Mühendislik Bilimleri ve Tasarım Dergisi
6(4), 579– 589, 2018
e-ISSN: 1308-6693

Journal of Engineering Sciences and Design
DOI: 10.21923/jesd.402784

Araştırma Makalesi Research Article

WEB VE MOBİL TABANLI BAKIM ONARIM VE VARLIK YÖNETİM SİSTEMİNDE ÖNBELLEKLEME YAKLAŞIMLARI

Serdar BİROĞUL[1], **Kenan KOÇER**[*]

[1] Düzce Üniversitesi, Teknoloji Fakültesi Bilgisayar Mühendisliği Bölümü, Düzce, Türkiye
[2] Kocaeli Üniversitesi, Teknoloji Geliştirme Bölgesi, Bimser Çözüm A.Ş., Kocaeli, Türkiye

Anahtar Kelimeler
Bakım–Onarım Sistemi,
Entegrasyon,
AppFabric,
Redis,
Önbellek.

Öz

İşletmeler, bakım-onarım çalışmalarını takip edebilmek için bilgi işlem tabanlı sistemlere ihtiyaç duymaktadır. Bakım-onarım sisteminin sorunsuz çalışması için kaydedilen verilerin ve bu verileri elde edebilmek için yapılan işlemlerin saklanması gerekir. İşletme yöneticileri ellerindeki kayıtlı verilerle, sistemlerden çekecekleri grafikler ve raporlar neticesinde işletmelerin bakım–onarım maliyetlerini düşürebilmektedirler. Gerçekleştirilen bu çalışmada bakım–onarım sisteminin şirketler ve kurumlara uyumlu hale getirilip tamamlanabilmesi için müşteriden gelen geri bildirimler alınmıştır. Bu bildirimlere göre yazılımın geliştirilmesi sağlanmıştır. Yeni modüller eklenmiştir. Entegrasyonlar yapılmıştır. Elektronik imza uygulaması dahil edilmiştir. Yapılan geliştirmeler doğrultusunda kaydedilecek verilerin yönetilmesi ve sistemin hızının düşmemesi için önbellekleme ihtiyacı ortaya çıkmıştır. Kullanıcıların arayüzlere daha hızlı ulaşması ve sistemi dinamik kullanabilmeleri için AppFabric teknolojisi kullanılarak önbellekleme sağlanmıştır. Yapılan yeni geliştirmelerde sadece kullanıcı değil sistemin geliştiricilerinin de verileri kolay elde edip yönetebilmeleri için AppFabric uygulaması yerine Redis teknolojisine geçilmiştir. Böylece hem sistem geliştiricileri hem son kullanıcılar verileri önbellekten kolayca yönetebileceklerdir.

MOBILE AND WEB BASED SYSTEM FOR MAINTENANCE & REPAIR AND ASSET MANAGEMENT CACHING APPROACHES

Keywords
Maintenance & Repair System,
Integration,
AppFabric,
Redis,
Cache.

Abstract

Enterprises (Operations) need IT-based systems in order to carry out repair and maintenance works. To ensure the proper functioning of the repair and maintenance system, it is necessary to obtain and store data on the operations that are carried out. Managers of the enterprise can reduce the cost of repair and maintenance by using the graphics and reports that they can produce with the system with the help of the recorded data. In this work, feedback coming from the customer has been reviewed and evaluated to make the maintenance and repair system more complete and compatible with the operations of the companies and institutions. Software has been improved in accordance with feedback taken from the customer. New models have been added. Integrations have been completed; and electronic signature application has been included. There is a need for caching in order to manage the recorded data in the direction of the developments made and to keep the speed of the system from falling. Caching is provided using AppFabric technology so that users can access the interfaces faster and use the system dynamically. New developments have been made to Redis technology instead of AppFabric application so that system developers can easily obtain and manage the data, not only the user but the system developers and end users can easily manage the data from the cache.

[*] İlgili yazar / Corresponding author: kenankocer90@gmail.com

Alıntı / Cite

Biroğul, S., Koçer K., (2018). Web ve Mobil Tabanlı Bakım Onarım, Varlık Yönetim Sistemi Önbellekleme Yaklaşımları, *Journal of Engineering Sciences and Design*, 6(4), 579 – 589.

Yazar Kimliği / Author ID (ORCID Number)
S, Biroğul, 0000-0003-4966-5970
K. Koçer, 0000-0002-9756-7577

Makale Süreci / Article Process	
Başvuru Tarihi /Submission Date	07.03.2018
Revizyon Tarihi / Revision Date	15.05.2018
Kabul Tarihi / Accepted Date	02.10.2018
Yayım Tarihi / Published Date	10.12.2018

1. Giriş

Bir firmanın kuruluş aşamasında ilk adım olarak fiziksel bir birim oluşturulur. Daha sonra bu firmanın üretime geçebilmesi için ekipmanlarının alınması ve tesisin kurulması gerekmektedir. Tesis kurulduktan sonra üretime başlanabilmesi için üretimde kullanılacak hammadde ve bu maddeyi işleyecek olan kaynakların aktarılması gerekir. Bu işlemler sonucunda elde edilecek işlenmiş ürünün satış ve pazarlamasının yapılması döngünün devamlılığını ve tesisisin düzenli olarak çalışmasını sağlar. Burada firmanın rekabet koşullarında ayakta kalabilmesi ve büyüyebilmesi için hammadde kaynaklarını planlaması gerekir. Üretimde kullandığı kaynakların ve personellerin iş güçlerini ve performanslarını takip edebilmeli ve bu gücü zinde tutabilmelidir. Ayrıca firmanın karlılık oranını arttıracak olan en önemli etken, firmanın üretime geçebilmesi için aldığı ekipmanların ve kurmuş olduğu tesisin bakımlarının düzenli olarak yapılmasıdır. Sonuç olarak tesisin işleyiş gücünün ve yaşam süresinin uzaması sağlanmış olur.

Firmaların verimliliğinin ve kar oranlarını arttırmalarının arkasında varlık yönetim sistemleri yer almaktadır. Varlıkların bakımından sorumlu olan bakım personelleri, varlıkların bakımı için gerçekleştirmesi gereken iş adımlarına ve bakım planlarına hakimdirler. Buna rağmen işletme büyüdüğünde ve varlık hatları genişlediğinde tecrübeli bakımcılar hatta yetişememektedir. Bu durum firmayı yeni kaynak ve personel istihdamına yönlendirmektedir. Sonuç olarak fazladan maliyet ortaya çıkmaktadır. Ayrıca yeni personellere iş adımlarının ve bakım talimatlarının aktarılması problem yaratmaktadır. Bunların dokümante edilmesi ve yönetilmesi için bakım ve varlık yönetim sistemlerine ihtiyaç vardır. Firma sorumluları günlük, aylık ve yıllık olarak ne gibi bakımların yapıldığı, maliyetin ne olduğu, personel yönetiminin ne aşamada olduğu, sistemin/makinelerin duruş süreleri, işçi çalışma süreleri gibi bilgileri görmek istemektedirler. Bu grafikleri de bakım ve varlık yönetim sistemlerinden çekebilmektedirler.

