Engineering Sciences
ISSN: 1308-7231 (NWSAENS)
ID: 2019.14.3.1A0434

Status : Research Article
Received: 04.04.2019
Accepted: 10.07.2019

Taner Hacıoğlu, Ali Güneş
İstanbul Aydın University, İstanbul-Turkey
tanerhacioglu@gmail.com; aligunes@aydin.edu.tr

DOI	http://dx.doi.org/10.12739/NWSA.2019.14.3.1A0434	
ORCID ID	0000-0001-7190-2675	---
CORRESPODING AUTHOR	Taner Hacıoğlu	

ÇOK KATMANLI ÇOK KULLANICILI WEB SİSTEMLERİNDE PERFORMANS ANALİZİ VE BİR UYGULAMA

ÖZ

Bu çalışmada, son yıllarda web sistemlerinde rekabetin artmasıyla birlikte önem kazanan performans kavramı incelenmiştir. Performansa etki eden web teknolojilerinin günümüz programlama ve yazılım yöntemleri kullanılarak açıklanmasına gayret edilmiştir. Web uygulamasının kapsamı açıklanmış, katmanlı mimari kullanılarak uygulama geliştirmenin web performansına getirdiği avantajlardan bahsedilmiştir. Ayrıca MVC.NET kullanılarak bir haber portalı geliştirilmiş, uygulama süreçleri diğer güncel web teknik ve teknolojileri ile desteklenerek yazılım, veri tabanı ve sunucu-istemci taraflı testler yapılmıştır. Çalışmada ortalama gecikme süresi, istek yanıt süresi, yük altında çıkış değeri, işlemci, hafıza, I/O ve ağ metrikleri incelenerek site performansı değerlendirildikten sonra kod tarafında yeni güncellemeler yapılmıştır. Olumlu neticeler sonucunda "Çok Katmanlı Çok Kullanıcılı Web Sistemlerinde Performans Analizi ve Bir Uygulama" çalışması tamamlanmıştır. Projeler sürdürülebilir ve kaliteli yazılım geliştirmek isteyen öğrencilere hitap edecek şekilde tasarlanmıştır. Projeler istenildiği zaman genişletilmeye uygun gerçek kullanıma açılabilecek şekilde esnek yapıdadır.

Anahtar Kelimeler: WEB, Performans Analizi, Yük Testi, ASP.NET MVC, Katmanlı Mimari

PERFORMANCE ANALYSIS IN MULTI-LAYER MULTI USER WEB SYSTEMS AND AN APPLICATION

ABSTRACT

In this study, the concept of performance, which gained importance with increasing competition in web systems in recent years, is examined. Efforts have been made to explain the web technologies that affect performance by using today's programming and software methods. The scope of the web application is explained and the advantages of application development to web performance by using layered architecture are mentioned. In addition, a news portal was developed using MVC.NET, application processes were supported with other current web techniques and technologies, software, database and server-client side tests were performed. In this study, average latency, request response time, output value under load, processor, memory, I/O and network metrics were analyzed and site performance was evaluated and new updates were made on the code side. As a result of the positive results, "Performance Analysis in Multi-Layer Multi-User Web Systems and an Application" study was completed. The projects are designed to appeal to students who want to develop sustainable and quality software. The projects are flexible in such a way that they can be opened for real use suitable for expansion at any time.

Keywords: WEB, Performance Analysis, Load Test, ASP.NET MVC, Layered Architecture

How to Cite:
Hacıoğlu, T. ve Güneş, A., (2019). Çok Katmanlı Çok Kullanıcılı Web Sistemlerinde Performans Analizi ve Bir Uygulama, **Engineering Sciences (NWSAENS)**, 14(3):88-103, DOI: 10.12739/NWSA.2019.14.3.1A0434.

1. GİRİŞ (INTRODUCTION)

Ülkemizde web tabanlı hizmet veren firma sayısı her geçen gün artmaktadır. TÜİK tarafından 2017 yılında açıklanan bir araştırma raporuna göre girişimlerin %72'sinin en az bir web sitesine sahip olduğu, ticaret hacminin %3.5'luk kısmının internet üzerinden gerçekleştiği açıklanmıştır. Dünya e-ticaret ortalamasının %8,5 olduğunu belirten rapor Türkiye'de de bu yönde hızla bir artış olacağını belirtmektedir [17]. Ülkemizde e-ticaret uygulamaları dışında e-kütüphane, e-haber, e-lojistik, e-sağlık, e-eğitim, e-devlet gibi birçok farklı internet uygulaması bulunmakta olup her geçen gün daha kapsamlı web sayfaları hizmete girmektedir. Maalesef internet kullanıcısı veya web müşterisi denilen kesim sabırsızdır bu yüzden rekabetçi firmalar daha çok kullanıcıya hizmet verebilmek için daha hızlı daha kaliteli hizmet sunmayı hedeflerler. Web yayınının hem verimli, hem düşük maliyetli hem de rekabetçi olması için mümkün olan tüm unsur ve kaynakların optimum seviyede kullanılması gerekmektedir. Aksi takdirde web sisteminde tıkanıklıklar oluşması, tıkanmaların internet kullanıcılarının işlem hızlarında düşüşe sebep olması, sunucu tarafında istenmeyen durumların ortaya çıkması, veri tutarsızlığı, zamansız kilitlenmeler yaşanması kaçınılmaz olacaktır. Neticede firma birçok ek maliyete katlanmak durumunda kalacak yahut site yeterli ilgiyi görmediği gerekçesiyle yayından kaldırılarak zarara uğrayacaktır.

Web platformlarını performans bakımından optimize etmenin birçok faydası vardır bunlardan bazıları ziyaretçi sayısında artış, işlemci, anakart, hafıza gibi maliyetli parçalarının değişme sürelerinin ötelenmesi, sunucu barındırma ve kiralama maliyetlerinin düşürülmesi, kullanıcı veya müşterilere yansıyan gizli maliyetin düşürülmesidir. Bu makalede çok kullanıcılı bir web uygulaması yayına başlamadan evvel çeşitli çalışmalar yapılıp başarımının yükseltilmesi hedeflenmiştir. Performans çalışmaları yapılırken sunucu tabanlı ASP.NET MVC Framework tercih edilmiş, istemci tarafında ise yine bu çatıyla uyumlu JavaScript ile JS kütüphaneleri olan JQuery gibi güncel betik dilleri kullanılmıştır. Web başarımının iyileştirilmesi ve analizi kapsamlı bir konu olup gerçek hayatta site yayına alındıktan sonra benzer çalışmaların sürdürülmesi refactoring denilen kod düzenlemelerinin yapılması gerekmektedir. Proje başlangıcından itibaren güncel ihtiyaçların belirlenmesi belirlenen bu ölçütlere uygun şekilde kodlamanın yapılması sonradan yapılması gereken elden geçirme, inceleme süreçlerini azaltacaktır. Bu sebeplerden kullanılan yöntemler mümkün olduğunca günümüz ihtiyaç ve gerçekleriyle örtüşen şekilde kapsamlı tutulmaya çalışılmış, aynı gerçekler neticesinde sürdürülebilir, yönetilebilir, kolayca test edilebilir çok katmanlı n-tier yazılım mimarisi kullanılmıştır.

Çalışmadaki web tasarımı mobil, tablet ve masaüstü platformlarına uyumlu Responsive Web Design ile geliştirilmiştir. Böylece daha az kod yazarak daha çok cihaz ve dolayısıyla daha çok internet kullanıcısına erişilmesi amaçlanmıştır. Web performansı ise ihtiyaç ve imkanlar doğrultusunda göreceli bir kavram olduğundan çalışma ile nispeten yüksek performanslı bir web sitesi elde edilmeye gayret edilmiş, aynı donanım ve işletim sistemi üstünde daha fazla kullanıcıya hizmet verilmesi sağlanarak uygun başarımlı web site önermesi yapılmıştır. Bu makalede sadece performans analizine odaklanılmamış olup projenin ilk fazında düşünülmesi gereken performansa olumlu etki eden yazılım tekniklerine de değinilmiştir. Böylece proje hayata geçtikten sonra aniden ortaya çıkan yüksek kapasite maliyeti, ek donanım maliyeti, çeşitli operasyonel düzenlemeler gibi istenmeyen sürpriz yatırımların önüne geçilmesi veya ötelenmesi hedeflenmiştir. İlgili çalışmada günümüzdeki güncel

yöntemler kullanılarak sunucuya ait kaynaklarının uygun seviyede tutulması, optimum sayfa açılış sürelerinin elde edilmesi, bunlarla beraber web başarım sorunu yaşatacak problemlerin çözülmesine dair uygulama teknikleri kullanılarak uzman bir sistem geliştirilmiştir.

