ÜNİVERSİTE BİLGİ SİSTEMLERİNDE
SERVİS ODAKLI MİMARİ (SOA) KULLANARAK
ONLINE ÖDEME SİSTEMİ TASARIMI

Erkan BAYRAM

Yüksek Lisans Tezi
Bilgisayar Mühendisliği Anabilim Dalı
Doç. Dr. Tevhit KARACALI
2016

ATATÜRK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

ÜNİVERSİTE BİLGİ SİSTEMLERİNDE SERVİS ODAKLI MİMARİ (SOA) KULLANARAK ONLINE ÖDEME SİSTEMİ TASARIMI

Erkan BAYRAM

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

ERZURUM
2016

T.C.
ATATÜRK ÜNİVERSİTESİ
Fen Bilimleri Enstitüsü Müdürlüğü

TEZ ONAY FORMU

ÜNİVERSİTE BİLGİ SİSTEMLERİNDE SERVİS ODAKLI MİMARİ KULLANARAK ONLINE ÖDEME SİSTEMİ TASARLANMASI

Doç.Dr. Tevhit KARACALI danışmanlığında, **Erkan BAYRAM** tarafından hazırlanan bu çalışma, 23/06/2016 tarihinde aşağıdaki jüri tarafından **Bilgisayar Mühendisliği Anabilim Dalı Bilgisayar Mühendisliği** Bilim Dalı'nda yüksek lisans tezi olarak **oybirliği / ~~oy çokluğu~~ (.../...)** ile kabul edilmiştir.

Başkan: Doç. Dr. Tevhit KARACALI İmza :

Üye : Yard. Doç. Dr. Tolga AYDIN İmza :

Üye : Yard. Doç. Dr. Mustafa AĞAOĞLU İmza :

Yukarıdaki sonuç;

Enstitü Yönetim Kurulu'nun 30.06/2016 tarih ve ...27.../...36..... nolu kararı ile onaylanmıştır.

Prof. Dr. Ertan YILDIRIM
Enstitü Müdürü

Not: Bu tezde kullanılan özgün ve başka kaynaklardan yapılan bildiriş, çizelge, şekil ve fotoğrafların kaynak olarak kullanımı, 5846 sayılı Fikir ve Sanat Eserleri Kanunundaki hükümlere tabidir.

ÖZET

Yüksek Lisans Tezi

ÜNİVERSİTE BİLGİ SİSTEMLERİNDE SERVİS ODAKLI MİMARİ (SOA) KULLANARAK ONLINE ÖDEME SİSTEMİ TASARIMI

Erkan BAYRAM

Atatürk Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Tevhit KARACALI

Servislerin etkin bir şekilde koordine edilip kullanılabilmesi amacıyla bir araya getirilmesi ile oluşan servis odaklı mimarinin (SOA) kullanımı son zamanlarda giderek artmaktadır. Bir bilgi işlem stratejisi olan SOA temel yapı taşları birbirlerine gevşek bağlı olan servislerdir. Fonksiyonların servisler halinde sunulmasını benimseyen bir yaklaşım olan SOA, farklı platformlarda yazılmış olan kurumsal uygulamalar arasında entegrasyon sürecinde ortaya çıkan maliyet, süre, hız gibi problemlere çözümler sunmaktadır. Bankacılık sistemlerinde, üniversite bilgi sistemlerinde, online ödeme sistemlerinde, sağlık sektöründe ve esnekliğe, güvenliğe ve hıza ihtiyaç duyulan teknolojinin olduğu her yerde SOA kullanımı karşımıza çıkmaktadır.

Bu tezde SOA yaklaşımından faydalanılarak üniversite bilgi sistemlerinde online olarak harç, materyal ücreti, kart ücreti vb. ödemelerin alınabilmesini sağlayan sistemin tasarlanması amaçlanmıştır. Saniyeler içerisinde kişilerin bankalara, ATM'lere gitmeden oturdukları yerden ödemelerini yapabilmeleri ve anında ödemelerin ilgili sistemlere aktarılabilmesi günümüz hız çağında yaşayan insanlar için zamandan büyük tasarruf sağlayacak ve kişilerin, kurumların işlerini kolaylaştıracaktır.

2016, 93 sayfa

Anahtar Kelimeler: Kurumsal servis yolu, online ödeme sistemi, servis odaklı mimari, üniversite bilgi sistemleri, web servisleri

ABSTRACT

MS Thesis

THE DESIGN OF AN ONLINE PAYMENT SYSTEM USING SERVICE ORIENTED ARCHITECTURE IN UNIVERSTY INFORMATION SYSTEMS

Erkan BAYRAM

Atatürk University
Graduate School of Natural and Applied Sciences
Department of Computer Engineering

Supervisor: Assoc. Prof. Dr. Tevhit KARACALI

The use of service oriented architecture (SOA) consisting of different services has increased in an accelerated rate in recent years, which aims to use such services in a coordinated manner effectively. The core structure of SOA, which is an information processing strategy, is the services that are loose coupled to each other. SOA is an approach internalizing different functions as services provides solutions for cost, time and speed problems that are seen on the integration process of different corporate applications being coded different platforms. It is inevitable to encounter SOA practices on the technologies requiring flexibility, security and speed such as banking systems, university information systems, online banking systems, online health services.

In this thesis, it is aimed to provide the design of online services including tuition, material, and card fees of a public university information system build on SOA approach. The opportunity of paying fees on a seat within seconds without going banks and ATMs and the opportunity of transferring such payments to the related systems within seconds help people living in information age to save their time and thereby make individuals' and organizations' works easier.

2016, 93 pages

Keywords: Enterprise service bus, service oriented architecture, online payment system, university information systems, web services

TEŞEKKÜR

Bu tezin konusunun belirlenmesinde ve yazım aşamalarında danışmanlık yaparak yol gösteren danışman hocam Sayın Doç. Dr. Tevhit KARACALI başta olmak üzere, çalışmalarım sırasında ve programın yazılım, test süreçlerinde ve verilerin temini konusunda bana her türlü desteği veren Bilgisayar Bilimleri Araştırma ve Uygulama Merkezi (BAUM) müdürü Sayın Prof. Dr. Muhammet Dursun KAYA ve BAUM müdür yardımcısı Sayın Yrd. Doç. Dr. Serdar AYDIN'a ve tezin uygulamasının oluşturulma aşamasında desteğini esirgemeyen değerli arkadaşım Sayın Arş. Gör. Ferhat BOZKURT'a en içten teşekkürlerimi sunuyorum.

Ayrıca tez çalışması süresince manevi desteklerini benden esirgemeyen eşime ve aileme teşekkürlerimi sunarım.