2. Bakım Yönetimi

Bakım, bir parçanın istenilen ömür çevrimi içerisinde gerekli fonksiyonlarını yerine getirebilecek şekilde korunması veya muhafaza edilmesi için bütün teknik eylemlerin bir bütünü olarak tanımlanabilir (Marques, 2007). Bakım yönetimi, bakım önceliklerini, stratejilerini, planlamasını, kontrolünü, denetlenmesini ve organizasyonda ekonomik açıdan iyileştirmeleri içeren metotlar gibi sorumluluk ve uygulamalar ile ilgili bütün yönetim aktivitelerinin belirlenmesi olarak tanımlanabilir (Taşın, 2006). Yetersiz bakım yönetimi sonunda makina arızaları meydana gelir. Ham madde yönetilemez. Üretimde duruşlar olur. Üretimde meydana gelen gecikme veya iptaller ancak bakımın programlı ve kurallara uygun yapılmasıyla önlenebilmektedir(Bal, 2013).

Bakım yönetimi içerisinde bakımlar planlı ve plansız bakım olarak ikiye ayrılır. Plansız bakım, işletmelerde arıza çıktıkça yapılan bakım ve onarım şeklidir. Bu nedenle, onarım sırasında üretim kaybı fazla olmaktadır. Planlı bakım, kontrollü ve düzenli bir biçimde yapılan ve içinde her türlü bakım çeşidini içeren bir bakım çeşididir.

Arızi(Plansız) bakım, ekipmanlara işlevlerini yitirinceye, bozuluncaya kadar bakım yapılmamasını baz almaktadır. En ilkel bakım sistemidir. Bakım, ekipman arızalandıktan sonra yapılır. Periyodik bakım, ekipmanların işlevlerini yerine getirdikleri süre içerisinde önceden belirlenmiş zaman periyodlarında bakım işlemlerinin yapılmasıdır. Önleyici bakım, arızaya neden olabilecek temel faktörlerin ortadan kaldırılarak makinenin çalışmaya dayanıklı hale getirilmesidir. Problemlerin önceden tanınması ve giderilmesi işlemine önleyici bakım denir(Ayrancı, 1997). Kestirici bakım yönteminde, ekipmanların çalışma koşulları ve karakterleri göz önünde bulundurulur. Yapılan program çerçevesinde, üretimi durdurmadan bazı parametrelerin kontrolleri ve ölçümleri yapılmaktadır. Sonuçlar dahilinde ekipmanda oluşabilecek arızaların gelişimi incelenir. Hata belirlenirse gerekli yedek parça temin edilir, üretim durdurulur, kısa süre içerisinde bakım yapılır ve tekrar üretime devam edilir(Chaneski, 2002).

Toplam verimli bakım (TVB) tüm çalışanların katılımının ön görüldüğü, küçük grup faaliyetleri aracılığı ile gerçekleşen verimli bakım olarak tanımlanabilir. İşletmede üretim faaliyetleri içerisinde çalışan kaynak ve ekipmanların tamamının katılımını gerektirir. Operatörlere üzerlerinde çalıştıkları ekipmanların otonom bakım sorumluluğunu getirir. Arızaları önleyen, ekipman verimliliğini arttıran bir yaklaşımdır. Japon Fabrika Bakım Enstitüsü(JIMP) tarafından geliştirilen, ekipmanların sıfır arıza ve minimum üretim kayıplarına sahip olmalarını

hedefleyen bir kavramdır (İ.T.Ü. Department of Information Technologies, 2017).

2.1. Bakım Yöntemlerinin Karşılaştırılması

Bakım yöntemleri arasında en yüksek maliyet, işgücü, üretim kaybı ve tamir zamanına sahip olanı düzeltici bakımdır. Kestirimci bakım planlaması, makinalar üzerinden, periyodik aralıklar ile alınan, fiziksel parametre ölçümlerinin zaman içindeki eğilimlerini izleyerek, makina sağlığı hakkında geleceğe yönelik bir kestirimde bulunma yöntemidir(Acar, 2014). Toplam verimli bakım ise en etkin ve gelişmiş bakım yaklaşımıdır. Toplam verimli bakım, çalışanların bilgi ve becerilerinin artırılması, kullanılan ekipmanların en iyi şekilde korunması, tüm bakım faaliyetlerinin bilgisayar ortamında takip edilmesi ve gerekli önlemlerin alınmasıyla sıfır kaza, sıfır hata ve sıfır plansız duruşu amaçlayan bir yönetim sistemidir(Küçük, 2016).

3. Bakım Onarım ve Varlık Yönetim Sistemi(Boys)

Bakım–onarım varlık yönetim sistemi, firmalarda var olan ekipman ve makinelerin bakım süreçlerinin planlı ve sağlıklı bir şekilde tutulmasını ve yapılmasını sağlar. Bu işler yapılırken harcanan malzeme ve yedek parçaların stok kontrollerinin yapılmasını sağlar. Bakımı yapan personel ve kaynakların çalışma bilgilerini ve işçilik süresi ve işçilik maliyetlerini tutar. Miktarı azalan yedek parça ve malzemelerin satın alma yönetimini ve satın alma süreçlerini tutar.

Önceden ve hali hazırda ticari olarak kullanılan Boys sisteminin gereksinimleri karşılamada yetersiz kalmasından dolayı bu çalışma paralelinde yenilenmiştir. Müşteri istekleri doğrultusunda yeni geliştirmeler yapılmıştır. Bu geliştirmeler kapsamında yeni modüller ve fonksiyonlar eklenmiştir. Kullanıcı dostu ve daha fazla kullanıcı ihtiyacına cevap verebilecek hale getirilmiştir. Sürekli olarak yaşayan bir organizma gibi geliştirmelere devam edilmektedir. Gelişmeler neticesinde sistemin dinamik ve hızlı şekilde kullanılabilmesi için önbellekleme ihtiyaçları ortaya çıkmıştır. Kullanıcıların arayüzleri kullanma aşamasında bekleme sürelerini azaltmak ve verileri sunucu yerine önbelleğe alarak kullanım hızını arttırmak adına AppFabric teknoloji ile altyapı düzenlemesi yapılmıştır. Böylece son kullanıcıların bekleme süreleri ve programın dinamikliği arttırılmıştır. Sonra yapılan yeni araştırma geliştirme çalışmaları neticesinde uygulamayı geliştiricilerin önbelleğe alacakları verileri daha dinamik seçebilmeleri, bu verileri görebilmeleri ve yönetebilmeleri açısından AppFabric teknolojisi yerine Redis teknolojisine geçilmiştir. Böylece hem son kullanıcılar sistemi kullanırlarken hız kazandılar hem de geliştiriciler önbellek verilerini daha dinamik yönetebildiler.