2. ÇALIŞMANIN ÖNEMİ (RESEARCH SIGNIFICANCE)

Bu çalışmada web performansını artırma yöntemleri incelenmiş olup diğer çalışmaların aksine donanım çözümlerine değinilmemiştir. Mümkün olduğunda kullanılan yazılım ve veritabanı teknikleri üzerinde yoğunlaşılmış, donanım değişikliği ve ek maliyet oluşturulmadan sadece programlama teknik, yöntem ve araçları ile web performansını artıran etkenler değerlendirilmiştir. Web performansının işletme maliyetleri ve cirosuna olumlu etkileri araştırılmış. Mevcut web kullanıcısı ve/veya müşterilerinin kaybedilmemesi için ağ yazılımcısının en baştan dikkat etmesi gereken unsurlar, ilgili web sitesinin uzun ömürlü ve esnek yapıda olması için kullanılacak olan teknolojilerin seçim kriterleri ve günümüzdeki güncel teknolojilerin hangi amaçlar için kullanıldığı incelenmiştir.

3. LİTERATÜRDE YAPILMIŞ BENZER ÇALIŞMALAR (SIMILAR STUDIES IN LITERATURE)

3.1. Yoğun Yükte Çalışan Sistemlerde Yazılımsal Olarak Davranış Farklılıklarının Analiz Edilmesi (Analysis of Software Behavior Differences in Extreme Intensive Systems)

Tez içerisinde geçen performans uygulamalarında aynı donanım ve yazılım kaynakları ile yaklaşık olarak 10 katı performans iyileşmesi görülmüştür. Yapılan üç iyileşme fazında ilk olarak uygulama katmanı doğru bellek ve işlemci yapılandırması alınmıştır. İkinci fazda veri kaynağına giden katman ön bellekleme ile en az seviyeye çekilmiştir. Son aşamada ise arama ve sorgu gibi işlemler standart veri tabanı mimarisinden yeni nesil veri indeksleme mekanizmalarına alınmıştır. Tüm bunların yanında performans nedenli hatalar %14 seviyesinden %0 oranına alınmıştır. Tezin diğer bölümünde eş zamanlı erişilebilirlik problemlerinden bahsedilmiş ve çözüm önerileri sunulmuştur. Buna göre sistem tasarımlarında yer alan hataların üzerinde durulmuş problemlerin nedenlerine değinilmiştir [2].

3.2. Web Uygulama Sunucularının Performans Analizi (Performance Analysis of Web Application Servers)

Bu çalışma web sunucu ve uygulamalarını izleme, değerlendirme, sınıflama ve performans analizi tahmini amacıyla yapılmış, bu amaçla kullanılan bazı istemci ve sunucularda işlemci, hafıza kullanım oranı, farklı web uygulama teknolojileri, web sunucusunun hizmet edebileceği maksimum istek sayıları kullanılmıştır [3].

3.3. Elektronik Ticaret Sistemleri Ve Geliştirilen B2c Uygulamasının Performans Artırımına Yönelik Yapılan Analizler İle Sunulan Yeni Yaklaşımlar (Electronic Commerce Systems Performance Improvement Analysis and New Approaches for B2C Implementation)

Bu tez çalışmasında şirket ve tüketici arasındaki elektronik ticaret yapısı incelenmiştir. Tez kapsamında sadece bir elektronik ticaret uygulaması oluşturulmamış, aynı zamanda uygulama oluşturulurken sistem aşamalara ayrılarak bir yazılım proje yönetimi uygulanmıştır [16].

4. ÇALIŞMA KAPSAMINDA KULLANILAN GÜNÜMÜZ WEB GELİŞTİRME TEKNOLOJİLERİ (TODAY'S WEB TECHNOLOGIES USED WITHIN THE SCOPE OF THE STUDY)

- **MVC.NET:** ASP.Net MVC, MVC yazılım desenini. NET'e ilave etmek için Microsoft firmasının geliştirdiği framework çatısıdır [1]. Günümüzde MVC denince ilk olarak aklımıza Microsoft'un geliştirdiği ASP.NET MVC Framework gelmesine rağmen, esasen 1979 yılından beri yazılım dünyasında yer almakta olan Java, Php vb. dillerde kullanılan olgunlaşmış bir teknolojidir. Bundan birkaç yıl öncesine kadar ASP.Net zorluklarından bezmiş, hemen hemen tüm kodlayıcılar PHP kodlama diline geçiş yapmıştır. Bu kan kaybına daha fazla dayanamayan Microsoft firması diğer bir çok dilde mevcut olan MVC çekirdeğinin yapısını oluşturarak piyasaya sunmuştur. Böylece 2007 yılında ASP.Net MVC deseninin ortaya çıkmasıyla eski projeler lack ve view state kodlarından arındırılmıştır. Takım çalışmasına uygun olması, katmanlı mimari yapısını desteklemesi, zengin AJAX ve JQuery kütüphanelerine entegrasyon imkanı sağlaması belli başlı avantajları arasında gösterilebilir [4].
- **HTML:** HyperText Markup Language, bir markup (işaretleme, biçimlendirme) dilidir. Programlama dili ile biçimlendirme dili arasındaki farkı boyama kitabı örneği ile daha iyi anlatabiliriz; biçimlendirme dili, boyama kitabındaki resimleri boyar. Programlama dili ise, bu boyama kitabında kaç sayfa olduğunu sayar, ya da hangi sayfaların boyanıp boyanmamış olduğunu belirler [5]. HTML5 ise, İnternet'in en temel ve eski teknolojisi olan HTML'in beşinci sürümüdür. "Günümüzde kullanılan HTML 4.1 sürümü, CSS desteğiyle ne kadar düzenli ve sağlam bir yapıda kodlanırsa kodlansın, yine de fazladan yazılan kodların fonksiyonelliğini bozduğu bilinmektedir. Bu yüzden HTML5, bu ihtiyaçları karşılamak adına geliştirilmeye başlanmıştır" [13].
- **CSS:** CSS web sayfalarına istenen görünümün verilmesi amacıyla kullanılan bir teknolojidir. CSS, HTML ile birlikte web sayfalarını biçimlendiren iki temel teknolojiden biridir. HTML'i iskelet olarak düşünürsek CSS buna giydirilen kıyafet olarak düşünülebilir [6]. Web sitelerine ait stillerin sayfalar arasında paylaşılmasını, sayfaların farklı ortamlarda uyarlanmasını, içeriğin sunumdan ayrılarak HTML, XML vb. CSS bağımsız kodların şişmesini engellemektedir. Ayrıca sunumun büyük ve küçük ekranlar veya yazıcılar gibi farklı cihaz türlerine uyarlanmasını da sağlar [19]. 1995 yılında HTML'in ortaya çıkışıyla o zamanın şartlarına göre geliştirmecilere zengin bir metin biçimlendirme alanı sunulmuştu. Çok fazla geçmeden web geliştiricileri için yeni manevra alanı sunmaması ve sayfa tasarımlarında esneklik sağlamaması sebebiyle geliştiricileri yeni bir web teknolojisi arayışına zorladı. 1996 yılında CSS 1.0 W3C tarafından duyuruldu [6]. CSS kullanım kolaylığı, kullanışlılığı, HTML ile entegrasyonu nedeniyle kısa zamanda web geliştiricilerinin vazgeçilmezi haline geldi. CSS 2.1 yürürlükte olup CSS 3.0 çalışmaları devam etmektedir [6].
- **jQuery:** Amerikalı Bilgisayar Mühendisi ve Programcı olan John Resig'in 2006 yılında geliştirmiş olduğu açık kaynak kodlu bir JavaScript kütüphanesidir. JavaScript Kütüphanesi, herhangi bir web geliştiricisinin işlerini kolaylaştırmak amacıyla yeniden yazmasına gerek olmadan, önceden yazılmış ve belirli bir algoritma yapısına sahip olan komut ve fonksiyonlar bütünü olarak tariflenebilir. JQuery kütüphanesi geliştiricileri çoğu zaman uzun kod blokları yazmaktan kurtarmakta olup HTML ve CSS gibi

tekniklerin yetersiz kaldığı davranışların uygulanmasını sağlamaktadır [18]. JQuery az yer kaplayan bir kütüphanedir. Geliştirdikçe kapasitesi artacak olan bu kütüphane MIT veya GPL lisansına sahip olduğu için uygulamalarda lisansa dair sorun oluşturmamaktadır [6]. jQuery aşağıdaki avantajları sebebiyle dünya genelinde çokça tercih edilmektedir.