Erkan BAYRAM
Haziran 2016

İÇİNDEKİLER

ÖZET i
ABSTRACT ii
TEŞEKKÜR iii
SİMGELER ve KISALTMALAR DİZİNİ vi
ŞEKİLLER DİZİNİ vii
ÇİZELGELER DİZİNİ x
1. GİRİŞ 1
2. KAYNAK ÖZETLERİ 4
2.1. Servis odaklı mimarinin tarihçesi 4
2.2. Servis odaklı mimari (*service oriented architecture*) 4
2.2.1. Servis odaklı mimaride esneklik 6
2.2.2. Servis odaklı mimari ve dağıtık sistemler 7
2.2.3. Servisler 8
2.2.4. Gevşek bağlılık (loose coupling) 9
2.2.5. Birlikte çalışabilirlik (interoperability) 10
2.3. Servis odaklı mimari temel bileşenleri 10
2.3.1. Kurumsal servis yolu (*enterprise service bus*) 10
2.3.2. Mimari kavramı (*architecture*) 13
2.3.3. Süreçler 14
2.3.4. Yönetim 14
2.4. Web servisler 15
2.4.1 XML (*Extensible Markup Language*) 18
2.4.2. SOAP (*Simple Object Access Protocol*) 19
2.4.2.a. SOAP zarf özellikleri 21
2.4.2.b. Veri çözümleme kuralları 22
2.4.2.c. RPT sözleşmeleri 22
2.4.3. UDDI (*Universal Description, Discovery, and Integration*) 23
2.4.4. WSDL (*Web Services Description Language*) 24
2.5. Web servislerinde güvenlik standartları 26

2.6. Servis odaklı mimari dezavantajları 27
2.7. Servis odaklı mimari avantajları 28
2.8. Servis odaklı mimaride güvenlik 30
2.8.1. Servis odaklı mimaride güvenlik standartları 31
2.9. Servis odaklı mimarinin katmanları 33
2.10. WCF (*Windows Communication Foundation*) 35
2.10.1. WCF mimarisi 38
3. MATERYAL ve YÖNTEM 41
3.1. Uygulamanın geliştirildiği platform ve programlama dilleri 41
3.2. Projenin tanıtımı 41
3.3. WCF teknolojisiyle geliştirilmiş ödeme servisi 42
3.3.1. Ödeme servisindeki metotlar 43
3.3.2. Ödeme servisinde entity model 46
3.3.3. Servis yetki kontrolü 47
3.3.4. Serviste bulunan sınıflar (class) 47
3.4. Windows uygulaması 53
3.5. Veri modeli 65
3.5.1. Veri tabanında bulunan tablolar 67
3.5.2. Veri tabanında bulunan saklı yordamlar (*stored procedure*) 71
4. ARAŞTIRMA BULGULARI ve TARTIŞMA 85
4.1. WCF Servis İstatistikleri 85
4.2. Projenin tamamlanmasının getirdiği yararlar 87
5. SONUÇ ve ÖNERİLER 88
KAYNAKLAR 91
ÖZGEÇMİŞ 94

SİMGELER ve KISALTMALAR DİZİNİ

BT	Bilgisayar teknolojileri
DS	Dağıtık sistemler
HU	Heterojen uygulamalar
KSY	Kurumsal servis yolu
KY	Kurumsal yönetim
SOA	Servis odaklı mimari
SP	Saklı yordam
SS	Servis sağlayıcı
ST	Servis tüketici
ÜBS	Üniversite bilgi sistemleri
VM	Veri Modeli
WCF	Windows communication foundation
WS	Web servisler
WU	Windows Uygulaması

ŞEKİLLER DİZİNİ

Şekil 2.1. Noktadan-Noktaya ve kurumsal servis yolu yaklaşımı 7
Şekil 2.2. Kurumsal servis yolunun üç ana bileşeninin gösterimi 11
Şekil 2.3. Noktadan-noktaya gösterimi 12
Şekil 2.4. Mesaj aracısı gösterimi 12
Şekil 2.5. Kurumsal servis yolu mimarisi gösterimi 13
Şekil 2.6. Web servis Üretici-Tüketici-Sağlayıcı ilişkisi gösterimi 16
Şekil 2.7. SOAP mesaj yapısı iskeleti gösterimi 20
Şekil 2.8. Örnek bir SOAP zarf elemanı gösterimi 21
Şekil 2.9. İş analistleri ve teknoloji arasındaki ilişkinin gösterimi 24
Şekil 2.10. WSDL belge yapısı gösterimi 25
Şekil 2.11. Güvenlik standartlarının uygulanması 32
Şekil 2.12. Servis odaklı mimari katmanları gösterimi 33
Şekil 2.13. WCF mimarisinin gösterilmesi 38
Şekil 3.1. WCF ödeme servisinin genel görünümü 41
Şekil 3.2. BorcSorgula () metodu 43
Şekil 3.3. BorcOdeme () metodu 43
Şekil 3.4. IptalSorgu () metodu 44
Şekil 3.5. OdemeIptal () metodu 44
Şekil 3.6. MutabakatOnay () metodu 45
Şekil 3.7. MutabakatListe () metodu 45
Şekil 3.8. Ödeme servisi entity modeli gösterimi 46
Şekil 3.9. Kullanıcı adı ve şifre kontrolü yapan kodun gösterimi 47
Şekil 3.10. ogrBorc () sınıfı 47
Şekil 3.11. OgrBorcOdemeTalep () sınıfı 48
Şekil 3.12. OgrBorcOdeme () sınıfı 48
Şekil 3.13. OgrIptalSorguTalep () sınıfı 49
Şekil 3.14. OgrOIptalSorguYanit () sınıfı 49
Şekil 3.15. OgrOdemeIptalTalep () sınıfı 50
Şekil 3.16. OgrOdemeIptalYanit () sınıfı 50