4. Boys'un Sistem Mimarisi

Boys Varlık ve Bakım Yönetim Sistemi servis tabanlı mimariye sahip ve Internet Information Services (IIS) üzerinde çalışan web tabanlı sistemdir. Servise dayalı mimari, birbirinden bağımsız çalışan servis ve uygulamaların birbirleri ile sistemsel olarak bütünleşmesine imkan sağlayan bir sistem tasarımı anlayışıdır(Boy, 2017). SOA teknolojisindeki servisler yapboz parçalarına benzetilebilir. Servislerde yapılan yer değiştirmeler ve fonksiyonel değişiklikler yapboz parçalarının yerlerinden sökülüp istenilen yere takılması ya da yeni parçaların eklenmesi kadar esnek bir işlemdir. Süreçler bir bütün halinde bulunsalardı, parçalı bir yapı olmasaydı, yeni eklentiler ekleyip çıkarmak, parçaları birleştirmek hiç de kolay olmazdı(Şenferah, 2012).

Boys kaynak olarak Oracle ve Microsoft Sql Server veritabanlarından veri çekme yeteneğine sahiptir. Modüller üzerinde onay süreçlerinin kullanılması gerektiğinde (Bakım iş talebi ve satın alma süreçleri için), istemci araçları servisler üzerinden sunucuya 80 portundan iletişime geçmektedir. Böylece istemcilere port ayarı yapılmasına gerek duyulmaz. Boys' un alt yapısında altyapı katmanı, orta katman ve istemci katmanı olarak 3 ana kategori yer almaktadır(Akgündüz, 2015). Şekil 1' de istemci ve orta katman, orta katman ve altyapı katmanı ile ilişkiler gösterilmiştir.

4.1. Boys'un Altyapı Katmanı

Bu katman 3 bölümden oluşmaktadır.

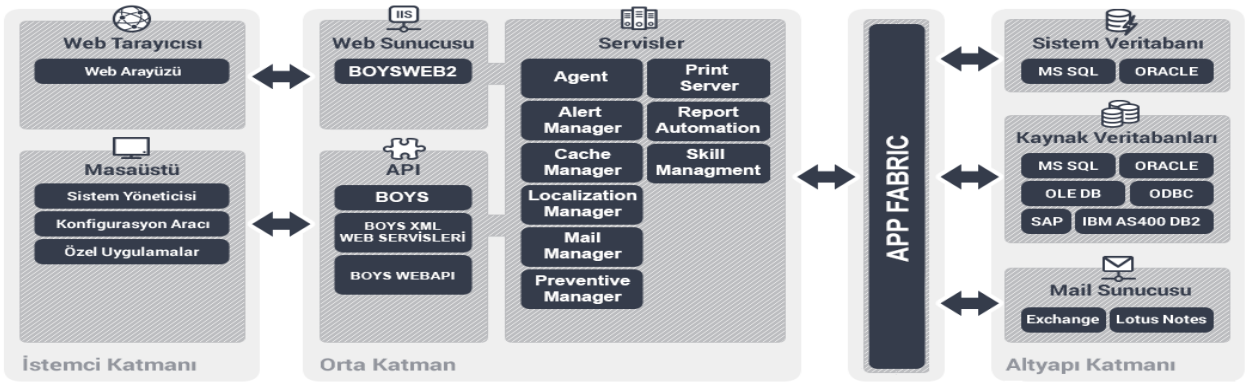
1)Sistem veri tabanı: Boys sisteminin verileri çekmek ve yönetmek için kullandığı Microsoft SQL Server ya da Oracle veri tabanlarını ifade etmektedir.

2)Kaynak veri tabanları: Boys sistemi veritabanlarına veya sistemlere erişerek verileri bakım emirleri içerisinde ve/veya malzeme yönetim sistemi içerisinde kullanabilmektedir. Örneğin MS Sql, Oracle, Odbc, SAP, LOGO Tiger, LOGO Unity, NETSIS ve Xml Servisleridir. Bunlara ek olarak dışa dönük veri bağlayıcılara erişimlerde sağlanabilir.

3)Mail Sunucusu: Sistem üzerinden gönderilecek maillerin gönderimini sağlayacak servistir. Kullanılacak mail sunucusunun SMTP, POP3 destekli olması yeterlidir.

4.2. Boys'un Orta Katmanı

Web sunucular, API' ler ve servislerin yer aldığı katmana verilen isimdir. Servisler, Web sunucular ve API'ler orta katmanda çalışmaktadır. Bu katmanda ki Boys' a ait servisler altyapı katmanı ile bağlantılı çalışmaktadır.



Şekil 1. Boys'un Sistem Mimarisi

Aşağıda Boys servislerinin kullanım içerikleri açıklanmaktadır;

1)Agent: Zamanlanmış bakım işlerini ve bazı küçük veri aktarım işlerini yapar.

2) Alert Manager: Uygulama içerisinde, kullanıcı tarafından ayarlanan uyarı işlerinin yürütülmesinden sorumludur.

3) Cache Manager: Çok kullanılan verilerin sunucu hafızasına yüklenerek kullanıma hazır hale getirilmesinden sorumludur.

4) Localization Manager: Uygulama arayüzünün çok dilli yapıda çalışmasını sağlar.

5) Mail Manager: Uygulamanın e-posta gönderim işlerinin yerine getirilmesinden sorumludur.

6) Preventive Manager: Periyodik olarak sistemi tarayarak, bakım için zamanı gelen periyodik bakım tanımlarından iş emri oluşturmaktan sorumludur.

7) Print Server: Yazdırılma listesinde bekleyen iş talep ve iş emri formlarının ilgili yazıcılara gönderilmesinden sorumludur.

8) Report Automation: Uygulama içerisinden, kullanıcı tarafından belirlenen raporların otomatik dağıtılmasından sorumludur.

9) Skill Management: Bakım iş emirlerine yeteneğe göre otomatik bakım sorumlusu atanmasından sorumludur.

4.3. Boys'un İstemci Katmanı

İstemci katmanı kullanıcılara yönelik arayüzlerden oluşmaktadır. 2 kısımdan oluşur.

Web Uygulamalar: Kullanıcılar ile tarayıcılar arasında iletişimi sağlayan uygulamalardır.

Masaüstü Uygulamalar: Boys uygulamasının yönetici araçlarından oluşan Windows tabanlı masaüstü uygulamalarıdır.

1)System Manager: Sistem araçlarının günlük kayıtlarının izlenebildiği ve yönetilebildiği yönetici aracıdır.