- Hafif ve küçük boyutludur. Bu şekilde oluşturulan kütüphanenin tek dezavantajı okunup anlaşılamamasıdır [6].
- Hazır eklentileri vardır. Açık kaynaklı bir kütüphane olduğundan web geliştiriciler kendi eklentilerini hazırlayıp hizmete sunmaktadır.
- Cross-browser desteği vardır. Web geliştiricileri genellikle kendi sitelerine her türlü internet tarayıcısından girilmesini istemektedirler.
- jQuery kolay öğrenilebilir. Temel olarak algoritma ve programlama bilen ve birazda JavaScript diline aşina olan herkes bu kütüphaneyi kolayca öğrenebilmektedir.
- Metot zincirlemesi vardır arka arkaya birçok metot yazılabilir.
- CSS3 Uyumludur.
- Geniş dokümantasyon ve kullanıcı kitlesi mevcuttur.
- Diğer kütüphaneler ile birlikte kullanılabilir.
- IBM, Intel, Wordpress, Mozilla, Adobe gibi firmalar jQuery'ye destek vermektedir. "Amazon.com, microsoft.com, bbc.com, cnn.com, nbc.com, eagames.com, twitter.com ve facebook.com jQuery kullandığı bilinmektedir" [8].

- **REDIS(Remote Dictionary Service):** Redis beş farklı veri tipinde anahtar değer saklayabilen ve ilgili değerleri veri tabanı ile ilişkilendirebilen, hafıza üstünde icra edildiğinden çok hızlı çalışan ilişkisel olmayan bir veri tabanıdır. Verileri hafızada String, Hash, List ve Sıralı List şeklinde tutabilmektedir. Scriptler arasında XML, JSON kullanmaya gerek kalmadan haberleşmeye izin verebilmektedir. Hafızada çalışmasına rağmen iki farklı yöntem destekleyerek veri kaybını engellemektedir. Bu yöntemler belirli zamanlarda snapshot alarak veriyi ilişkisel veri tabanına yazmak, diğeri ise redis komutlarının logdan geri dönülmesi olarak özetlenebilir. Hızlı ve senkron çalışması, veri kaybını engellemesi, bir çok veri türünü desteklemesi sebebiyle tercih edilmiştir [25]. İlişkisel veritabanları gibi kompleks sorguları desteklemediğinden projede önbellekleme için ek veritabanı olarak kullanılmıştır.

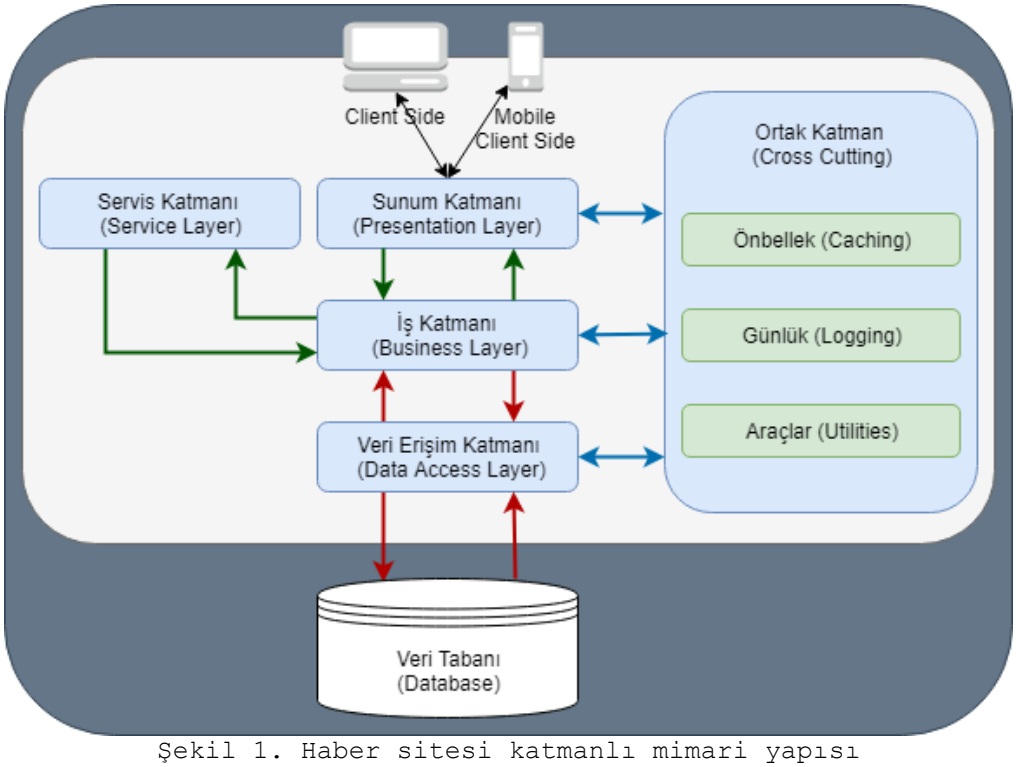
5. RESPONSIVE TASARIM (RESPONSE AND DESIGN)

Web sitesinin mobil, tablet ve cihazlardan girildiğinde site içindeki resim, yazı gibi elementlerinin ekran genişliğine göre yeniden şekillendirilip tam oturması için kullanılan yöntemlere responsive tasarım denilmektedir. Bu tip tasarımlar için açık kaynak kodlu Bootstrap, Metro UI, Semantic UI gibi birçok framework bulunmaktadır. Çok popüler olan bu çerçeveler web sitesine kolayca kopyalayıp yapıştırılabilecek birçok kod parçacığı sağlamaktadır. Fakat bu tip çerçeveler beraberinde bazı dezavantajlarda getirmektedir. Projeye ilk başlarken hazır ve güçlü tasarım sağlayan çerçevelerin bir süre sonra geliştiriciye teknik borç verdiği anlaşılmaktadır. Bunun neticesinde hızlı şekilde başlatılan projede bir süre sonra muğlak HTML kodları, aşırı özelleştirilmiş seçiciler, gereksiz kurallar sebebiyle daha fazla efor sarf edilip karmaşık stiller yazmak zorunda kalınabilmektedir [20]. Web sayfalarında hazır responsive çerçeveler kullanılması veya özel responsive tasarım

yapılması ihtiyaca göre belirlenmektedir. Bu çalışmada manuel olarak yazılan responsive uyumlu css teknolojisi kullanılmıştır.