Şekil 3.17. MutabakatOnayTalep () sınıfı.....51
Şekil 3.18. MutabakatOnayYanit () sınıfı.....51
Şekil 3.19. MutabakatListeTalep () sınıfı.....52
Şekil 3.20. MutabakatListeYanit () sınıfı.....52
Şekil 3.21. Uygulama arayüzü için login ekranı.....53
Şekil 3.22. Borç sorgulama - ödeme gerçekleştirme - ödeme iptal ekranı.....54
Şekil 3.23. Borçların listelenmesi.....54
Şekil 3.24. Borç sorgu yapan kod kısmı.....55
Şekil 3.25. Tahsilat alma durumu.....56
Şekil 3.26. Ödemeyi gerçekleştiren kod kısmı.....57
Şekil 3.27. İptal sorgu sonucu ekran görüntüsü.....58
Şekil 3.28. İptal sorgusunu yapan kod parçacığı.....59
Şekil 3.29. İptal işlemini yapan kod parçası.....60
Şekil 3.30. Mutabakat sorgusunu yapan kod parçası.....62
Şekil 3.31. Mutabakat sorgulama.....63
Şekil 3.32. Mutabakatın sağlanamaması durumu.....63
Şekil 3.33. Mutabakat listesini çeken kod parçası.....64
Şekil 3.34. Veri tabanında yer alan tablolar ve ilişkileri.....66
Şekil 3.35. Borç sorgularınının tablolara loglanması.....67
Şekil 3.36. Öğrenci birim bilgileri.....67
Şekil 3.37. Borçların kaydedildiği tablo.....68
Şekil 3.38. Birime göre ödeme miktarlarının tutulduğu tablo.....69
Şekil 3.39. Borçların son ödeme tarihlerinin tutuldugu tablo.....69
Şekil 3.40. Ceza oranları bilgilerinin tutulduğu tablo.....70
Şekil 3.41. Servis hata kodları ve açıklamalarının tutulduğu tablo.....71
Şekil 3.42. Son ödeme tarihiyle aynı yılda yapılan ödemeye gecikme cezası hesabı.....72
Şekil 3.43. Son ödeme tarihinden farklı yıllardaki ödemelere gecikme cezası hesabı.....73
Şekil 3.44. Hesaplanan Toplam miktarın return edilmesi.....73
Şekil 3.45. Hangi ödemenin hangi hesaba yatacağının belirlenmesi.....74
Şekil 3.46. Tüm borç sorgularının loglanması.....75
Şekil 3.47. Uzaktan eğitim öğrencisinin borçlu olduğu dönemlerin hesabı.....76
Şekil 3.48. Uzaktan eğitim öğrencisinin borçlarının tabloya eklenmesi.....77

Şekil 3.49. Açıköğretim öğrencilerinin ödemesi gereken dönem ve borç bilgileri 78
Şekil 3.50. Açık öğretim temp tablolarının doldurulması 78
Şekil 3.51. Açıköğretim öğrencilerinin borçlarının bireyborclar'a yazılması 79
Şekil 3.52. Yeni kayıt lisans-önlisans öğrencisinin güz dönemi borçlandırılması 80
Şekil 3.53. Önlisans-lisans ara sınıf öğrencilerinin güz dönemi borçlandırılması 81
Şekil 3.54. Önlisans-lisans öğrenci borçlarının bireyborclar tablosuna eklenmesi 82
Şekil 3.55. Yaz okulu seçilen derslerin ücretlerinin bireyborclara eklenmesi.....83
Şekil 3.56. Yaz okulu ders ücretinin hesaplanması 84
Şekil 4.1. Üç haftalık süreçte borç sorguya dönülen cevapların istatistiği 85
Şekil 4.2. Üç haftalık ödeme sorguya dönülen cevapların istatistiği.....86

ÇİZELGELER DİZİNİ

Çizelge 2.1. Sıkı bağlılık ile gevşek bağlılığın karşılaştırılması.....9
Çizelge 2.2. WCF'deki bağlayıcı tipleri 37

1. GİRİŞ

Hızlı ve esnek bir sistem yapısının oluşturulmasını amaçlayan servis odaklı mimari (SOA) kullanımı son zamanlarla giderek artmaktadır. İlk olarak 1994 yılında ortaya atılan servis odaklı mimari terimi Microsoft tarafından web servislerin geliştirilmesiyle büyük bir ivme kazanmış ve gelişim göstermiştir (Josuttis 2007). En temel yapı taşları servisler olan SOA'nın içerisinde bulundurduğu yazılım bileşenlerinin yeniden kullanılabilir olması, birlikte çalışılabilirliği desteklemesi ve platformdan bağımsız olması gibi yeteneklere sahip olması oldukça önemlidir.

Günümüzde kamu ve büyük şirketlerin en büyük problemlerinden biri bünyelerinde bulundurdukları eski uygulamalar ile çok farklı platformlar üzerinde yeni teknolojilerle geliştirilmiş farklı uygulamaların birbirleriyle konuşturulması zorluğudur. SOA, platform bağımsız olması ve farklı programlama dillerinde yazılmış uygulamaların entegasyonlarını kolaylıkla gerçekleştirebilmesi nedeniyle bu zorluklara büyük çözümler getirmiştir.

SOA ile kurumların kendi içlerindeki iş süreçleri standart bir temele oturtulmuştur. Entegrasyon ve diğer süreçler hızlandırılmıştır. Bu da süreçlerin daha yönetilebilir olmasını sağlamıştır. SOA'nın esneklik ve gevşek bağlılık yetenekleri ile kurumlar bir teknolojiyi kullanmayı bırakıp, yeni bir teknolojiye geçmek istediklerinde zorluklar yaşamayacaklardır. Geçiş süreçleri kolay ve kısa sürede tamamlanacak ve var olan işleyişler etkilenmeyecektir.

Günümüz hız çağı bireyler için zaman ve hız kavramı çok önem kazanmıştır. Artık kişiler için bankalarda sıra beklemek, alışveriş merkezlerine gidip alışveriş yapmak, vergilerini ödemek için vergi dairelerine gitmek büyük zaman kaybı olarak görülmektedir. Her ne kadar, çok mesafeler alsalar da kamu kurumları ve özel kuruluşların insanların tüm ihtiyaçlarını çevrimiçi (online) olarak karşılayabilecek seviyeye ulaşamadıkları günlük hayatta gözlemlenmektedir.

Bu çalışmada SOA kullanılarak, öğrencilerin üniversiteleriyle ilgili katkı payı, materyal ücreti, kart parası gibi ödemelerini internet bankacılığı, ATM'ler ve şubeler üzerinden gerçekleştirebilmesi amaçlanmıştır.

Bu çalışmamızda öncelikli olarak literatür taraması yapılmış, SOA kavramı, temel bileşenleri, avantajları, dezavantajları, web servisleri, SOA ve web servislerin güvenliği, SOA'nın katmanları gibi konular bu bölümde tanıtılmıştır. Ayrıca Microsoft firmasının SOA yaklaşımı için geliştirdiği Windows Communication Foundation (WCF) teknolojisinden bahsedilmiş, kullanımı ve mimari yapısı anlatılmıştır. Yüksek Öğretim Kurulu Başkanlığı'nın web sitesinde yer alan ulusal tez merkezi (https://tez.yok.gov.tr/UlusalTezMerkezi/) sayfasında SOA ile ilgili daha önce yazılmış tezler incelendiğinde genel olarak SOA kavramının ne olduğu, mimarisinin hangi bileşenlerden oluştuğu, SOA ile farklı uygulamaların entegrasyonun nasıl gerçekleştirildiği, var olan eski uygulamaların SOA yaklaşımına uygun olarak yeniden nasıl tasarlandığı, iş süreçleri üzerine SOA'nın nasıl uygulanabileceği, mobil uygulama geliştirmede kullanımı gibi konularda çalışmalar yapıldığı tespit edilmiştir.