2)ConfigurationEditor: Uygulama kullanım parametrelerinin ayarlandığı ve izlendiği yönetici aracıdır.

3)Workflow Studio: Bakım iş talebi ve satın alma süreçlerinin kurgulandığı kısıtlı kullanım lisanslı iş akış yönetici aracıdır.

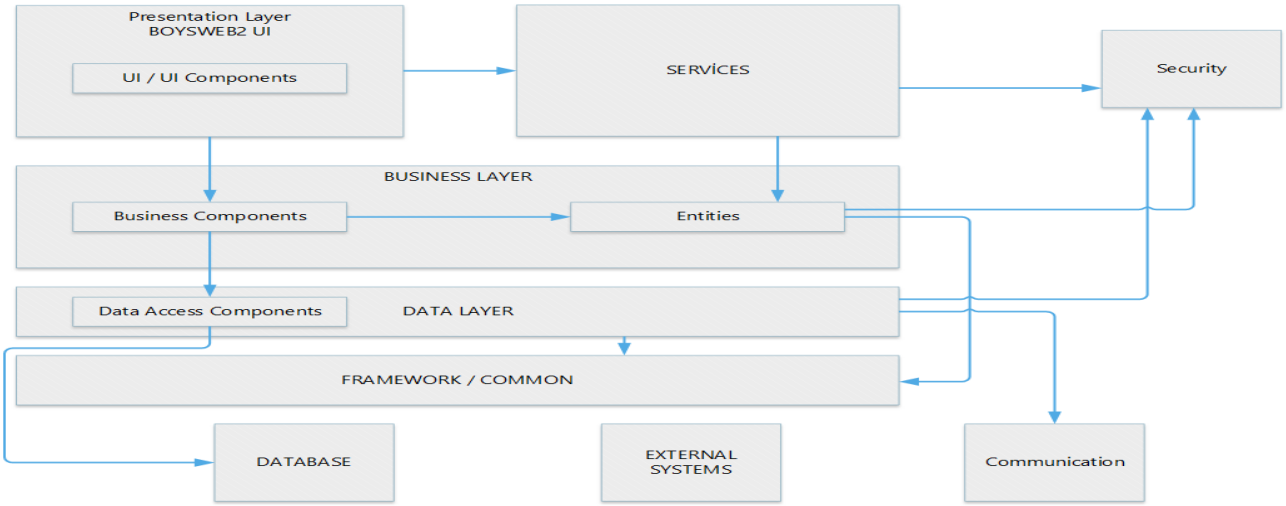
4)ServerConfigurationEditor: Uygulama veritabanı bağlantısının ayarlandığı yönetici aracıdır.

5)AppFabricConfigurator: Bir önbellek(Cache), gerekli yapılandırmayı bilen tüm programlarca erişilebilir, kullanılabilir/ayarlanabilir ortak hafıza alanıdır. Her bir önbellek alanı kendisine ait tekil ismi üzerinde kullanıma sunulmaktadır. Bölgeler(Region), önbellekler içerisinde yer alan yapılar olup ek veri taşıyıcı alanlardır (Doğan, 2014). Bölgeler üzerinden önbellekler ile yapabileceğiniz standart işlemleri yapabilir, verdiğiniz anahtar üzerinden istediğiniz nesneye ulaşabilirsiniz. Bu özelliklerinin yanında; tag adı verilen ve string veri türünde veri tutabilen alanlar sayesinden bölge içerisindeki tüm nesnelerde arama yapılabilir (Doğan, 2014). AppFabricConfigurator, Boys üzerinde cache ve region oluşturma, silme ve yönetme işlemlerinin yapıldığı AppFabric yardımcı yönetici aracıdır.

6)ExcelImport: Excel dosya içerisinden önceden belirlenmiş yapıya uygun olarak hedef tabloya veri aktarımı yapan yönetici aracıdır.

4.4. Boys Katmanlarının Bağlılıkları

Boys uygulamasındaki katmanların bağlılıkları aşağıda şekil 2' de detaylı şekilde anlatılmıştır.



Şekil 2. Boys Katmanları

Presentation Layer: Verinin anlaşılır halde son kullanıcıya sunulduğu katmandır. Bu katmanda Boys web arayüzü ve native mobil uygulamalar bulunmaktadır. Bu katmanın servis katmanı ve Business katmanına bağlılığı vardır.
Services: Uygulamada arka planda çalışan ve veri dağıtım ve iletişim görevlerini üstlenen katmandır. Bu katmanın security ve entity katmanlarına bağlılığı vardır.

Businnes Layer: Uygulamanın iş mantığının yer aldığı katmandır. Data, entity ve framework katmanlarına bağlılığı vardır.

Data Layer: Veritabanına erişimin sağlandığı ve transaction'ların yönetildiği katmandır. Database, framework ve communication katmanına bağlılığı vardır.

5. Boys Önbellekleme Yaklaşımları

Önbellekleme web sayfasının sıklıkla erişeceği sayfaların bir kısmını ya da tamamını veya belirli bir verinin bellekle tutulması işlemidir. Bu işlemi performansın iyileştirilmesi için yapılmaktadır. Bir verinin bellekten okunması veritabanı ya da diskten okunmasından çok daha hızlı sonuç vermektedir. Önbelleklemede dikkat edilmesi gereken nokta belleğe alınacak verinin iyi optimize edilmesi gerekir. Böylece bellek şişmesi ve programın hata vermesinin önüne geçilmiş olur.

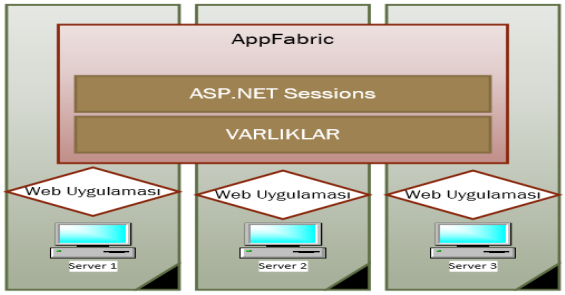
5.1. AppFabric Nedir

IIS (Internet Information Server), web sayfalarının yayınlanmasını ve web uygulamalarının çalışmasını sağlayan, istemcilerden HTTP ve FTP üzerinden gelen talepleri Microsoft Windows sunucu tabanlı işletim sistemlerinde karşılayan birimdir(Kılıçaslan, 2013).

AppFabric , IIS üzerinde çalışır ve önbellek verilerini bilgisayar belleğinde saklar. Böylece sıklıkla kullanılacak olan veriler için sunucuya gidip gelmek yerine bilgisayar belleğinde saklanan bu veriler kullanarak programda hızdan ve maliyetten kazanç sağlanabilmektedir(Lowy, 2015). Bu kazancı sağlayabilmek ve belleğin çok fazla şişmesinin önüne geçebilmek için hangi verilerin appfabric tarafından cachelenmesi gerektiği kararının güzel verilmesi gerekir. Boys uygulaması appfabric üzerinde kullanıcı bilgilerini (kullanıcı adı, şifre vb.), uygulamada kullanılan arayüzlerin özelliklerini, uygulama ekranlarında kullanılan yazıların çoklu dil dosyalarını, listeleri, uygulamadan dönderilecek mesajları, uygulama ekranlarının sayfa renklerini saklamaktadır.