6. KATMANLI MİMARİ (LAYERED ARCHITECTURE)

Katmanlı yazılım mimarisi, kodlarımızı daha küçük yapılara bölerek esneklik, güncellenebilirlik, kontrol kolaylığı, güvenlik, sürdürülebilirlik gibi birçok yarar sağlamaktadır. Monolithic yaklaşımlarda bir projede tüm kodlar butonlar altında yer almakta proje tasarımı değiştirilmek istendiğinde tüm butonlar yeniden kodlanmak zorunda kalmaktadır. Katmanlı mimari en basit yapısıyla veri erişim katmanı, iş katmanı ve sunum katmanı olarak üç kısımdan oluşmaktadır. İhtiyaca göre diğer üçüne ilave olarak ön bellek, hata yakalama, log kaydı gibi işlemleri yürütecek çapraz katmanlar eklenerek n-tier yapısı genişletilebilir. N-tier uygulamalarda her katman kendi üstüne düşen görevi yerine getirdiğinden tasarımı değiştirdiğimizde veri erişim katmanını değiştirmemize gerek duyulmaz. Yahut veri ve iş katmanlarında yaptığımız değişikliklerden sonra sunum katmanını revize etmemize gerek kalmamaktadır. Katmanlara ayrılmış bir projeyi gerektiğinde birden fazla geliştirici kodlayabileceği gibi oluşan sorunların hangi katmanda ortaya çıktığı da kolayca tespit edilebilmektedir. Gerekli katmanda yapılan iyileştirme uygulamanın genelinde aynı etkiyi yaratacağından uzun vadede geliştirme zamanı, test süreleri ve alınacak riskleri azaltmaktadır. Örneğin kullanıcının girdiği e-posta adresinin doğruluk kontrolü iş katmanında yapılacağından veri tabanında gereksiz işlem yükü yaratılmayacaktır. Mobil uygulamalar, web sitesi, web servisleri ayrı sınıflarda kodlanmalarına rağmen aynı veri erişim katmanını kullanacaklarından uygulamada tutarsızlık ve performans farklılıkları beklenmez. Geliştirici veri tabanına hâkim olmasa da mimariye ait veri nesnelerini kullanarak kayıt, sorgulama, silme gibi işlemleri sorunsuzca gerçekleştirebilir. İlk etapta fazla iş yükü getirmiş olmasına rağmen çeşitli avantajları ve en önemlisi sürdürülebilir yapısı sebebiyle çalışmada katmanlı mimari tercih edilmiştir. Çalışmada üçlü mimariye ek olarak sunum katmanı ile aynı seviyede servis katmanı ve resim sıkıştırma ve küçültme işlemlerini yapacak olan ortak yahut çapraz katman olarak isimlendirilen mimari kullanılmıştır. Aşağıdaki şekilde haber sitesine ait katman yapısı görülmektedir.



Şekil 1. Haber sitesi katmanlı mimari yapısı
(Figure 1. Layered Architecture of News Website)

7. WEB OPTİMİZASYON VE PERFORMANS FAKTÖRLERİ (WEB OPTIMIZATION AND PERFORMANCE FACTORS)

Web performansı belki de site verimliliğini etkileyen en önemli faktördür. Çoğunlukla bir web sitesinin sadece hizmet vermesi yeterli değildir. İstenen hizmeti hem kaliteli hem de hızlı bir şekilde ziyaretçiye sunmayan web sitelerinin kullanıcı sayısında önemli düşüşler gözlemlenmiştir. Ters bakış açısıyla web sitesinde hızlanma sağlayan yazılımlarda kullanıcı ve tercih edilme oranında yükseliş gözlemlenmiştir. Walmart.com her 1 saniyelik performans artışı ile %2 oranında dönüşüm artışı sağlamıştır. Firefox tarayıcı ise ortalama sayfa yüklenme sürelerini 2.2 saniyeye kadar düşürerek indirilme sayısında %15.4 artış yaşamıştır. Böylece 10 milyondan fazla yeni müşteri kazanmıştır [15]. Bir e-ticaret sitesi üzerinde mobil kullanıcılar için yapılan 12 haftalık bir araştırmaya göre sayfanın görüntülenme süresinde 200ms, 500ms, 1000ms'ye gecikmeler oluşturulmuştur. Çalışma sonucunda sayfa görüntülenmesinde %9.4'lük azalma izlenirken %3.5'luk kesimin hiç geri dönmediği ve müşterilerin tamamen kaybedildiği anlaşılmıştır [11].

Tablo 1. HTML gecikmesinin mobil işletme metriklerine etkisi
(Table 1. Impact of HTML latency on mobile business metrics)

	Hemen Çıkma Oranı	Dönüşüm Oranı	Alışveriş Sepeti	Sayfa Görüntüleme
200 Ms	-	-	-	-1.2%
500 Ms	-4.7%	-1.9%	-	-5.7%
1000 Ms	-8.3%	-3.5%	-2.1%	-9.4%

Tablo 1 için araştırma yöntemi şu şekildedir. "12 hafta boyunca, mobil trafik dört gruba ayrılmıştır. Bu dört grup için 200ms gecikme, 500ms gecikme ve 1000ms gecikme olmak üzere tam optimize edilmiş bir bölme testi gerçekleştirilmiştir. Hemen çıkma oranı, dönüşüm oranı,

alışveriş sepeti boyutu ve sayfa görüntüleme olmak üzere dört metrik izlenmiştir. Ayrıca kullanıcıların hızlanan bir site hizmeti almaya başlamasından sonra bile yavaş performansın uzun vadeli etkisini ölçmek için, test sona erdikten sonra 6 hafta boyunca kullanıcı davranışları izlenip analiz edilmiştir" [11]. Walmart araştırmasında genel ve algılanan performans olarak iki ana başlıkta ele alınan web performansını artırmaya yönelik yazılım çözümleri uygulanmadığı takdirde işletme metriklerine müşteri kaybı olarak yansıdığı belirtilmektedir [21]. Web performans artırım adımlarını aşağıdaki gibi birkaç alt başlıkta toplamak daha açıklayıcı olacaktır.

7.1. İstekleri Azaltmak (Reducing Request)

Bir web sitesini açtığımızda sayfa görüntülenmeden önce o sayfa ile ilgili tüm verilerin tarayıcı tarafından indirilmesi gerekmektedir. İndirilen dosya içinde bulunan tüm nesnelere o sayfanın ağırlığı denmektedir. CSS, HTML, Flash, Javascript, resimler ve diğer içerik bir web sayfasının tüm ağırlığını oluşturmaktadır. Tüm ağırlık içinde ortalama olarak resim ve JavaScript kodları %77'lik bir dilimi kapsamaktadır.

Site ağırlığının büyük bölümünü kapsayan içerik için;

- Tarayıcı Önbelleği,
- JS ve CSS dosyalarının küçültülüp gerekiyorsa birleştirilmesi,
- Tembel İçerik Yükleme ile web nesnesinin kullanıcı tarafından gerçekten ihtiyaç duyuluyorsa indirilmesi,
- Küçük resim dosyalarının URI ile kodlanarak base64 formatında direkt indirilmesine izin verilmesi
- Şartlı Yükleme yani kullanıcıya ait web tarayıcısının desteklediği uygulamaların yüklenmesi gibi yöntemler kullanılmalıdır [22].

7.2. Aktarım Boyutunu Azaltmak (Reducing The Transfer Size)

Aktarım boyutunu azaltmak için kullanılabilecek ilk yöntem JS ve CSS gibi metin tabanlı dosyaların sunucu tarafından sıkıştırıldıktan sonra tarayıcıya gönderilmesidir. Apache, IIS vb. web serverlar basit parametre ayarları sonrasında %80 civarı bandwidth tasarrufu sağlayan dinamik sıkıştırmayı desteklerler.

Bunun yanında;

- Resim sıkıştırma,
- SVG, PNG, JPEG gibi formatlar kullanarak büyük boyutlu resimlerden kaçınma,
- Minify yöntemiyle sayfa içi boşluk, sekme gibi karakterleri temizlemek ve sadece gereken işlem için kodlanmış kısıtlı JS kütüphaneleri kullanmak aktarım boyutlarında önemli kazançlar sağlamaktadır [23].
- Farklı ekran boyutları için farklı boyutlarda resimler gösterme
- Web servis gibi veri taşınmasını sağlayan katmanlarda mümkün olduğunca XML gibi eski teknikler yerine JSON benzeri nispeten daha hafif ve yeni formatların kullanılması sağlanmalıdır.