Çalışmamızın bir sonraki kısmında üniversite bilgi sistemlerine yönelik geliştirilen ödeme sistemi uygulaması tanıtılmıştır. Uygulamanın geliştirme ortamlarından bahsedilmiş ve üç temel ayağı olan WCF servisi, Windows uygulaması ve veri modeli şekillerle desteklenerek ayrıntılı olarak anlatılmıştır. WCF servisi SOA için geliştirilmiş bir teknoloji olduğu için bu çalışmamızda tercih sebebi olmuştur. WCF servisinin genel görünümü şekilsel olarak gösterildikten sonra bu serviste yer alan metotlar, entity model, sınıflar (class) gibi servisi oluşturan temel elemanlar şekillerle birlikte anlatılmıştır.

Servisin anlatılmasından sonra servisin test edilebilmesi için geliştirilmiş Windows uygulaması tanıtılmıştır. WU için geliştirilen ara yüzler şekilsel olarak gösterilmiş ve bu ara yüzlerin arkalarında yer alan kodlardan bahsedilmiştir.

Daha sonra projenin üçüncü ayağı olan VM anlatılmıştır. VM'yi oluşturan SP'ler, tablolar gibi bileşenler şekillerle desteklenerek gösterilmiştir.

Daha sonraki bölüm olan araştırma bulguları kısmında WCF servisine gelen isteklerin üç haftalık süreç için istatistiksel bilgiler grafikler yardımıyla belirtilmiştir. Ayrıca projenin tamamlanmasının getirdiği faydalar ayrıntılı bir şekilde anlatılmıştır.

Bu çalışmanın daha da geliştirilmesiyle Yüksek Öğretim Kurulu'na bağlı tüm üniversitelerde öğrencilerin ödemeleri gereken ücretler YÖK ya da Maliye Bakanlığı'na bağlı tek bir merkezden denetlenebilir duruma getirilebilir. Her üniversiteye ilgili bankalardan gelen ödeme bilgileri aynı zamanda ortak bir veri havuzuna da gönderilecek ve bu bilgilere ihtiyaç duyan tüm kurumlar üniversitelere resmi yazı yazmadan, belirli süreçleri ve prosedürleri beklemeden saniyeler içerisinde verilere ulaşabilecektir.

2. KAYNAK ÖZETLERİ

2.1. Servis Odaklı Mimarinin Tarihçesi

Garthner'ın eski analistlerinden olan Alexander Pasik 1994 yılında SOA (Service Oriented Architecture) kelimesini türetti. Daha sonra yine Garthner'ın analistlerinden olan Roy W. Schulte and Yefim V. Natis SOA hakkında ilk raporu 1996 yılında yayınladılar. SOA'ya asıl ivme kazandıran olay Microsoft tarafından web servislerinin 2000 yılında bulunmasıdır. Daha sonra IBM, Oracle, HP, SAP ve Sun gibi diğer şirketler de bu işin içine girdiler (Josuttis 2007).

2.2. Servis Odaklı Mimari (*Service Oriented Architecture*)

Servis Odaklı Mimari hızlı ve esnek bir sistem yapısını amaçlayan, servislerin bir araya gelmesiyle oluşturulan bir mimaridir. Bu mimari satın alınabilecek hazır bir uygulama değildir. SOA'yı bir yaklaşım olarak düşünebiliriz. Şirketler veya kurumlar bu mimariyi kendi iş süreçlerine göre tasarlamak zorundadır. Sistemler büyüyebilir, daha karmaşık hale gelebilir, SOA sistemlerin her durumda esnek kalmasını ve hızlı olmasını sağlayacaktır. Farklı platformlar üzerinde yazılmış programlar, otomasyonlar SOA yardımıyla konuşabilir, veri alışverişinde bulunabilirler. Birbirinden farklı kurumlar birlikte iş yapmak için bir araya geldiklerinde birbirlerinin sistemlerine, uygulamalarına, bilgilerine ihtiyaç duyacaklardır. Bu durumda ya sıfırdan hepsinin ortak kullanacağı bir uygulama yazacaklar ve eski verilerini de bu uygulamada kullanmak üzere aktaracaklar ya da SOA ile heterejon yapıdaki farklı sistemlerini konuşturacaklar. SOA ile kurumların yeni bir sistemle entegrasyonları çok kolay gerçekleştirilebilir. Eski programlar, otomasyonlar çöpe atılmadan entegrasyon sağlandığı için kurumların maliyetleri artmayacak ve ekonomik olarak tasarruf etmeleri sağlanacaktır. Bu mimarinin temel taşları olarak servisleri, kurumsal servis yolunu (Enterprise Service Bus) ve her sistemin kendine has politika ve süreçlerini sayabiliriz. Bu mimarinin temel taşlarından olan servisler tekrar terkrar çağrılabilir ve kullanılabilirler. SOA birden fazla

servisin bir araya gelip uyum içerisinde çalışabilmesini amaçlayan bir yaklaşım olarak da tanımlanabilir. SOA ile dağıtık uygulamalar basitleştirilir. Bu mimari ile kurumun içerisindeki iş ihtiyaçlarını karşılayan farklı uygulamalar birbirleriyle entegre edilip konuşturulabilmektedir.

Kullanıcıların cihazlarla iletişimde bulunma zorunluluğunun yanı sıra, cihazlar da kendi aralarında işbirliği yapmak mecburiyetindedirler. Dağıtık sistemler üzerinde yazılım geliştirmede ara yüzlerin çok olması problemiyle baş edebilmek için, SOA ümit verici bir teknolojidir (Dönmez *et al.* 2010).

Nesne yönelimli yazılım mimarisi karmaşık ve yeniden kullanılabilir yazılım oluşturmak için hiyerarşik olarak yapılandırılmıştır. En düşük düzeyde işlevler bir nesne için kapsüle edilmiştir. Bir dizi etkileşimli yazılım nesneleri bir bileşen halinde toplanır. SOA ise dağıtık sistemler için bu hiyerarşik yapıyı genişletir ve bileşenlerin bir koleksiyonu olarak servisler tanımlanır (Gehlen and Pham 2005).

Şimdilerde SOA'nın tekdüze bir tanımı yoktur. Daha kapsamlı tanımı; servislerin bir araya toplanmasıdır. Servisler arasındaki iletişimde basit bir veri transferi için iki veya daha fazla koordineli servis aktif edilebilir. SOA, gevşek bağlı servisler ve yazılım mimarisinin dolaylı adresleme mimarisi ile servis hesaplama kaynaklarına dayalı bir organizasyondur. Temel olarak SOA, tasarım ve gevşek yazılım çözümleri yaklaşımı oluşturmak içindir (Zamani and Gupta 2015).