5.2. Boys Neden AppFabric'e İhtiyaç Duyar

Boys bir işletme tarafından kullanılırken, işletmenin bünyesinde yer olan varlıkların tamamı Boys sistemine tanımlanır. Veritabanında kayıtlı olan her bir varlık için binlerce kullanıcı Boys sistemine girerek bu varlıklara ait arızaları bildirmek için veritabanına iş talebi ve iş emri kayıtları atmaktadırlar. Böylece veritabanına milyonlarca başvuru olmaktadır. Veritabanı bu yükü ne ölçüde kaldırabilecek ve bu isteklere ne kadar hızla yanıt verebilecek soruları ortaya çıkmaktadır. Kullanıcıların çok fazla beklemelerin önüne geçmek, hatta aşırı yüklenmelerden kaynaklı sistem çökmelerinin önüne geçebilmek adına AppFabric uygulamasından yardım alınmaktadır.



Şekil 3. Boys AppFabric Uygulama Şeması

Appfabric uygulaması ile kullanıcılar, varlıklar, sayfa özellikleri ve düzenleri, sayfada kullanılan listeler, mesajlar önbellekte tutulmaktadır. Böylece yeni bir varlık eklenmediği veya var olan bir varlık güncellenmediği takdirde tüm varlıklar önbellek sunucularına yazılacaktır. Boys' a giren tüm kullanıcılar varlıklara ulaşmak için veritabanına başvurmayacaktır. Verileri önbellek sunucularından hazır olarak alacaklardır. Bu sayede veritabanına yapılan başvuru sayısı ve yükü azalacaktır. Eskiye nazaran çekilecek olan veriler için harcanan süre kısalacaktır. Şekil 3' de farklı sunucular üzerinden Boys uygulamasına erişilmeye çalışıldığında AppFabric uygulaması ile önbelleğe alınan verilerin şeması yer almaktadır.

Use-CacheCluster : AppFabric uygulaması komutlarını kullanılacaktır.

New-Cache SUNUCU1: Boys uygulamasının kullanacağı yeni sunucuları tanımlamaktadır.

Remove-Cache SUNUCU1: Boys uygulamasının kullanmaktan çıkardığı sunucuyu Appfabric uygulamasınında önbelleklemekten çıkarması için kullanılır.

Start-CacheCluster: AppFabric servisini çalıştırarak önbellekleme işlemine başlanılmasını sağlar.

Stop-CacheCluster: AppFabric servisini durdurarak önbellekleme işlemine son verilmesini sağlar.

Get-Cache: Hangi sunucuda hangi verilerin önbellekleme işlemine alındığını anlık gösterir.

Şekil 4' de SUNUCU1 sunucusunun pagesettings ve menus verilerini önbelleğe aldığı görülmektedir. Bu işlemi yapabilmek için SUNUCU1 sunucusuna yani SUNUCU1 veritabanına girilerek iş talebi sayfasına giriş yapmamız yeterli olmuştur. Böylece iş talebi sayfasını ve ana ekrandaki menü bilgilerini AppFabric önbelleğe almıştır.

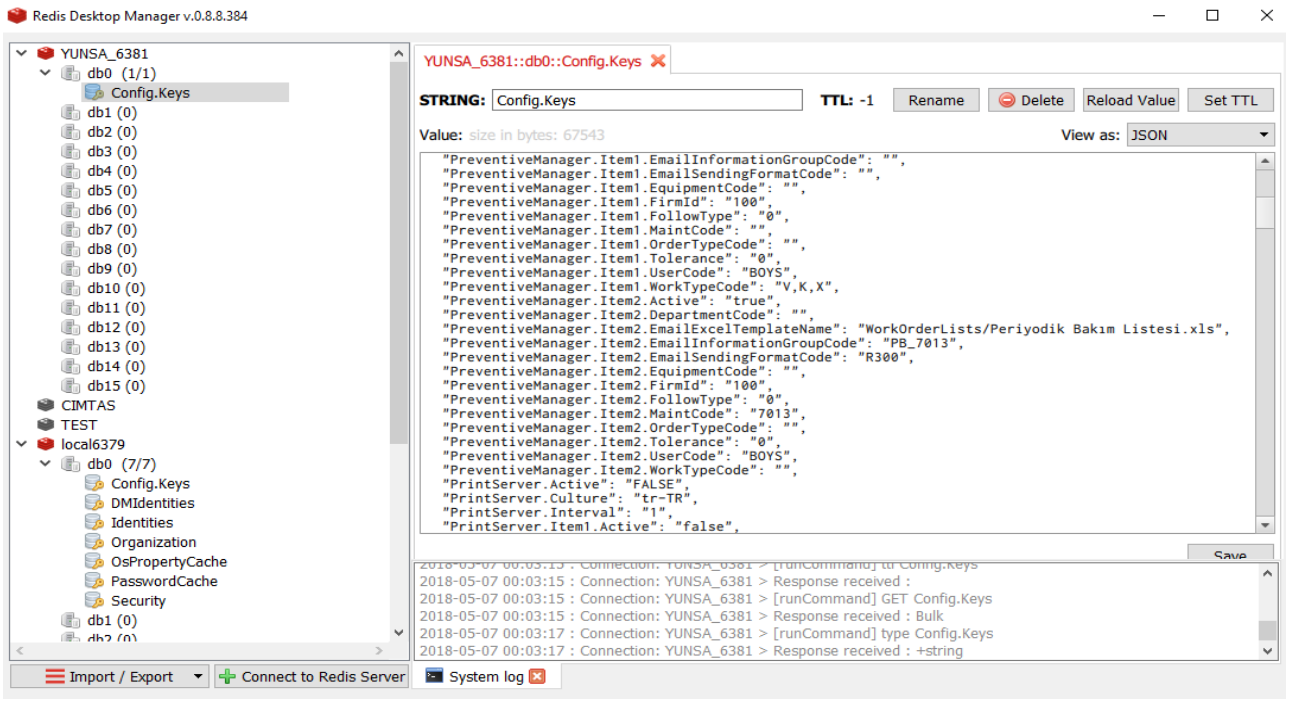
```
PS C:\Windows\system32> use-cachecluster
PS C:\Windows\system32> stop-cachecluster

HostName : CachePort Service Name            Service Status Version Info
-------------------- ------------            -------------- ------------
BoysDemo:22233       AppFabricCachingService DOWN           3 [3,3][1,3]

PS C:\Windows\system32> start-cachecluster

HostName : CachePort Service Name            Service Status Version Info
-------------------- ------------            -------------- ------------
BoysDemo:22233       AppFabricCachingService UP             3 [3,3][1,3]

PS C:\Windows\system32> get-cache

CacheName            [Host]
                     Regions
---------            ---------------
default              [BoysDemo:22233]
                     Default_Region_0361(Primary)
                     Default_Region_0019(Primary)
                     Default_Region_0942(Primary)
                     Default_Region_0640(Primary)
                     Default_Region_0297(Primary)
                     Default_Region_0989(Primary)
                     Default_Region_0918(Primary)
                     Default_Region_0869(Primary)
                     Default_Region_0678(Primary) Menus(Primary)
                     Default_Region_0028(Primary)
                     Default_Region_0123(Primary)
                     Default_Region_0775(Primary)
                     Default_Region_0075(Primary)

session
BIMSER
AGACSANAYII          [BoysDemo:22233]
                     Default_Region_0736(Primary) UI(Primary)
                     Default_Region_0918(Primary)
                     Default_Region_0905(Primary)
                     Default_Region_0775(Primary) Users(Primary)
                     Default_Region_0678(Primary) Firms(Primary)
                     GridLayouts(Primary) PageSettings(Primary)
                     Default_Region_0640(Primary) Menus(Primary)
                     Default_Region_0551(Primary)
```

