1ce988. [Erişildi: 3 Kasım 2017].

[4] [Çevrimiçi]. Available: https://en.wikipedia.org/wiki/.NET_Framework. [Erişildi: 5 Kasım 2017].

[5] [Çevrimiçi]. Available: https://msdn.microsoft.com/en-us/library/ms972976.aspx?f=255&MSPPError=-2147217396. [Erişildi: 6 Kasım 2017].

[6] [Çevrimiçi]. Available: https://en.wikipedia.org/wiki/C_Sharp_(programming_language). [Erişildi: 7 Kasım 2017].

[7] [Çevrimiçi]. Available: https://tr.wikipedia.org/wiki/Model-View-Controller. [Erişildi: 9 Kasım 2017].

[8] [Çevrimiçi]. Available: http://www.iztim.com/blog/yazilimteknolojisi/mvc-nedir. [Erişildi: 9 Kasım 2017].

[9] [Çevrimiçi]. Available: http://www.borakasmer.com/asp-net-mvc-nedir-ne-ise-yarar/. [Erişildi: 11 Kasım 2017].

[10] [Çevrimiçi]. Available: https://en.wikipedia.org/wiki/Database. [Erişildi: 13 Kasım 2017].

[11] [Çevrimiçi]. Available: https://ozdemiryazilim.wordpress.com/tag/linq-to-sql/. [Erişildi: 14 Kasım 2017].

[12] [Çevrimiçi]. Available: http://www.semgoksu.com/sqlce-linqce-bolum-1-yazisi/551.aspx. [Erişildi: 14 Kasım 2017].

[13] [Çevrimiçi]. Available: http://www.kadinyazilimci.com/linq-teknolojisi/. [Erişildi: 14 Kasım 2017].

[14] [Çevrimiçi]. Available: http://www.yazgelistir.com/makale/linq-baslarken-1. [Erişildi: 14 Kasım 2017].

[15] [Çevrimiçi]. Available: http://deryagunduz.com/?tag=linq-nedir. [Erişildi: 15 Kasım 2017].

[16] [Çevrimiçi]. Available: http://belgeler.org/recs/xpath/. [Erişildi: 19 Kasım 2017].

[17] [Çevrimiçi]. Available: https://docs.microsoft.com/en-us/nuget/what-is-nuget. [Erişildi: 20 Kasım 2017].

[18] [Çevrimiçi]. Available: http://html-agility-pack.net. [Erişildi: 20 Kasım 2017].

[19] [Çevrimiçi]. Available: https://en.wikipedia.org/wiki/Data_mining. [Erişildi: 22 Kasım 2017].

[20] Ş. E. ŞEKER, «Metin Madenciliği,» MISSozluk, [Çevrimiçi]. Available: http://mis.sadievrenseker.com/2014/06/metin-madenciligi-text-mining/. [Erişildi: 23 Kasım 2017].

[21] [Çevrimiçi]. Available: https://www.airpair.com/nlp/keyword-extraction-tutorial. [Erişildi: 25 Kasım 2017].