7.3. Sayfa Dönüşümü (Render) Hızlandırmak (Speed Up Page Conversion)

Web sayfasının tamamı veya bir kısmının yenilenmesi DOM reflows ve DOM repaint olarak isimlendirilmektedir. DOM üzerinde öğe ekleme, kaldırma veya güncelleme işlemleri sayfa yenilenmesine sebep olduğundan performansı olumsuz etkilemektedir. Mümkün olduğunca DOM nesnesini etkileyen işlemlerden kaçınılması gerekmektedir. Bazı durumlarda web sayfası yüklendikten sonra JS nesnelerini indirmek

sayfanın doğru çalışması ve gereksiz performans kullanımını engelleyecektir. Daha önce resimler için bahsettiğimiz Lazy Load yani Tembel Yükleme JS nesneleri içinde kullanılabilmektedir. Eğer mobil tasarım tarafında sayfa kodluyorsak onClick() yerine onTouch() nesnesini tercih etmemiz 300ms gibi bir hız artışı sağlayacaktır. Kısacası amaçlanan nesneye uygun kodlama yapılması render performansı için önemlidir. Farklı sitelerden elde edilen içerikler HTML, Flash, JS ile kodlanmış şekilde elde ediliyorsa buna Widget eklentisi denmektedir. RSS içerik okuyucu, hesap makinesi, takvim, video oynatıcı vb. birçok widget internet üzerinde bulunmaktadır. Bu eklentiler çevrimiçi yahut çevrimdışı olabilir. Fakat performans kaygısı olmayan farklı web sitelerinden elde edilen bu tip eklentiler site üstünde ciddi kaynak artışı yaratarak sayfa yüklenmesi ve render süresinde ciddi artışlara neden olabilir. Bunun yerine basit linkler verilerek Widget kullanımından kaçınmak gerekmektedir.

7.4. Veritabanı I/O İşlemlerini Azaltmak (Reducing Database I/O Operations)

Günümüz web siteleri çoğunlukla dinamik yapıda olduklarından kullanıcılarla sürekli olarak veri paylaşımı yapmaktadır. Bunlar alışveriş sepeti, ürün listesi, sınav sonuçları, uçak bileti fiyatları gibi kullanıcıyı ilgilendiren değişken veriler olabilir. Örneğin ürüne ait bir kampanya mevcut ise kampanyadan yararlanan web sitesi üyesi için farklı bir fiyat göstermek gerekecektir. Bunun gibi sabit olmayan veriler için önbellek kullanılmaz ve her seferinde sunucundan veri istenir. Üniversite sınav sonuçlarının açıklanması, banka ödeme günleri, efsane cuma kampanyaları gibi yoğunluğun çok olduğu zamanlarda sistemde oluşan yavaşlıklar uygulama akışında değişmeler kilitlenmeler ve tamamlanmamış işlemlere neden olmaktadır. Bu gibi yavaşlamalarda web kullanıcılarının ilk giriş yaptığı ana sayfa gibi ekranlarda 10 saniye üzeri bekledikten sonra vazgeçtikleri, iç sayfa gezintilerinde ise ortalama 2 saniye üzeri beklemelerde ziyaretçi süreçlerini tamamlamadan işlemden vazgeçtikleri gözlemlenmiştir [24]. Veri tabanlarında yavaşlığa sebep olan ana faktör okuma ve yazma işlemleridir. Okuma işlemi genellikle yazma işlemine göre daha yavaş çalışmaktadır. Veriye erişmek için milyonlarca kayıt içinden seçim yapılması okuma işlemlerinin sistem kaynaklarını daha fazla tüketmesine sebep olmaktadır. Bu sebepten ilk etapta yoğun okuma işlemlerinden kaçınılmalı, web servis, rapor gibi yoğun sorgulama yapan işlemlerin hedeflenen süreyi aşıp aşmadığı ve hangi sayıda okuma yaptığı DBMS Profiler olarak adlandırılan veri tabanı izleme araçları veya açık kaynak kodlu DBeaver benzeri monitoring yapan programlarla takip edilmelidir. Veri tabanı performansı için aşağıdaki yöntemler göz ardı edilmemelidir.

- Kayıt sayısı fazla olan tablolar için indeksler oluşturulmalıdır.
- Execution Plan incelenerek sorgunun en az okuma ile istenen sonucu vermesi sağlanmalıdır.
- Sorguyu küçük parçalara bölmek çoğu zaman hız artışı sağlamaktadır.
- Kod tarafında SQL veriler mümkünse bir kez çekilmeli, count, toplam vb. değerler liste değişkenlerinde tutularak hafıza alanında sorgulanmalıdır.
- Sorgu çekilirken sadece ihtiyaç duyulan satır ve sütunları talep edilmelidir. Direkt SQL sorguları yerine Linq, Entity Framework gibi teknolojiler tercih edildiğinde Lazy Loading etkinleştirilmeli böylece ihtiyaç olan sütunların okunması sağlanmalıdır.

- Web servis ve raporlar için zorunlu filtreler ilave edilerek kullanıcılar ihtiyaç duydukları verilere yönlendirilmelidir.
- I/O işlemlerini azaltmak için NOT, IS NOT NULL, LIKE, ORDER BY ifadeleri dikkatle kullanılmalıdır.
- FOR, LOOP gibi imleç kullanımından kaçınılmalıdır.
- Veritabanını parçalayacak, istatistikleri bozacak ve kirli data oluşturacak işlemlerden kaçınılmalıdır. Örneğin drop table sonrası create table kullanmak yerine truncate table komutu kullanılarak temiz bir çalışma sağlanabilir.
- Yük oluşturacak raporlar için DataWarehouse, DataMart yapıları kullanılmalı, haftalık, aylık, günlük raporların yoğunluğun olmadığı saatlerde hazırlanıp gösterimi sağlanmalıdır [12].
- DBMS izleme yazılımları ile kilitlenen, normalden uzun süre çalışan sorgular kontrol edilerek, gerekiyorsa yeniden yazılmalıdır.
- Veritabanı bakım işlemlerinin muhakkak yapılması parçalanmış dosyaların shrink, vacuum ile küçültülmesi önemlidir [10].

Veri tabanı performansında disk hızına bağlı darboğazlar yaşanabilmektedir. Okuma performansında geleneksel manyetik HDD disklere oranla SSD NAND flaş teknolojili diskler iyi bir çözüm sunmaktadır. Fakat SSD disklerin büyük veri bloklarını yazarken yer açmak için belirli bir bloğu önceden silmek zorunda olmaları, veri kurtarmaya uygun olmamaları ve nispeten yüksek maliyetli oluşları dezavantaj olarak görülebilir. Yoğun sorgu yükü içeren sistemlerde sabit disklere göre fazlasıyla hızlı, SSD disklere göre veri kayıp riski çok az olan ayrıca uygun maliyetli in-memory teknolojiler tercih edilebilir. In-memory teknolojiler veri tabanı işlemlerinde sunucuya ait RAM olarak isimlendirilen geçici hafıza alanını kullanmaktadırlar. Redis, Memcached gibi çözümler verileri disk üzerinde değil sunucu belleği üzerinde yöneterek I/O işlemini azaltmakta bunun neticesinde dinamik verilerin maksimum performans ile ziyaretçiye gösterilmesini sağlamaktadırlar [25].

7.5. Yazılım Geliştirme Süreçleri ve ISO 27001 Bilgi Güvenliği (Software Development Processes and ISO 27001 Information Security)

Bir yazılımcı "Ne kodlarsan kodla en iyisini kodla" mantığı ile geliştirme yapmaya başlamalı ve bu şekilde projelere katkı sağlamalıdır. Nesneye yönelik programlama ve katmanlı mimari kötü, anlaşılmaz ve kirli kod süreçlerini azaltmış olsa da başından itibaren algoritma ve gereksinimlerin iyi analiz edilmediği projeler birçok refactoring yapılmasını gerektirmektedir. Her ne kadar kaliteli kod yazmak tecrübe gerektiren bir durum olsa da, bazen özensiz ve aceleci davranan tecrübeli yazılımcıların kötü kodlama yapabildiği görülmektedir. Bu yüzden yazılım şirketlerinin çalışanlarına geliştirme süreç ve politikalarını aktarmaları önem taşımaktadır. Dünyada yazılım kalite, güvenlik ve politikalarının sorgulandığı, akademisyenler ve firma temsilcilerinin yerel ve ulusal düzeyde bir araya gelerek düzenledikleri birçok çalıştay bulunmaktadır. Bu toplantılar ve uluslararası bazı kurumların gayretleri neticesinde çeşitli manifesto ve yazılım geliştirme prensipleri duyurulmuştur. DevOps kavramları, Agile Manifestosu, ITIL Sürüm Yönetimi, IEEE Yazılım Gereksinim Standartları, IEEE Mühendislik ve Etik Standartları, ISO 27001 Bilgi Güvenliği standartları gibi birçok farklı kaynak bu toplantılar ve yapılan çeşitli çalışmalar neticesinde ortaya çıkmıştır [7]. Yazılımların yaygın olarak kullanılmaya başlandığı ilk yıllarda kaliteli ve olgun yazılım üretmek, son yıllarda ise özellikle güvenli yazılım geliştirmek için çok sayıda

model ve çerçeve üzerinde çalışılmıştır. Bu durumun en büyük tetikleyicisi son yıllarda güvenlik açıklıklarının artmasıdır [9]. Güvenlik açıklarında sistemler yavaşlar, performans etkinliği düşer, ağlar kilitlenmeye başlar ve erişim hızları yavaşlar. Bu yüzden performansa doğrudan ve dolaylı etkisi olan aşağıdaki bazı ipuçlarının dikkate alınması önem arz etmektedir.