Şekil 4. Boys Uygulaması AppFabric konfigurasyonu terminali

Appfabric uygulamasında hangi sunucuların önbelleklendiğini ve hangi özelliklerin önbelleğe alındığı bilgisini anlık görebilmek için şekil 4' de gösterildiği gibi Windows PowerShell uygulamasından yararlanılmaktadır.

5.3. Redis Nedir, Redis ve AppFabric Farkları

Açık kaynak kodlu bir key-value (anahtar-değer) mantığıyla çalışan bir depolama aracıdır.



Şekil 5. Redis masaüstü uygulaması önbelleğe alınan verinin gösterilmesi

AppFabric veriyi bellekte tutar ama bu veriyi dosyaya yazmaz.

Sunucu açık kaldığı sürece veriyi saklar. Sunucu kapatıldığında veriyi bellekten siler. Ancak Redis belirli aralıklara veriyi dosyalara kaydedebilir(Carlson, 2013).

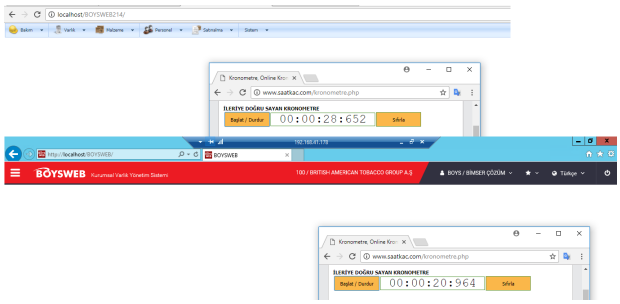
Appfabric sadece veri giriş ve çıkışı yapmaktadır. Veriyi alır sonra kullanılmak istendiğinde veriyi geri verir. Ancak Redis veri yapıları kullandığı için birçok fonksiyonu desteklemektedir.

Redis ile string(karakter kümesi), hash(Map<String, String>), list(liste), set(küme), sorted set(sıralı küme) veri tiplerinde veriyi saklayabiliriz(Yılmaz, 2016).

Redis genel olarak daha hızlı ve küçük değişkenler için kullanılır. Küme özellikleri ile daha performanslı çalışır ve daha az veri kaybı sağlar.

Şekil 5' de Redis masaüstü uygulaması açılır. Her bir sunucu için connect to redis server butonuna tıklanarak yeni bir intance name klasörü tanımlanır. Herbir cache klasörü içerisinde 15 adet db vardır. Bunlardan db0 içerisinde label list gibi localization dosyalarının verileri tutulmaktadır. Db1 de company user bilgileri tutulmaktadır. Aynı şekilde ilk 5 db dolu olarak gelmektedir.

Diğer db ler boştur doldurma işlemleri arge sırasında kod tarafından geliştirilebilir. Cache kontrolü sırasında JSON değeri seçilerek cachelenmiş verilere şekil 5' deki gibi erişilebilir. Görüldüğü gibi Redis uygulamasında yer alan 15 adet db değeri ve bu değerler içerisinde istediğimiz kadar ekleyebileceğimiz alt kırılım verileri ile AppFabric uygulamasına göre Redis uygulaması önbelleğe alınacak verilerin yönetilmesi konusunda daha kullanışlıdır. Önbellek verileri daha düzenli tutulmaktadır.



Şekil 6. Appfabric – Redis teknolojisi alt yapısı ile geliştirilen Boys uygulamasının ilk yüklenme süresi karşılaştırması

Şekil 6' da Önbellekleme ilkelerinde Appfabric teknolojisi kullanılarak geliştirilen Boys versiyonunda önbelleğe alınmış veriler silindikten sonra uygulama ilk açılış süresi 28.6 saniyedir. 28.6 saniyede appfabric uygulamasının önbelleğine arka planda istediğimiz veriler alınmakta ve sayfa çağrılarak yüklenmektedir.

Önbellekleme ilkelerinde Redis teknolojisi kullanılarak geliştirilen Boys versiyonunda önbelleğe alınmış veriler silindikten sonra uygulama ilk açılış süresi 20.9 saniyedir. 20.9 saniyede redis

uygulamasının önbelleğine istenen veriler alınmış ve sayfa yüklenmiştir.

Appfabric uygulaması içerisinde firma, menu, sayfa isimleri, sayfa ayarları, sayfa tagları tutulmaktadır. Appfabric teknolojisinin bir masaüstü programı olmadığı için Powershell uygulaması aracılığı ile tutulan verilerin sadece taglarına ulaşabiliyoruz. Asıl verilere Boys uygulamasının arka kısmında kodlama yaparken görebiliyoruz.

Redis uygulaması içerisinde firma, menu, sayfa isimleri, sayfa ayarları, sayfa tagları, açılan her sayfanın kontrol ayarları tutulmaktadır. Ayrıca Redis uygulaması içerisinde 15 adet db ismi verilen boş önbellek depoları bulunmaktadır. Tutulan bu verilere ek olarak eklenebilecek yeni kümelenmiş veriler boş db alanlarında gruplar halinde yeni isimler verilerek tutulabilir. Bu kontrolleri ve ayarları yapabilmek için Redis teknolojisinin masaüstü uygulaması mevcuttur. Masaüstü uygulaması sayesinde önbelleğe alınmış veriler Boys uygulamasının arka kısmında kodlama aşamasında görülmesi yerine direk masaüstü uygulaması

içerisinde kümelenmiş şekilde JSON vb. veri tiplerinde Şekil 8' deki gibi görüntülenebilir. Ayrıca önbelleklenmesi istenen yeni veriler yine masaüstü uygulaması sayesinde kolaylıkla önbellek teknolojisine aktarılabilir.