EKLER

Ek-1: Master'in Kodları

```
using System.IO;
using AraTurkaMaster.Properties;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text.RegularExpressions;
using System.Net;
using HtmlAgilityPack;

namespace AraTurkaMaster
{
    class Program
    {
        static bool showErrors = false;
        static WebSayfa sayfa = null;
        static Guid sayfa_id = Guid.Empty;
        static Rake RAKE = null;
        static void Main(string[] args)
        {
            RequestHandler("Hataları göstersin mi (e/H)?");
            string temp = Console.ReadLine();
            if (temp.Trim().ToLower() == "e" || temp.Trim().ToLower() ==
"evet")
                showErrors = true;
            Engage();
        }
        static void Engage()
        {
            try
            {
                RAKE = new Rake();
                while (true)
                {
                    sayfa_id = NextLink();
                    sayfa = GetHtml(sayfa_id);
                    if (sayfa != null)
                    {
                        SuccessHandler(sayfa.url);
                        InfoHandler(sayfa.Id);
                        if (sayfa.icerik != null)
                        {
                            sayfa.baslik = ExtractTitle(sayfa.icerik,
sayfa.Id);
```

```
                            sayfa.aciklama =
ExtractDescription(sayfa.icerik, sayfa.Id);
                            ExtractLinks(sayfa.icerik, new
Uri(sayfa.url));
                            sayfa.icerik =
ExtractPageText(sayfa.icerik);
                            if (sayfa.icerik != null)
                            {
                                Dictionary<string, double> allkeys =
RAKE.Run(sayfa.icerik);
                                foreach (var kelime in allkeys)

insertSayfaAnahtari(insertAnahtarIfade(kelime.Key), sayfa.Id,
kelime.Value);
                            }
                        }
                        if (sayfa.icerik != null)
                            InfoHandler(sayfa.icerik);
                        if (sayfa.baslik != null)
                            InfoHandler(sayfa.baslik);
                        if (sayfa.aciklama != null)
                            InfoHandler(sayfa.aciklama);
                        if (sayfa.response_code != null)
                            InfoHandler(sayfa.response_code);
                        if (sayfa.content_type != null)
                            InfoHandler(sayfa.content_type);
                        using (DBDataContext db = new DBDataContext())
                        {
                            WebSayfa original =
db.WebSayfalar.FirstOrDefault(w => w.Id == sayfa.Id);
                            original.baslik = sayfa.baslik != null ?
sayfa.baslik : original.baslik;
                            original.aciklama = sayfa.aciklama != null ?
sayfa.aciklama : original.aciklama;
                            original.icerik = sayfa.icerik != null ?
sayfa.icerik : original.icerik;
                            original.response_code = sayfa.response_code
!= null ? sayfa.response_code : original.response_code;
                            original.content_type = sayfa.content_type
!= null ? sayfa.content_type : original.content_type;
                            original.scanned = true;
                            db.SubmitChanges();
                        }
                    }
                    else
                    {
                        using (DBDataContext db = new DBDataContext())
```

```
                        {
                            WebSayfa original =
db.WebSayfalar.FirstOrDefault(w => w.Id == sayfa_id);
                            original.scanned = true;
                            db.SubmitChanges();
                        }
                    }
                }
            }
            catch(Exception e)
            {
                InfoHandler(sayfa.Id);
                InfoHandler(sayfa.url);
                InfoHandler(sayfa.icerik);
                InfoHandler(sayfa.content_type);
                InfoHandler(sayfa.response_code);
                InfoHandler(sayfa.baslik);
                InfoHandler(sayfa.aciklama);
                ErrorHandler(e, null);
            }
        }
        static WebSayfa GetHtml(Guid link_id)
        {
            try
            {
                using (DBDataContext db = new DBDataContext())
                {
                    WebSayfa sayfa = db.WebSayfalar.FirstOrDefault(l =>
l.Id == link_id);
                    if (sayfa == null)
                        return null;
                    HttpWebRequest request =
(HttpWebRequest)WebRequest.Create(sayfa.url);
                    request.AutomaticDecompression =
DecompressionMethods.GZip;
                    try
                    {
                        using (HttpWebResponse response =
(HttpWebResponse)request.GetResponse())
                        using (Stream stream =
response.GetResponseStream())
                        using (StreamReader reader = new
StreamReader(stream))
                        {
                            sayfa.content_type = response.ContentType;