Yapı taşları servisler olan SOA, teknoloji perspektifinden sadece servisleri kapsamaz aynı zamanda politikaları, servis sunumu ve tüketimini yöneten uygulamaları da içerir. SOA, yazılım bileşenlerinin yeniden kullanabilirliğini ve protokol bağımsızlığını sağlar, uygulama entegrasyonunu kolaylaştırır. Modüler tasarım, soyutlama ve kapsülleme gibi temel yazılım mimarisi prensiplerini uygular. XML ve SOAP gibi açık standartlar ile farklı platformlar üzerinde farklı dillerde yazılmış uygulamalar arasında birlikte çalışabilirliği sağlar (Kart *et al*. 2008).

SOA son kullanıcılar ya da diğer hizmetler için istenen sonuçları elde etmek için isteğe bağlı iş ve işlem kaynaklarını birbirlerine bağlamak için bir tasarımdır. SOA, birçok benzersiz özelliklere sahip yazılım mimarisinin özel bir türüdür. Servis tasarımcıları ve geliştiricilerin WS'den en etkili şekilde yararlanabilmeleri için SOA kavramlarını anlamaları önemlidir (Valipour *et al*. 2009).

SOA, kurumsal sistemlere mimari bir yaklaşım sunmak için geçmişin bilgisayar mühendisliği yaklaşımları üzerine bir ağ üzerinde servis odaklı olarak inşa edilmiştir. Bu servis odaklı yaklaşımda servis arayüzleri ve öngörülebilir servis davranışları tanımı yer almaktadır (Raines 2009).

SOA kurumsal uygulama geliştirme için umut verici yaklaşımlardan birisidir. SOA ile geliştirilmiş farklı uygulamaların bileşenleri entegre edilebilir. İş işlevleri daha önceden tanımlanan ve platform bağımsız olan SOA aracılığıyla gerçekleştirilir. Kendini tanımlayan ve herhangi bir platformdan bağımsız olan iş ve işlevler SOA aracılığıyla çalışabilirler. SOA aynı zamanda esnek bir şekilde bu iş ve işlevler için paylaşılan bir mekanizma sağlar (Bokhari *et al*. 2015).

2.2.1. Servis odaklı mimaride esneklik

Günümüzde farklı programlama dillerinde yazılmış veya farklı işletim sistemleri üzerinde çalıştırılan uygulamaların aynı kurum içerisinde kullanılma durumu ile sıklıkla karşılaşmaktayız. Bu uygulamalar kendi içlerinde çok karmaşık bir yapıya sahip olabilirler. Bu kadar karmaşık yapıya sahip uygulamalar nasıl olacak da birbirleriyle bir uyum içerisinde konuşabilecekler? Uygulamaların hepsini yeni baştan aynı programlama diliyle mi yazacağız? Bu durumda çözüm olarak esnek bir mimari olan SOA imdadımıza yetişiyor. SOA'da farklı palatformlarda yazılmış uygulamalar, her ne kadar kendi içlerinde karmaşık bir yapıya sahip de olsalar birbirleriyle konuşturulabilir. Uygulamanın hangi dilde yazıldığı önemli değildir. SOA kullanımı ile uygulamanın yazıldığı platformlar veya verilerin hangi veri tabanlarında tutulduğu bilgisi bizim için bir problem teşkil etmeyecektir.

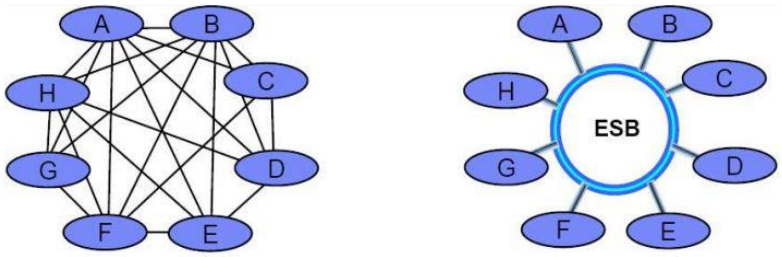
2.2.2. Servis odaklı mimari ve dağıtık sistemler

Bilgisayar ağları her yerde bulunmaktadır. Bu ağlardan bir tanesi de internettir. Cep telefonu şebekeleri, kurumsal ağlar, fabrika ağları, kampüs ağları gibi ağların hepsi ya ayrı ayrı ya da kombinasyon halinde temel özelliklerini paylaşırlar.

Ağ üzerindeki farklı bilgisayarlarda bulunan donanım veya yazılım bileşenlerinin kendi aralarında haberleşme ve koordinasyonlarının sadece mesaj göndererek sağlanabildiği sisteme dağıtık sistem denir. Bir ağ ile bağlı bilgisayarlar konumsal bir mesafe ile birbirlerinden ayrılabilir. Onlar aynı binada, aynı odada ya da ayrı kıtada olabilir (Coulouris *et al*. 2012).

DS'de kullanıcılar ağı göremezler. Ağ üzerindeki işlevlerden haberdar değillerdir. DS farklı yerlerdeki disk, yazıcı, veri tabanı, dosya vb. kaynakların paylaşımını amaçlar. DS'de SOA kullanımı ile farklı ekipler, farklı departmanlar, farklı şirketlerin farklı sistemleri yönetilebilir. Bu farklı sistemler küçük ölçekli veya büyük ölçekli olup, bu sistemler arasında uyum eksikliği olabilir. Büyük sistemler farklı platformlar üzerinde, farklı katmanlarda, farklı programlama dilleriyle oluşturulmuş olabilir. SOA ile bu sistemlerin her birisi birbirleriyle problemsiz konuşturulabilir.

SOA ile noktadan-noktaya yaklaşımın yerine kurumsal servis yolu yaklaşımı benimsenir.



Şekil 2.1. Noktadan-noktaya ve kurumsal servis yolu yaklaşımı (Beklen 2009).

Noktadan noktaya yaklaşımda bileşenler birbirleriyle iletişimlerinde API (program arayüzü) kullanırlarken, KSY yaklaşımında her bir bağlantı servis yoluna bir adaptör (servis dağıtıcı) yardımıyla bağlanır. Noktadan noktaya yaklaşımda bir bileşeni fiziksel olarak taşımak için etkileşimde olduğu her noktanın değiştirilmesi gerekli iken KSY'de bileşenler uygulamalara etki etmeden taşınabilmektedir (Beklen 2009).

2.2.3. Servisler

Servis terimi web hizmetlerinden iş süreçlerine kadar her şeyi tanımlamak için kullanılmıştır. İş fonksiyonlarının ne iş yaptığının ya da neyi yapıp neyi yapamayacağının sınırlarını belirlemek için servisleri kullanmak gereklidir. Teknoloji odaklı bir yaklaşımdan ziyade iş odaklı bir yaklaşımdır. Servis yönlendirme iş işlevselliği içerisinde tanımlanan bir iş odaklı modelleme stratejisidir (Pathak 2011).