Redis uygulaması daha fazla önbelleğe alınmış veri ve daha düzenli grup veriler şeklinde 20.9 saniyede Boys uygulamasını yükleyebilirken, appfabric uygulaması daha az veri önbelleğe alarak, verileri kümelemeden ve geliştiriciye direk gösteremeden 28.6 saniyede Boys uygulamasını yükleyebilmiştir.

Boys uygulamasına Redis uygulaması entegre edilmesi sırasında TCache isimli bir sınıf oluşturulur. Bu sınıf üzerinden Redis ile önbelleğe alınacak verilerin gönderilmesi, alınması işlemlerini gerçekleştirecek bağlantı kurulur. TCache sınıfı önbelleğe alma işlemlerini ICacheProvider arayüz sınıfı üzerinden gerçekleştirir. TCache sınfı içerisinde önbelleğe alma, silme, listeyi önbelleğe ekleme gibi metodlar tanımlıdır. ICacheProvider arayüz sınıfı içerisinde ise sayfalar açıldıkça tanımlı metodlara geçilen parametreler aracılığı ile hangi bilgilerin önbelleğe aktarılacağı bilgilerinin yer aldığı metodlar bulunmaktadır.

TCacheProvider sınıfı:

Ek 1' de CacheProvider sınıfı türünde string değer saklayan cacheProvider isimli sözlük tanımlaması yapılır. Bu veri türünde yapılan tanımlamada çok sayıda veri sınıfı saklanabilir. Saklanan veri detayı bir anahtar ve bu anahtarlar ile eşleşen çok sayıda değer tanımıdır. Örneğin anahtar değer masa ise masanın uzunluğu, genişliği, cinsi, kaç kişilik olduğu gibi bilgilerin her biri birer değer olarak masa anahtarı içerisine saklanır. Böyle binlerce veri cacheprovider içerisinde tanımlanabilir.

Ek 2' de GetCacheProvider metodu tanımlanır. Bu metod içerisinde sunucuya(veritabanına) göre verilerin önbelleğe atılması işlemi gerçekleştirilir.

Ek 3' de Remove metodu ile önbelleğe alınan verinin silinmesi işlemi gerçekleştirilir. Silme işlemi yapılırken hangi database için hangi anahtara göre silme işleminin gerçekleştirileceği belirtilir.

Ek 4' de Get metodu ile önbellekteki verinin kullanılmak üzere çağırılması sağlanır. Hangi veritabanından hangi anahtar çağrılacağı bilgisi verilir.

Ek 5' de Put metodu ile verilerin önbelleğe eklenmesi işlemi gerçekleştirilir. Ekleme işlemi sırasında veritabanı ve anahtar bilgisi verilir.

ICacheProvider arayüz sınıfı:

Bu sınıf üzerinden hangi verilerin ekleneceği tanımlanır.

Ek 6' da RemoveMaterialGroupsCache metodu ile ITEMGROUPS tablosu önbellekten silinir.

Ek 7' de GetMaterialGroups metodu ile ITEMGROUPS tablosu önbellekten alınarak sistemden çağırılır. Verinin önbellekten alınarak sayfalarda kullanılması için gereklidir.

Ek 8' de LoadAccountsToCacheAsync metodu ile sisteme giriş yapan kullanıcı bilgileri ACCOUNTS tablosu üzerinden çekilerek önbelleğe alınmaktadır.

Boys uygulamasına AppFabric uygulaması entegre edilirken TCache sınıfı oluşturulur. Bu sınıf üzerinden AppFabric uygulamasına önbelleklenecek veriler gönderilir.

TCache sınıfı:

Ek 9' da CacheService servisi açılır.

Ek 10' da GetUserParamValue metodu ile önbelleğe alınmış login olan kullanıcı bilgilerinin sistemde kullanılmak üzere çağrılması sağlanır.

Ek 11' de GetCompanyParamValue metodu ile önbelleğe alınmış login olan işletme bilgileri sistemde kullanılmak için çağırılır.

Redis uygulamasındaki gibi arayüz sınıfı mevcut değildir. Önbelleğe alınacak veriler CacheService servisi aracılığı ile AppFabric uygulamasına gönderilmektedir. Arayüz sınıfı olmadığı için verilere müdahale edilme imkanı düşüktür. Veriler servise gönderilir ve beklenir. Kütüphane mantığı yoktur. Anahtar veri değeri ilişkisi yer almaz. Bu yüzden Redis uygulamasındaki gibi çok sayıda veri gruplanarak önbellekte tutulamaz. Veritabanı tablolarının isimlendirilerek önbellekte tutulması durumu söz konusu değildir. Daha basit ve az sayıda veri önbellekte tutulabilmektedir.

Tablo 1. Boyut ve Veri Türüne Göre AppFabric ve Redis Performansları

BOYUT	METOD	TÜR	APPFABRİC	REDİS
4 Byte	Set	String	918 ms	240 ms
4 Byte	Get	String	587 ms	137 ms
115 KB	Set	Array	11 ms	15 ms
115 KB	Get	Array	11 ms	7.1 ms

Tablo 1' de Boys uygulamasına bağlanan AppFabric ve Redis uygulaması için kullanıcı sisteme giriş anında string türde alınan ve gönderilen değişkenin AppFabric ve Redis uygulamalarından kaç saniyede yüklendiği ve geldiği bilgileri yer almaktadır. Bu test önce string türünde bir değişken ile daha sonra dizi türünde bir veri kümesi ile gerçekleştirildi. Sisteme giriş esnasında kullanıcının tüm bilgilerinin çekilmesi gibi büyük boyutlu veriler ile işlem yapıldığında aradaki fark artmaktadır. Test sonucunda aynı şartlarda Redis uygulamasının AppFabric uygulamasına göre daha performanslı olduğu belirlenmiştir.