                            sayfa.response_code =
(short)response.StatusCode;
```

```
                            if
(response.ContentType.Contains("text/html"))
                                sayfa.icerik = reader.ReadToEnd();
                        }
                        return sayfa;
                    }
                    catch (WebException webException)
                    {
                        try
                        {
                            sayfa.response_code =
(short)(webException.Response as HttpWebResponse).StatusCode;
                        }
                        catch (Exception e)
                        {
                            ErrorHandler(e, null);
                        }
                    }
                    catch (Exception e)
                    {
                        ErrorHandler(e, null);
                    }
                }
            }
            catch (Exception e)
            {
                ErrorHandler(e, null);
            }
            return null;
        }
        static string ExtractTitle(string html, Guid link_id)
        {
            var htmlDoc = new HtmlDocument();
            htmlDoc.LoadHtml(html);
            var x = htmlDoc.DocumentNode.SelectSingleNode("//title");

            string title = null;

            if (x != null)
            {
                string text = x.InnerText;

                PrepareKeywords(text, link_id);
                insertSayfaAnahtari(insertAnahtarIfade(text), link_id);
                title = text;
            }
```

```
            if (title != null)
                return Regex.Replace(title, @"\s+", " ").Trim();
            return null;
        }
        static string ExtractDescription(string html, Guid link_id)
        {
            var htmlDoc = new HtmlDocument();
            htmlDoc.LoadHtml(html);
            var x =
htmlDoc.DocumentNode.SelectSingleNode("//meta[@name='description']");

            string description = null;

            if (x != null)
            {
                if (x.Attributes["content"] == null)
                    return null;
                string text = x.Attributes["content"].Value;

                PrepareKeywords(text, link_id);
                description = text;
            }

            if (description != null)
                return Regex.Replace(description, @"\s+", " ").Trim();
            return null;
        }
        public static string ExtractPageText(string html)
        {
            var htmlDoc = new HtmlDocument();
            htmlDoc.LoadHtml(html);
            var x =
htmlDoc.DocumentNode.SelectNodes("//body//text()[not(ancestor::style)][n
ot(ancestor::script)][not(ancestor::button)][not(ancestor::select)][not(
ancestor::ul)][not(ancestor::ol)]");
            string page_text = null;
            if (x != null)
            {
                string text = "";
                foreach (HtmlNode node in x)
                {
                    text += " " + node.InnerText.Trim();
                    text.Trim();
                    if (text.Length > 1023) break;
                }
                text = text.Trim();
```

```
                page_text = text.Substring(0, Math.Min(1024,
text.Length));
            }
            if (page_text != null)
                return Regex.Replace(page_text, @"\s+", " ").Trim();
            return null;
        }
        static void ExtractLinks(string html, Uri uri)
        {
            string prePath = uri.AbsoluteUri;
            prePath = prePath.Substring(0, prePath.LastIndexOf('/') +
1);
            var htmlDoc = new HtmlDocument();
            htmlDoc.LoadHtml(html);
            var nodes = htmlDoc.DocumentNode.SelectNodes("//a[@href]");
            if (nodes != null)
            {
                foreach (var node in nodes)
                {
                    try
                    {
                        string link = node.Attributes["href"].Value;
                        if (link.Substring(0, Math.Min(4, link.Length))
!= "http")
                        {
                            if (link == "")
                                continue;
                            if (link[0] != '/')
                                link = prePath + link;
                            else
                            {
                                string tempPre =
prePath.Substring(prePath.IndexOf("://") + 3);
                                link = prePath.Substring(0,
prePath.IndexOf("://") + 3) + tempPre.Substring(0, tempPre.IndexOf("/"))
+ link;
                            }
                        }
                        if (link != null && link.Length > 7 &&
link.Length < 2048 && link.Contains("websitem"))
                        {
                            Guid link_id = insertSayfa(link, null);
                            PrepareKeywords(link, link_id);

insertSayfaAnahtari(insertAnahtarIfade(uri.Authority +
uri.AbsolutePath), link_id);
                        }
```

```
                    }
                    catch (Exception e)
                    {