SOA'nın yapı taşlarından olan servislerin kendi kendine yetebilirlik, yeniden çağrılabilirlik, birlikte çalışabilirlik, yeniden kullanılabilirlik gibi özellikleri vardır.

Her bir servis için hizmetlerin iyi tanımlanmış ve kullanılabilir olması için sırayla belirli özelliklerini tanımlamak gerekir. Servis iyi tanımlanmış değilse tüketiciler için servisten tam olarak ne gibi değerler döndüğü, servisin ne kadar maliyetli olduğu net olmayacaktır. Bu durum servisi kullanmak isteyen kişilerde karmaşıklığa yol açacaktır. Servislerin 3 temel bileşeni vardır. Bunlar:

- Sözleşme (Contract)
- Arayüz (Interface)
- Uygulama (Implementation)

Sözleşme tüketicilerin servislerden ne beklediğine dair tahmin edilen ihtiyaçlarına göre nasıl bir servis sağlayıcı sunmak gerektiğinin belirlenmesidir. Arayüz, tüketicilerin uygulama hizmetinin gerçekleşmesi noktasında servisten nasıl faydalanacağı ve servise nasıl erişebileceğini tanımlar. Sözleşme ve arayüz dışarıya açıktır, görülebilir.

Uygulama ise tüketiciler için daha gizli bir özelliktir. Tüketiciler genellikle onun nasıl uygulandığını umursamazlar (Dikmans and Luttikhuizen 2012).

2.2.4. Gevşek bağlılık (*Loose coupling*)

Bağlılık iki ya da daha fazla uygulama arasındaki bağımlılık ilişkisidir. Uygulamalar ya da sistemler arasındaki bağlılık derecesi ne kadar kuvvetli (sıkı) ise değiştirilebilirlik de o kadar zordur. Sunucu uygulamasındaki değişiklikler istemci uygulamasını da etkileyeceğinden o kısımlarda da gerekli değişikliklerin yapılması gerekmektedir. Uygulamalar arasında kullanılan servisler, kendisini kullanan diğer uygulamalara hissettirilmeden yer ve teknoloji değiştirilebiliyorsa bu uygulamalar birbirlerine gevşek bağlıdır denilebilir.

Servisler birbirlerine şema ve kontratlarla bağlı oldukları için birbirlerine bağımlılıkları zayıftır. Bir servisi kaldırıp yerine -farklı bir programlama diliyle yazılsa bile- aynı şemaya uygun şekilde tasarlanmış başka bir servisi koyabiliriz. Servislerin bir tanesinde ortaya çıkan bir problemin diğer servisleri etkilemesi çok zayıf bir ihtimaldir. Örneğin SMS gönderme servisindeki bir problem, kişinin bilgilerini getiren servisi etkilemeyecektir. Gevşek bağlantının amacı bağımlılıkları en aza indirmektir. Birbirlerine daha az bağımlı olan sistemlerin herhangi birisinde ortaya çıkabilecek bir aksaklık ya da arıza diğer sistemleri çok fazla etkilemeyecektir (Josuttis 2007).

Çizelge 2.1. Sıkı bağlılık ile gevşek bağlılığın karşılaştırılması (Josuttis 2007).

	Sıkı Bağlılık	**Gevşek Bağlılık**
Fiziksel Bağlantı	Noktadan noktaya	Arabulucu ile
İletişim Tarzı	Senkron	Asenkron
Veri Modeli	Ortak karmaşık tip	Sadece basit yaygın tip
Sistem tipi	Güçlü	Zayıf
Etkileşim modeli	Karmaşık nesne ağaçları içinden gezinerek	Veri merkezli, kendi kendine yeten mesaj

Çizelge 2.1. (devam)

Süreç mantık kontrolü	Merkezi kontrol	Dağıtık kontrol
Bağlayıcı	Statik	Dinamik
Platform	Güçlü platform bağımlılığı	Platform bağımsız
İşleme	2PC (iki fazlı işlemek)	Dengeli
Açılma	Eş zamanlı	Farklı zamanlarda
Versiyonlama	Açık güncelleme	Kapalı güncelleme

2.2.5. Birlikte çalışabilirlik (*Interoperability*)

SOA'da birlikte çalışabilirlik önemlidir. Heterojen sistemleri birbiriyle konuşturabilmek, SOA'nın genel amaçlarındandır. Birlikte çalışabilirlik, farklı sistemlerin birbirleriyle konuşabilmesi ve birbirlerinin yeteneklerini, hizmetlerini kullanabilmeleri durumudur. Bir sistem başka bir sistem için herhangi bir işlemi gerçekleştirebilir. Bilgisayar teknolojisi açısından bakarsak iki heterojen bilgisayar sisteminin birlikte çalışması, karşılıklı olarak kendi kaynaklarına erişime izin verilmesi durumudur. Ağ işletmeleri bağlamında bakıldığında birlikte çalışabilirlik kurumsal sistemler arasında bilgi ve hizmet alışverişi yeteneğini ifade eder. Birlikte çalışabilirlikte etkileşimler üç farklı düzeyde yer alabilir. Bunlar; belirli bir iş bağlamında tanımlanan semantik veri, servis ve süreçler olarak sıralanabilir (Chen *et al*. 2008).

2.3. Servis Odaklı Mimari Temel Bileşenleri

2.3.1. Kurumsal servis yolu (*Enterprise service bus*)

Altyapı, yüksek çalışabilirlik sağlayan SOA'nın teknik bir parçasıdır. SOA'nın alt yapısı kurumsal servis yolu olarak adlandırılır. Bu terim ismini kurumsal uygulama entegrasyonundan alır. KSY'nin en önemli özelliği heterojen sistemler arasında servis çağırmayı mümkün kılmasıdır. KSY'nin sorumlulukları veri dönüşümü, akıllı

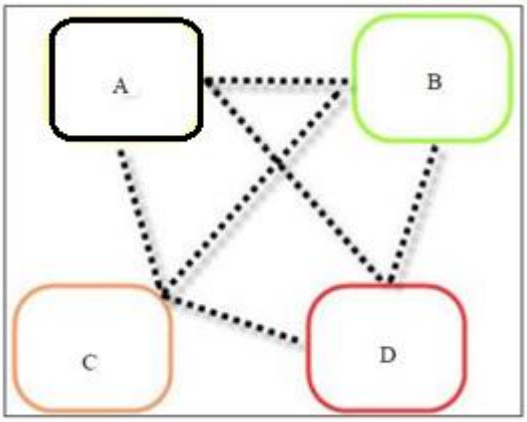
yönlendirme, güvenlik ve güvenilirlik, servis yönetimi, izleme ve loglamayı kapsamaktadır (Josuttis 2007).