6. Sonuç ve Tartışma

Boys uygulamasında önbellekleme işlemini gerçekleştirebilmek için güncel ve popüler olarak kullanılan iki uygulama, Boys altyapısına eklenip konfigure edilerek test edilmiştir. AppFabric ve Redis uygulamalarından hangisinin daha az maliyetli olacağı, hangisinin daha hızlı ve programı yormayacak şekilde önbellekleme işlemini gerçekleştirebileceğini test edebilmek için önce AppFabric uygulaması Boys ugulamasına dahil edilmiş hız ve kabiliyetleri gözden geçirilmiş daha sonra Redis uygulaması Boys uygulamasına dahil edilmiş, hız ve kabiliyetleri gözlemlenmiştir. Appfabric ve Redis uygulamaları için iki uygulamanında önbelleğe alınmış verilerinin listesi çekildiğinde Redis uygulamasında önbelleğe alınan veri daha düzenli ve daha okunabilir durumdadır. Redis uygulaması daha çok ve daha küçük ölçekli verileri önbellekleyebilmekte v bu verileri kümeleyerek düzenli halde önbellekte saklayabilmektedir. Ayrıca istendiği takdirde Redis uygulaması önbelleğe aldığı verileri dosyaya yazabilmektedir. Böylece sunucu kapatıldığında dahi geliştiricinin kaybolmasını istemediği veriler dosyalarda saklanarak sunucu açıldığında tekrar kullanılabilir hale getirilmektedir. Bununla birlikte redis uygulaması appfabric uygulamasına oranla verileri daha hızlı önbelleğe almakta ve daha hızlı uygulamaya tekrar geri dönmektedir.

Teşekkür

Düzce Üniversitesi Teknoloji Fakültesi Bilgisayar Mühendisliği Bölümü'ne ve Kocaeli Üniversitesi Teknoparkı Bimser Çözüm A.Ş. firmasına katkılarından ve yardımlarından dolayı teşekkür ederiz.

Conflict of Interest / Çıkar Çatışması

Yazarlar tarafından herhangi bir çıkar çatışması beyan edilmemiştir.

No conflict of interest was declared by the authors.

Kaynaklar

Marquez A.C., 2007. The Maintenance Management Framework, Seville, Spain, Springer Mobley.

Taşın M.F., 2006. Determination of optimal stock level with fuzzy logic method under preventive maintenance policy, MSc Thesis , Sakarya Universty, Sakarya,Turkey.

Bal A., 2013. RFID- Asisted Maintenance Management for Production Facilities, MSc Thesis, İ.T.Ü., İstanbul.

Doğan S., 2014. Computer Aided Maintenance Management System Kardemir Rail And Profile Rolling Mill Applicability, MSc Thesis, Karabük University, Karabük,.

Ayrancı M.M., 1997. Computer-aided Maintenance Methods and Maintenance Management in Vessels, İ.T.Ü., İstanbul.

Chaneski W.S., 2002. Total Productive Maintenance An Effective Technique, 75, 46-47, Modern Machine Shop.

İ.T.Ü. Department of Information Technologies, Service Oriented Architecture (SOA), http://bidb.itu.edu.tr/seyirdefteri/blog/2013/09/06/servis-y%C3%B6nelimli-mimari-(service-oriented-architecture-soa (15.01.2017)

Acar G., 2014. Preventive Maintenance Methods And Applications Used In Automotive Industry, MSc Thesis, Kocaeli Üniversity, Kocaeli, Turkey.

Küçük B.B., 2016. Determination Of Sectoral Background Oriented to Total Productive Maintenance In Forest Products Industry, The Example Of Inegöl, Düzce Üniversity, Düzce, Turkey.

Fatih Boy, Appfabric Önbellek Mimarisi, http://www.csharpnedir.com/articles/read/?id=1066, (15.02.2017).

Şenferah O.H., 2012. Designing and Implementing Interfaces for Master Student Pre-Admission Automation System, Dumlupınar University, Kütahya, Turkey.

Akgündüz M.H., 2015. Distributed Architecture Desingn And Management System Development, İ.T.Ü, İstanbul, Turkey.

Kılıçaslan M., 2013. An Example Application For Online Video Transfer Useable In E-lesson Contents, Süleyman Demirel University, Isparta, Turkey.

Lowy J., 2015. Programmig WCF Services: Mastering WCF and the Azure AppFabric Service Bus, O'Reilly Media, Web.

Carlson J., 2013. Redis In Action, Mannig Publications, Web.

Yılmaz E.C., 2016. "Web Application Development With Docker" , Journal of Global Engineering Studies, 3(2), 12-14.

Ekler

Ek 1.
```
static Dictionary<string, CacheProvider> cacheProvider = new Dictionary<string, eBACache.CacheProvider>();
```

Ek 2.
```
static CacheProvider GetCacheProvider(string instanceName)
{
if(!cacheProvider.ContainsKey(instanceName))
{
lock (locker)
{
if(!cacheProvider.ContainsKey(instanceName))
{
var provider =TBoysHelper.GetCacheProvider(instanceName);
if (provider != null)
{ provider.Connect();
cacheProvider.Add(instanceName, provider);
}
return provider;
}
}
}
return cacheProvider[instanceName];
}
```

Ek 3.
```
public static void Remove(string instanceName, int databaseId, string key)
{

GetCacheProvider(instanceName).Databases[databaseId].Remove(key);
}
```

Ek 4.
```
public static T Get<T>(string instanceName, int databaseId, string key)
{
return GetCacheProvider(instanceName).Databases[databaseId].Get<T>(key);
}
```

Ek 5.
```
public static void Put<T>(string instanceName, int databaseId, string key, object value)
{
GetCacheProvider(instanceName).Databases[databaseId].Add(key, ObjectToJson<T>(value));
}
```

Ek 6.
```
public static void RemoveMaterialGroupsCache()
{

Boys.Cache.CacheProvider.Current.Remove("ITEMGROUPS", Boys.Cache.eCacheRegion.DataTable);
}
```

Ek 7.
```
public static DataTable GetMaterialGroups()
{
DataTable result = null;

result = Boys.Cache.CacheProvider.Current.Get<DataTable>("ITEMGROUPS", Boys.Cache.eCacheRegion.DataTable, () =>
{
return Bimser.Boysweb.Data.TDataHelper.Current.GetMaterialGroups(new eBADB.Filter.Criteria());
});
return result;
}
```

Ek 8.
```
public static async void LoadAccountsToCacheAsync()
{
await Task.Factory.StartNew(() =>
{
CacheProvider.Current.Set("ACCOUNTS", Bimser.Boysweb.Data.TDataHelper.Current.GetAccounts(), eCacheRegion.DataTable);
});
}
```

Ek 9.
```
private CacheServiceClient cacheService = null;
```

CacheService servisi tanımlanır.

```
public static CacheServiceClient
GetCacheServiceClient()
{
CacheServiceClient result = new
CacheServiceClient();
result.Open();
return result;
}
```

Ek 10.
```
public int GetUserParamValue(string
paramStr)
{
if (IsCacheModeEnable())
return CacheService.GetUserParamValue(UserId,
paramStr, LicDesc);
else
return
BoyswebPage.AppSession.User.GetUserRightValue(p
aramStr);
}
```

Ek 11.
```
public int GetCompanyParamValue(string
paramStr)
{
if (IsCacheModeEnable())
return
CacheService.GetCompanyParamValue(CompanyId,
paramStr, LicDesc);
else
return
BoyswebPage.AppSession.Company.GetParamValue(
paramStr); }
```