                        ErrorHandler(e, null);
                    }
                }
            }
            nodes = htmlDoc.DocumentNode.SelectNodes("//img[@src]");
            if (nodes != null)
            {
                foreach (var node in nodes)
                {
                    try
                    {
                        string link = node.Attributes["src"].Value;
                        string title = null;
                        if (node.Attributes["alt"] != null)
                            title = node.Attributes["alt"].Value;
                        if (link.Substring(0, Math.Min(4, link.Length))
!= "http")
                        {
                            if (link == "")
                                continue;

                            if (link[0] != '/')
                                link = prePath + link;
                            else
                            {
                                string tempPre =
prePath.Substring(prePath.IndexOf("://") + 3);
                                link = prePath.Substring(0,
prePath.IndexOf("://") + 3) + tempPre.Substring(0, tempPre.IndexOf("/"))
+ link;
                            }
                        }
                        if (link != null && link.Length > 7 &&
link.Length < 2048)
                        {
                            Guid link_id = insertSayfa(link, title);
                            PrepareKeywords(link, link_id);

insertSayfaAnahtari(insertAnahtarIfade(link), link_id);
                        }
                    }
                    catch (Exception e)
                    {
                        ErrorHandler(e, null);
```

```
                    }
                }
            }
        }
        static Guid NextLink()
        {
            try
            {
                using (DBDataContext db = new DBDataContext())
                {
                    WebSayfa sayfa = db.WebSayfalar.FirstOrDefault(l =>
l.scanned == false);
                    if (sayfa != null)
                        return sayfa.Id;
                    RequestHandler("No link in queue! Specify one
(http://example.com): ");
                    string url = Console.ReadLine();
                    Guid link_id = insertSayfa(url, null);
                    insertSayfaAnahtari(insertAnahtarIfade(url),
link_id);
                    PrepareKeywords(url, link_id);
                    return link_id;
                }
            }
            catch (Exception e)
            {
                ErrorHandler(e, null);
            }
            return Guid.Empty;
        }
        static Guid insertSayfa(string url, string title)
        {
            try
            {
                using (DBDataContext db = new DBDataContext())
                {
                    WebSayfa sayfa = db.WebSayfalar.FirstOrDefault(l =>
l.url == url);
                    if (sayfa != null)
                        return sayfa.Id;
                    Uri uri = new Uri(url);
                    sayfa = new WebSayfa()
                    {
                        Id = Guid.NewGuid(),
                        url = url,
                        website_id = insertSite(uri),
                        path = uri.AbsolutePath,
```

```
                        query = uri.Query,
                        uzanti =
System.IO.Path.GetExtension(uri.AbsolutePath),
                        scanned = false
                    };
                    if (title != null)
                    {
                        sayfa.baslik = title;
                    }
                    db.WebSayfalar.InsertOnSubmit(sayfa);
                    db.SubmitChanges();
                    if (Guid.Empty != sayfa_id)
                    {
                        SiteSitesi ss = new SiteSitesi()
                        {
                            Id = Guid.NewGuid(),
                            site1 = sayfa_id,
                            site2 = sayfa.Id
                        };
                        db.SiteSiteleri.InsertOnSubmit(ss);
                        db.SubmitChanges();
                    }
                    WarningHandler(url);
                    return sayfa.Id;
                }
            }
            catch (Exception e)
            {
                ErrorHandler(e, null);
            }
            return Guid.Empty;
        }
        static Guid insertSite(Uri uri)
        {
            try
            {
                using (DBDataContext db = new DBDataContext())
                {
                    WebSite site = db.WebSiteler.FirstOrDefault(h =>
h.url == uri.Authority);
                    if (site != null)
                    {
                        return site.Id;
                    }
                    site = new WebSite()
                    {
                        Id = Guid.NewGuid(),
```

```
                        url = uri.Authority
                    };
                    db.WebSiteler.InsertOnSubmit(site);
                    db.SubmitChanges();
                    return site.Id;
                }
            }
            catch (Exception e)