KSY, yazılım sektöründe nispeten yeni bir terimdir. Bir KSY'nin faydalarını anlamak için büyük bir işletmenin var olan altyapı ihtiyaçlarını incelemek gereklidir. Bir kurumda yüzlerce, binlerce uygulama mevcut olabilir. Bu uygulamaların şirketin işlerinde, birlikte çalışabilirliğin ve veri alışverişinin gerçekleştirilebilmesi için, birbirleriyle iletişim kurabilmeleri gerekir. KSY, güvenli ve güvenilir bir şekilde farklı dağıtık uygulamaların etkileşimlerini kolaylaştırmak için yönlendirme, çağırma ve arabuluculuk hizmeti sağlayan mesaj tabanlı, dağıtılmış, entegrasyon alt yapısı olan açık standartlardır (Menga 2007).

KSY daha karmaşık mimariler için temel hizmetleri sağlamak için SOA kullanan bir katman olarak tanımlanır. Bir KSY'nin görevlerini servisler arasında mesaj alışverişinin kontrolü, servislerin yönlendirilmesi, dağıtımı, izlenmesi olarak tanımlayabiliriz. KSY mimarisi servis tüketici, servis üretici ve KSY olarak 3 parçaya ayrılmıştır. Bunlar kullanıldığında KSY çeşitli faydalar sağlar. Bu faydalar için ihtiyaçlara göre değişimin kolaylığı anlamına gelen esnekliği, entegrasyonun kodlamadan ziyade yapılandırma gerektirmesini söyleyebiliriz (Cruz 2014).

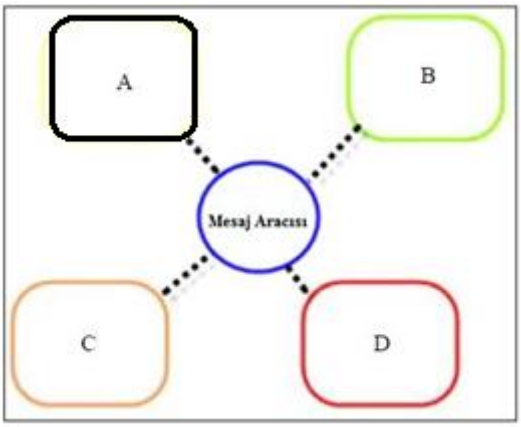
Şekil 2.2.. KSY'nin üç ana bileşeni (Kress *et al.* 2013).

Noktadan noktaya model için (Şekil 2.3) uygulama entegrasyonunun karmaşıklığı, iletişime ve paylaşıma ihtiyacı olan her yeni uygulama ile önemli artış gösterir. Her yeni uygulamanın yazılmasında özel kod olması gereklidir, bu da bakım maliyetlerini artırır.



Şekil 2.3. Noktadan- Noktaya

Bu verimsiz model kurumsal uygulama entegrasyonu (Şekil 2.4.) olarak adlandırılan yeni bir yaklaşıma yol açmıştır. Bu mimaride, tüm iletişim mesaj aracısı tarafından gerçekleştirilir. Mesaj aracısı sadece yönlendirme için tasarlanmış değildir, çoğu zaman da veri transformasyonu için kullanılır.



Şekil 2.4. Mesaj aracısı

KSY (Şekil 2.5) bu iki mimari üzerinde bir gelişmedir. Heterojen uygulamaları ve SOA'nın servislerini bağlamada kritik bir rol oynar.

Şekil 2.5. Kurumsal servis yolu mimarisi (Madhusoodanan and Krishnan 2008)

Bu katman sadece veri taşımadan sorumlu değildir aynı zamanda bir dönüşüm katmanı olarak da hizmet vermektedir. Bu veri dönüşümü eski sistemlere, daha yeni uygulamalarla iletişim ve veri paylaşımını sağlar. ST'ler ile SS'ler arasındaki iletişimin sorumluluğunu KSY yürütmektedir. KSY'nin bu çekirdek işlevselliği herhangi bir organizasyonun gelişmekte olan uygulamaları için çok önemli bir özelliktir. Tüketici istekleri ve mesajların yönlendirilmesi KSY'nin bir başka önemli rolüdür. Bu işlevsellik farklı uygulamaların birbirleri ile iletişim kurma ve entegrasyon çabalarını basitleştirmeye yardımcı olur. KSY tarafından yapılan yönlendirme kararları mesajın içeriği, mesajın başlığı ve ileşitim türü gibi bir dizi faktörlere bağlı olabilir. Sanallaştırma ya da proxy, bir KSY'nin oynayabileceği başka bir roldür. Bu rolde, KSY SS'ler için bir proxy gibi davranır ve ST'lerin isteklerini işler. KSY kimlik doğrulama, yetkilendirme ve denetimi de gerçekleştirebilir (Ahuja and Patel 2011).

2.3.2. Mimari kavramı (Architecture)

İş mimarisinde ilk öneri Zachman tarafından yapıldığından beri bazı araştırmacılar da kendi mimari modellerini ve metodolojilerini geliştirdiler. Bununla birlikte detaylardaki ayrıntıların eksikliği, güncel teknolojilerin iş gereksinimlerinin çok uzağında olması sebebiyle gerçek projelerde ihtiyaç duyulan sonuçları veremediler (Serrano *et al*. 2014).

WS'nin icadı ile servis yönelimli programlama dağıtık uygulamalar için bir bağlantı yöntemi olarak standart ve mimarileri geliştirmek için en yaygın yol haline gelmiştir. SOA günümüz karmaşık ve heterojen bilgi işlem ortamlarında kolaylıkla entegrasyon gerçekleştirebilme özelliği ve mimari yaklaşımıyla hızlı bir şekilde yerini almaktadır. SOA'yı sadece servis yazılım dağıtıcı olarak değil, kuruluşların kendi iş modelleri, servis odaklı analiz ve tasarım teknikleri, destek planları, ortak müşteri-tedarikçi ilişkileri ile birlikte de düşünmek gerekmektedir (Papazoglou and Heuvel 2006).

2.3.3. Süreçler

Büyük sistemleri karmaşık kılan şey çok farklı insanları ve ekipleri içlerinde barındırmalarıdır. Bunun bir sonucu olarak bir işin fikir aşamasından yapılmasına kadar süreç uzun bir yoldur. Çünkü herşeyi kontrol eden sadece bir veya birkaç kişi değildir. Bu nedenle açıkça tanımlanmış uygun süreçler kurulmak zorundadır. SOA'da bu süreçler şu şekildedir:

- **İş süreçleri modelleme:** Etkinlik ve görevlerin daha kısa iş süreçlerine bölünmesi durumudur.
- **Servis yaşam döngüleri:** Bir servisin çalışma süresinin ne nadar olacağı, hangi adımları takip edeceği bilgileridir.
- **Yazılım geliştirme aşaması:** Tasarlanann modeller için kod üretme sürecidir (Josuttis 2007).