- Kodlama yapılırken SQL Injection ve XSS (Cross-site scripting) saldırılarına imkân verilmemelidir.
- Veri tabanı bakım işlemleri periyodik görevler ile otomatik olarak yapılmalıdır. Özellikle kirli veriyi temizleyip database boyutunun küçültülmesi ile execution planları saklayan istatistik kayıtlarının güncellenmesi olası bir çok problemi kendiliğinden çözecektir.
- Gün, ay, yıl benzeri sabit değerler için string değişkenler yerine enum sabitlerinin kullanılması daha uygundur.
- Yazılım sürüm araçları olan Git ve TFS benzeri çözümler kullanılarak eski kodların yorum satırı olarak tutulmaması sağlanabilir. Yorum yazmak gerekiyorsa kısa ve anlaşılır olması sağlanmalıdır. Web sayfası içine yazılan Html ve Js Commentlar tarayıcıya taşındığı için gereksiz yük yaratacaktır.
- Yüksek kayıt barındıran tablolarda muhakkak indeksleme çalışması yapılmalıdır.
- Sayfalama gerektiren liste ekranlarına ait sorgularda join kullanımı yerine AJAX vb. metodlar kullanılarak sadece görünen kayıtlara ait değerlerin okunması sağlanmalıdır. Böylece 100 bin satır join edilerek web kullanıcısının bekletilmesi yerine her sayfa değişiminde az sayıda kayıt okunacaktır. Hatta sürekli olarak sayfalama yapılması düşünülüyor is PostgreSQL gibi ofset komutu içeren farklı bir veri tabanı çözümü düşünülebilir.
- Kod tarafında bir fonksiyon bir sorumluluğu üstlenmeli ve basit yanıtlar geri döndürmelidir. Fonksiyona gereksiz yere çok fazla parametre gönderilmesinden kaçınılmalıdır.
- Muhakkak Exception denilen hata kontrol blokları ilave edilmeli, oluşan hatalar loglanmalı, mümkünse görevlilere uyarı maili gönderilmelidir.
- Bool statement'lar ters kontrol edilmemelidir.
- Switch Case bloklarında default clause boş ve hiçbir işlem yapmayacaksa silinmelidir. Benzer şekilde diğer kontrol bloklarında içeriği null olan kontroller programı yavaşlatacaktır.
- C# benzeri diller kullanılıyor ise unmanaged yani yönetilemeyen kaynaklar using blokları içinde kullanılmalıdır. Böylece ilgili görev çalışması tamamlandığında bellekten atılabilmektedir. Diğer yazılım dillerinde geliştiricinin yönetmesi gereken nesnelerin Dispose edilmesi unutulmamalıdır [26].
- Query Where bloğunda mümkün olduğunca fonksiyon kullanılmasından kaçınılmalıdır. Çünkü şart bloğunda fonksiyon kullanımı Index Seek yapılmasını engelleyici bir durumdur.
- Kodlama yapılırken Cost yani Transaction maliyeti düşük tutulacak şekilde planlanmalıdır. Örneğin bir alışveriş sepetindeki ürün miktarlarının sipariş onaylandığı esnada tek tek kontrol edilmesi yani stok miktar kontrolü yerine tek seferde kontrol edilmesi genel sepet kontrolü daha verimli olacaktır.
- Cost maliyetini düşürmek için sıklıkla execution plan ekranı açılmamalıdır. Öncelikle sorgunun dikkatlice yazıldığından emin olunduktan sonra execution plan incelenmelidir. Çünkü execution

plan hazırlanırken olası tüm ihtimaller değerlendirilip en uygun olan çözüm istatistiğe yazıldığından kodun Index_Seek kullanması sağlanmalıdır. Index_Scan veya Key_Lookup kullanan sorgular yüksek okuma maliyeti yaratacaktır.

8. ÖRNEK WEB UYGULAMASI (SAMPLE WEB APPLICATION)

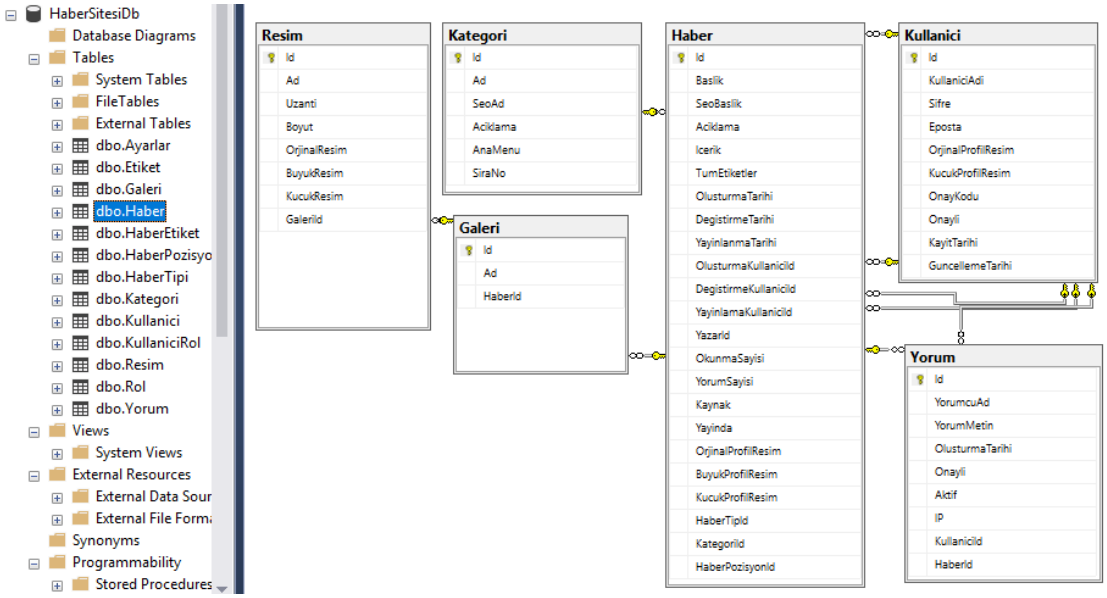
Örnek web uygulaması olarak haber portalı tercih edilmiştir. Bu tercih yapılırken veri tabanında yüksek sayıda metin temelli içerik bulunması, tasarım tarafında çokça resim içermesi, üyelik, yorum, anket gibi kullanıcı etkileşimi sunması ve haber sitelerinin en çok ziyaret edilen sitelerin başında yer alması etkili olmuştur.

8.1. Yazılımın Oluşturulması (Creation of Software)

Çalışmada kullanılan yazılım araçları güncel, yaygın ve ücretsiz sürümler olduğundan herkes tarafından kolayca erişilebilir. Bunlar ASP.NET MVC geliştirmeleri için Visual Studio 2017 Community, veri tabanı tarafında SQL 2017 Express, In-Memory uygulama desteği sağlanmasında Redis ve istemci taraflı geliştirmeler için Jquery, HTML, CSS gibi tamamen ücretsiz araç ve teknolojilerdir. Çalışmalar Windows işletim sistemi üzerinde yapılmıştır. Genellikle Microsoft ürünlerinin tercih edilme sebebi ülkemizde yaygın olarak kullanılan ve en kolay ulaşılabilen işletim sistemi olmasından kaynaklanmaktadır.