            {
                ErrorHandler(e, null);
            }
            return Guid.Empty;
        }
        static Guid insertAnahtarIfade(string word)
        {
            word = word.Substring(0, Math.Min(2048, word.Length));
            try
            {
                using (DBDataContext db = new DBDataContext())
                {
                    AnahtarIfade ifade =
db.AnahtarIfadeler.FirstOrDefault(k => k.ifade == word);
                    if (ifade != null)
                        return ifade.Id;
                    ifade = new AnahtarIfade()
                    {
                        Id = Guid.NewGuid(),
                        ifade = word
                    };
                    db.AnahtarIfadeler.InsertOnSubmit(ifade);
                    db.SubmitChanges();
                    return ifade.Id;
                }
            }
            catch (Exception e)
            {
                ErrorHandler(e, word);
            }
            return Guid.Empty;
        }
        static Guid insertSayfaAnahtari(Guid keyword_id, Guid link_id,
double puan = 10)
        {
            try
            {
                using (DBDataContext db = new DBDataContext())
                {
```

```
                    SayfaAnahtari sayfaAnahtari =
db.SayfaAnahtarlari.FirstOrDefault(l => l.ifade_id == keyword_id &&
l.sayfa_id == link_id);
                    if (sayfaAnahtari != null)
                        return sayfaAnahtari.Id;
                    sayfaAnahtari = new SayfaAnahtari()
                    {
                        Id = Guid.NewGuid(),
                        ifade_id = keyword_id,
                        sayfa_id = link_id,
                        puan = puan
                    };
                    db.SayfaAnahtarlari.InsertOnSubmit(sayfaAnahtari);
                    db.SubmitChanges();
                    return sayfaAnahtari.Id;
                }
            }
            catch (Exception e)
            {
                ErrorHandler(e, null);
            }
            return Guid.Empty;
        }
        static void PrepareKeywords(string text, Guid link_id)
        {
            if (text == null)
                return;
            MatchCollection matches = Regex.Matches(text,
@"\b[\w]{2,}\b", RegexOptions.IgnoreCase);
            insertAnahtarIfadeler(matches, link_id);
        }
        static void insertAnahtarIfadeler(MatchCollection keywords, Guid
link_id)
        {
            foreach (Match match in keywords)
            {
                string word = match.Value;
                try
                {
                    insertSayfaAnahtari(insertAnahtarIfade(word),
link_id);
                }
                catch (Exception e)
                {
                    ErrorHandler(e, link_id + " " + word);
                }
            }
```

```
        }
        static void ErrorHandler(Exception e, object obj)
        {
            if (showErrors)
            {
                ColorfulConsole.Write.error("[  error  ] ");
                ColorfulConsole.WriteLine.error(e.Message);
                ColorfulConsole.WriteLine.error(e.StackTrace + "\n\n");
                if (obj != null)
                {
                    ColorfulConsole.WriteLine.error("------------------
----------------------------------------");
                    ColorfulConsole.WriteLine.error(obj);
                    ColorfulConsole.WriteLine.error("------------------
----------------------------------------\n\n");
                }
                Console.ReadKey();
            }
        }
        static void WarningHandler(object text)
        {
            ColorfulConsole.Write.warning("[ warning ] ");
            ColorfulConsole.WriteLine._default(text);
        }
        static void SuccessHandler(object text)
        {
            ColorfulConsole.Write.success("[ success ] ");
            ColorfulConsole.WriteLine._default(text);
        }
        static void InfoHandler(object text)
        {
            ColorfulConsole.Write._default("[  info   ] ");
            ColorfulConsole.WriteLine._default(text);
        }
        static void RequestHandler(object text)
        {
            ColorfulConsole.Write.primary("[ request ] ");
            ColorfulConsole.Write.warning(text);
        }
    }
}
```

ÖZGEÇMİŞ

Kişisel Bilgiler

Adı Soyadı	: Ibrokhim SHOKIROV
Uyruğu	: Tacik
Doğum Yeri ve Tarihi	: Tacikistan, 28.06.1994
Medeni Hali	: Evli
E-Posta	: ibrokhim.shokirov@gmail.com

Eğitim

Lisans	: Gazi Üniversitesi - (*Örgün Öğretim*)
	Teknoloji Fakültesi, Bilgisayar Mühendisliği (*2014 - Halen*)
Lise	: Lyceum For Gifted Students in Dushanbe (2008-2013)

Dil Bilgiler

Yabancı Dil	: İngilizce, Rusça, Farsça, Türkçe, Arapça

Bilgisayar Bilgileri

Diller	: C/C++, C#, PHP, Delphy, Pascal, Java, ASP, HTML 5, CSS 3, JavaScript, Python, SQL
Programlar	: AutoCAD, Visual Studio, Android Studio, Eclipse, IntelliJ, Arduino
İlgi duyulan alanlar	: Siber Güvenlik, gömülü sistemler, nesne yönelimli programlama, web programlama, yapay zeka, veri madenciliği, algoritmalar

GAZİ GELECEKTİR...