8.1.1. Veritabanı Oluşturulması (Database Creation)

Veritabanı MSSQL Server 2017 Express üzerinde oluşturulmuştur. Veri tabanının daha hızlı sonuç üretmesi için Id alanları primary key olarak tanımlanmış ve böylece her tabloda standart indeks şartı sağlanmıştır. Veri bütünlüğünün bozulmaması için primary-key ve foreign-keyler diagram üstünde tanımlanmış, programcının hata yapması durumunda ilişkili kayıtların silinmesi engellenmiştir. Haber açıklama ve içerikler gibi text bazlı alanlarda daha hızlı sonuç elde edilmesi için ve full-text index çalışması yapılmıştır. Resim 1'de projeye ait kritik tablo ilişkileri yer almaktadır.



Resim 1. Haber sitesi veri tabanı Yapısı
(Figure 1. Database structure of this news site)

8.2. Web Uygulamasının Test Edilmesi (Testing the Web Application)

Bir web sisteminde gerçek ortamda aşırı talep altında, bellek, disk, bant genişliği ve işlemci yetersizliği gibi durumlar oluşabilir. Ziyaretçilerin çevrimiçi kullandıkları bir platformda ilgili kaynakları sonradan ilave etmek ciddi sorunlara yol açabileceğinden geliştiriciler benzer ortamlar yaratmak istemektedirler. Sistemin canlıya taşınmasından önce gerekli kaynak ihtiyaçlarını belirleyebilmek için beklentilerin yük testleri ile planlanması gerekir. Yük testlerinde amaç mümkün olduğunca gerçek kullanıcı davranışlarına benzeyen durumların oluşmasını sağlamaktadır. Bunun için aşağıdaki farklı senaryolar planlanmıştır. Testlerin tümü i7 işlemcili, 6GB ram kullanan, Windows10 64bit işletim sistemi üstünde Jmeter kullanılarak yapılmıştır.

Senaryo 1

- Tek bir sayfa çağırımı
- Anlık 300 kullanıcı
- 2 tekrar
- 20 saniye artış süresi

Senaryoya göre sistem anlık 300 kullanıcıya toplam 20 saniyede her bir saniye için 15 kullanıcı ekleyerek ulaşacak ve bu işlemi 2 kez tekrarlayacaktır.

Senaryo 2

- Tek bir mobil sayfa çağırımı
- Anlık 100 kullanıcı
- 2 tekrar
- 10 saniye artış süresi

Senaryoya göre cep telefonu, tablet gibi sistemleri ölçümlemek için anlık 100 kullanıcıya toplam 10 saniyede her saniye 10 kullanıcı ekleyerek ulaşacak ve her bir kullanıcı için bu işlemi 2 kez tekrarlayacaktır.

Senaryo 3

- Web servis veri çağırımı
- XML & JSON karşılaştırılması

Senaryoya göre aynı veri setini kullanan SOAP ve REST servislerin performans testi yapılarak öneride bulunulacaktır.

Senaryo 4

Redis cache hizmetinin kapalı ve açık olduğu durumda anasayfa açılması ve haber arama işlemi yapılarak Cpu, Memory ve Disk I/O değerleri incelenecektir. İşlem yapılırken Performans Monitör programı kullanılacaktır.

Senaryo 5

Redis cache hizmetinin kapalı ve açık olduğu durumda anasayfa için maksimum kullanıcı sayıları tespit edilecektir.

8.2.1. Web Uygulaması Test Araçları (Web Application Test Tools)

Web sitesini gerçeğe yakın sonuç üreten yük testleri ile sınamadan evvel geliştirme aşamasında sayfa bazlı statik testlere sokmak gerekmektedir. Nispeten daha durağan olan bu testler sitedeki yavaşlık, güvenlik, kullanılabilirlik ve bariz hataların giderilmesini sağlamaktadır. İlgili ön testler için geliştiricilere yönelik bazen hataya tarayıcı üstünde müdahale etme imkanı sağlayan ücretsiz site ve eklentiler bulunmaktadır. Bunlardan en yaygın olanları Firefox Firebug, Google Chrome DevTools, Google PageSpeed Insights, YSlow ve Pingdom Website Speed Test'tir [14]. Yük testi ortamı yaratmak amacıyla sektörde birçok kurumsal ve ücretsiz yazılım aracı mevcuttur. Bunlardan yaygın olarak kullanılan kurumsal araçlar LoadRunner, Silk

Performer, Rational Performance Tester iken açık kaynak kodlu olanlar OpenSTA, JMeter, Grinder, WebLoad Basic Edition'dur [27]. Bunlar dışında yine ücretsiz olan Google sağlayıcısı üstünde çevrimiçi hizmet sunan PageSpeed Insights sade yapısıyla geliştiricilere performans hakkında ipuçları vermekte ve önerilerde bulunmaktadır. Geliştirme çevrimdışı olarak yerel bilgisayarda yapıldığından test için Chrome Devtools ve JMeter aracı kullanılmıştır. JMeter ücretsiz olması yanında yük ölçeklemesi yapılmasına da olanak sağlamaktadır.

8.2.2. Web Uygulaması Test Değerleri (Web Application Test Results)

Gerçekleştirilen performans iyileştirmeleri sonucunda elde edilen test sonuç ve yorumları aşağıda anlatılmıştır.

Tablo 2. Senaryo 1: Test sonuçları
(Table 2. Scenario 1: Test results)

Açıklama	Eski	Yeni
Ortalama Yükleme Süresi	214ms	123ms
Kullanıcılar	300	300
Hatalar	1	0
Bant Genişliği	10Mbit	10Mbit

Senaryo 1'de eski durumda web sayfasında orijinal boyutlu resim kullanılmıştır. Yeni durumda ise resim kalitesi %90 olarak ayarlanmış resim kullanılmıştır. Ayrıca js kontrolü ile belirli bir yavaşlamadan sonra sayfadaki reklam banner alanı kapatılmıştır.

Tablo 3. Senaryo 2: Test sonuçları
(Table 3. Scenario 2: Test results)

	Eski	Yeni
Yükleme Süresi	142ms	87ms
Kullanıcılar	100	100
Hatalar	0	0
Bant Genişliği	10Mbit	10Mbit

Senaryo 2'de eski durumda orijinal resim boyutu kullanılmıştır. Yeni durumda ise hem resim kalitesi %90 yapılmış hem de resim çerçeve ölçüleri küçültülmüştür.

Tablo 4. Senaryo 3: Test sonuçları
(Table 4. Scenario 3: Test results)

Açıklama	XML	JSON - Newtonsoft	Performans/Bandwith Etkisi
Boyut (Byte)	25943	12718	%51
Süre (ms)	1083	484	%56

Senaryo 3'te SOAP yerine REST servis kullanılarak taşınan veri boyutu hafifletilmiştir. Mümkün olduğunda JSON tabanlı servis kullanımının avantajlı olduğu tespit edilmiştir. Senaryo 4'te önce redis servisi kapatılarak web sitesi haber arama alanında üç dakika süreyle farklı kelimeler ile haber arama işlemi yapılmıştır. Aynı şekilde redis hizmeti başlatılarak işlem tekrar edilmiş ve performans monitör ekranı izlenmiştir. Sonuç olarak sorgunun sürekli olarak I/O kullanmadığı durumda işlemci ve disk kullanımı belirgin şekilde düşerken, memory kullanımında çok az düşüş gözlemlenmiştir. Redis servisi kullanmak işlemci ve disk okumasında belirgin düşüş ve avantaj sağlamaktadır. Senaryo 5'te maksimum kullanıcı sayısı belirlenirken sistemin yanıt veremez duruma geldiği ilk nokta baz alınmıştır. Sistem kararsız duruma düşmeden önce aşırı disk okuması sonucu CPU

kullanımında yüksek artış yaşamış ve ilk hata değerini döndürmüştür. Redis hizmeti kapalı olduğu durumda sistem 300 kullanıcıya kadar hata vermemiştir. Redis servisi açık olduğu durumda ise web sitesi 425 kullanıcıda hata vermiştir. Sonuç aynı donanım üstünde %40'tan daha çok web kullanıcısına hizmet verilebileceği görülmüştür.

Sistem testleri yapılırken i5 işlemci, 4 GB ram ve Jmeter tarafında 10Mbit bant genişliği kısıtı getirilmiştir. Günümüz hosting şirketleri her ne kadar 100Mbit hız beyan ediyor olsalar da genellikle aynı lokasyonda paylaşımlı birçok web sitesi bulunduğundan bant genişliği kısıtlı tutularak gerçeğe yakın analiz yapılmaya gayret edilmiştir. Neticede aynı donanımı kullanan beş farklı test senaryosunda, web performans artışı sağlanırken kaynak tüketimi azaltılmış olup daha fazla kullanıcıya hizmet verilmesinin mümkün olduğu gözlemlenmiştir.

9. SONUÇLAR (CONCLUSION)

İnternetin en yaygın unsuru sayılan web teknolojileri ilk çıktığı andan itibaren durmadan gelişmekte ve son yıllarda müthiş bir rekabete sahne olmaktadır. İnternet gelirlerinin önemi ve kullanıcı sayısı arttıkça rekabet artmakta bu durum kalite ve başarım oranı yüksek yeni web teknolojilerinin gelişimlerini takip etme zorunluluğu getirmektedir. Zira yükselen standartlara ayak uyduramayan firmaların sektörlerinde kalması zordur. Günümüzde sayfa açılış sürelerinde yaşanan anlık gecikmeler çok sayıda ziyaretçinin artık siteyi kullanmayı bırakmasına sebep olmakta hızlanmalar ise yeni müşteriler kazandırmaktadır. Yoğun yük içeren sistemlerde ziyaretçileri sitede tutmak, aynı zamanda da yüksek sunucu maliyetlerinden kurtulmak için genel ve algılanan performans iyileştirmeleri yapılmasını gerektirmektedir. İşe başlarken performans kaygısı gözetilmeden tasarlanan bir sitenin maliyeti her zaman daha yüksek olmaktadır. Bu çalışmada web projesine başlarken performans kaygısı gözetilmiş ve ilk aşamadan itibaren performansa etki edebilecek faktörler göz önünde bulundurulmuştur. Proje bitiminde sitenin performansı web ve mobil ortamlar için test edilmiş performansa etki etmesi beklenen veri tabanı ve web servis gibi dolaylı diğer etkenler için iyileştirmeler yapılmıştır. Çalışma neticesinde aynı donanım kaynakları üstünde eskiye oranla fark edilir performans iyileştirmesi sağlanmıştır. İyileştirme çalışmalarında taleplerin azaltılması, sayfaların hafifletilmesi, render işlemlerini hızlandırma, disk erişimini azaltma, belirli standartlara bağlı kalınarak kodlama ve test sonrası düzenlemeler yapılmıştır. Bunlar yapılırken yazılım kaynaklı düşük performans nedeniyle oluşan hata oranında düşüş gözlemlenmiştir.

Çalışmada elde edilen bilgiler ışığında, çok kullanıcılı web sistemlerinde kaynak artırımına başvurulmadan önce performansa yönelik iyileştirmelerin uygulanması tercih edilmelidir. Zira çalışmada bahsedilen yöntemlerle geliştirilen web sistemlerinin düşük sorun, yüksek hız ve daha az maliyetlerle hizmet verebileceği gözlemlenmiştir. Neticede web kullanıcıları ile yayıncıların performans beklentilerinin karşılanması, nispeten daha uzun süre ve düşük maliyetli hizmet sunulması ayrıca hız faktörü neticesinde müşteri kaybı yaşanmaması beklenmektedir. Genel bir sonuç olarak diyebiliriz ki web uygulama performansına ışık tutmayı hedefleyen bu çalışmada, işin en başında web başarımı odaklı uygun teknikler, buna yönelik teknolojiler ve doğru yöntemler kullanılarak gelişime açık, sürdürülebilir, esnek, genişletilebilir ve buna mukabil hedeflenen kitleye düşük maliyetler ile hizmet verebilen web uygulamalarının geliştirilebildiği izlenmiştir. Yoğun yük altında çalışan, çok kullanıcılı ve uzmanlaşmış web sistemlerinde performans olgusuna dikkat çekilmesi amaçlanmıştır.

KAYNAKLAR (REFERENCES)

[1] Kızmaz, V.U., (2015). ASP.NET MVC 5, Temmuz, Kodlab Yayın ISBN: 978-605-5201-46-3.
[2] Sakarya, M., (2016). Yoğun Yükte Çalışan Sistemlerde Yazılımsal Olarak Davranış Farklılıklarının Analiz Edilmesi. İstanbul: Beykent Üniversitesi Fen Bilimleri Enstitüsü.
[3] Aydilek, İ.B., (2006). Web Uygulama ve Sunucularının Performans Analizi. Konya: Selçuk Üniversitesi Fen Bilimleri Enstitüsü.
[4] Karayel, A., Bayğuş, İ. ve Yılmaz, U., (2017). ASP.NET MVC İle E-TİCARET, Mayıs, PUSULA, ISBN: 978-605-6730-16-0.
[5] Tokak, B., (2017). HTML5, CSS3 ve JavaScript ile Web Tasarımı, Dıkeyeksen, ISBN 978-605-4898-16-9.
[6] Baltalı, S.J., (2011). KODLAB, ISBN: 978-605-4205-39-4.
[7] http://www.acmagile.com [20.01.2019].
[8] https://publicwww.com/websites/jquery/ [16.06.2017].
[9] Özbilgin, İ.G. ve Mustafa Özlü, M., (). Yazılım Geliştirme Süreçleri ve ISO 27001 Bilgi Güvenliği Yönetim Sistemi. Ankara: Sermaye Piyasası Kurulu & Türk Patent Enstitüsü. https://ab.org.tr/ab10/bildiri/51.pdf.
[10] Kim, S., (2017). PostgreSQL, 9.6 Performance Story, ISBN 978-1521009307.
[11] http://www.webperformancetoday.com/2011/11/23/case-study-slow-page-load-mobile-business-metrics/ [27.11.2016].
[12] Telciler, C., (2013). Veritabanı Kavramı ve MS-SQL Uygulamaları, ISBN 978-605-5106-00-3.
[13] https://tr.wikipedia.org/wiki/HTML5 [15.06.2017].
[14] https://www.keycdn.com/blog/website-speed-test-tools/ [20.06.2018].
[15] https://blog.radware.com/applicationdelivery/wpo/2015/04/new-findings-state-union-ecommerce-page-speed-web-performance-spring-2015/ [19.06.2018].
[16] Kızılkan M.E., (2009). Elektronik Ticaret Sistemleri ve Geliştirilen B2C Uygulamasının Performans Artırımına Yönelik Yapılan Analizler ile Sunulan Yeni Yaklaşımlar. Dumlupınar Üniversitesi, Fen Bilimleri Enstitüsü, Kütahya.
[17] Demirdöğmez, M., Gültekin, N. ve Taş, H.Y., (2018). Türkiye'de E-Ticaret Sektörünün Yıllara Göre Gelişimi. ISSN:2528-9527 E-ISSN:2528-9535.
[18] https://learn.shayhowe.com/advanced-html-css/jquery/ [01.04.2018].
[19] https://www.w3.org/standards/webdesign/htmlcss [01.04.2018].
[20] https://hacks.mozilla.org/2016/04/you-might-not-need-a-css-framework/[21.04.2018).
[21] http://httparchive.org.
[22] Souders, S., (2007). High Performance Web Sites. O'reily Media.
[23] Souders, S., (2009). Even Faster Web Sites. O'reily Media ISBN: 978-0596522308.
[24] Nah, F., (2004). A Study on Tolerable Waiting Time: How Long Are Web Users Willing to Wait? Behaviour & Information Technology, forthcoming.
[25] Josıah, L.C., (2013). Redis in Action. ISBN 9781935182054, ISBN10 1-617290-85-8.
[26] http://www.dofactory.com/reference/csharp-coding-standards.
[27] https://pdfs.semanticscholar.org/presentation/0b13/6d6892372afa70057960d0c78f7c89311a31.pdf [22.04.2018].