2.3.4. Yönetim

SOA sistem geliştirme ve yönetim mekanizmalarına yönelik yeni yaklaşımları gerekli kılar. Merkezi yönetim mimarisi geliştirilerek kontrolün sağlanması SOA mimarisi için önemlidir (Hustad and Lange 2014).

Kurumsal yönetim, bir şirketin nasıl çalıştırılması gerektiğini etkileyen kurallar, yasalar, politikalar ve düzenlemeler kümesini olarak tanımlanır. KY'de, şirketlerin verimli ve

sorumlulukla, düzgün çalıştığından emin olunmalıdır. BY yönetim hedefi BT projelerinin risklerini en aza indirmek ve BT'nin gerçek iş değeri sağladığından emin olmaktır. BT projelerinin neredeyse üçte ikisinin başarısız olduğunu düşünürsek, iyi bir yönetim birimine ihtiyaç duyulduğu anlaşılır. SOA için yönetimin yinelemeli bir yaklaşım olarak uygulanması gerektiğini bilmek önemlidir. SOA'da yönetimin uygulanmasının amacı SOA için en iyisini elde edebilmektir. Bunun için aşağıdaki adımların atılması gereklidir:

- Uygulamada istenen politikaların tanımlanması
- Tasarım sırasında bu politikaların uygulanması
- Çalışma sırasında politikaların uygulama ve izleme çalışmaları (Dirksen 2013).

2.4. Web Servisler

Bir WS internet üzerinden kendisini kullanılabilir kılan ve standartlaştırılmış bir XML mesajlaşma sistemini kullanan yazılımın bir parçasıdır. XML, bir web servisinde tüm iletişimi kodlamak için kullanılır. Örneğin, bir istemci (client) bir XML mesajı göndererek bir web servisini çağırır ve bu çağrıya gelecek olan XML yanıtı bekler. Tüm iletişim XML olduğundan WS herhangi bir işletim sistemi veya programlama diline bağımlı değildir. Java uygulamaları Perl uygulamaları ile konuşabilir, Windows uygulamamaları Unix uygulamaları ile konuşabilir (Cerami 2002).

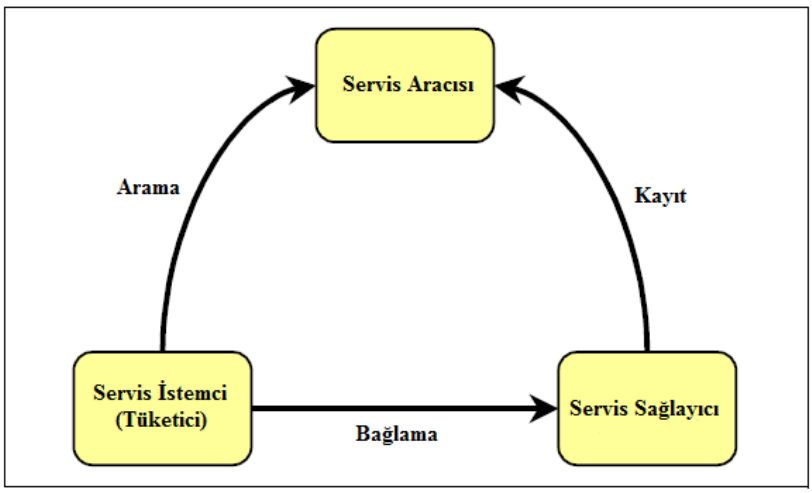
WS'ler SOA'da bir uygulama olarak kullanılan önemli bir teknolojidir. WS'ler internet üzerinden HU'ları birbirleriyle entegre etmek için bir dağıtık bilgisayar yaklaşımı sağlar. WS özellikleri, ST ve sağlayıcı arasında gevşek bağlantı tanıtmak için programlama dili, işletim sistemi ve donanımdan tamamen bağımsızdır (Endrei *et al*. 2004).

WS'ler ağ üzerinden yayınlanan, çağrılan ve kendi kendine yetebilen, modüler, dağıtık dinamik uygulamalardır. Tipik bir WS başka mevcut uygulamadan bir hizmet isteyen iş uygulamasıdır. İstek, bir HTTP veya Java Message Service (JMS) üzerinden SOAP

mesajları kullanarak belirli bir web adresi üzerinden işlenir. Aynı zamanda doğrudan Enterprise JavaBeans (EJB) olarak da çağrılabilir. Servis isteği alır, bunu işler ve bir yanıt verir. Basit bir web servisine örnek olarak hisse senedi bilgilerini almak verilebilir. Metot senkrondur yani sonuç hazır olana kadar metot bekler. WS'ler, gerçek servis veya servise erişen istemci içerebilir. WS'ler uygulamalar ile entegre edilerek iş süreçlerinin esnekliğini artırmaya yardımcı web uygulamalarıdır. WS'ler SOA yaklaşımını yansıtmaktadır. Bu yaklaşım uygulamaları ve ağda kullanılabilir servisleri keşfederek veya mevcut uygulamaları çağırarak bir görevi gerçekleştirme yaklaşımına dayanır. WS'ler birlikte çalışabilirlik sağlar. WS uygulamaları farklı programlama dillerinde oluşturulan bileşenlere sanki aynı dili kullanmışlar gibi birlikte çalışabilme imkânı sağlar (IBM 2016).

SOA'da uygulama tarafında ihtiyaç duyulan tüm işlevselliği sağlamak için uygulamalar gevşek bağlı servislerden oluşur. Her servis genellikle kendi kendine yetecek şekilde tasarlanmıştır. SOA içerisinde web servislerle ilgili üç büyük rol vardır. Bunlar:

- Servis Sağlayıcı
- Servis Aracısı
- Servis Tüketicisi



Şekil 2.6. Web servis aracı-tüketici-sağlayıcı ilişkisi

Servis sağlayıcı: SS'ler bir servisi oluşturup, servis aracısı için kendi arayüzü ve erişim bilgilerini yayınlayabilir. Bir servis SS servislerin gösterilmesini ve bunun nasıl gerçekleşeceğini belirler.

Servis aracısı: Servis aracısı herhangi bir potansiyel servis isteğinde kullanılmak üzere servis arayüzü ve uygulama erişim bilgilerini yapmakla sorumludur. Servis aracısı kayıt ve servisleri bulmak için mekanizmalar sağlar. Belirli bir servis aracısı kamu veya özel, sadece sınırlı bir kitle için mevcut olabilir.

Servis tüketici: Servis istemcisi olarak da bilinen ST'ler, servisleri keşfeder ve daha sonra yapılacak işlem kapsamında bunları kullanır. ST'ler SS'ler tarafından sağlanan servisleri kullanır (Keen *et al*. 2012).

Juric *et al.* (2007) web servislerin özelliklerini aşağıdaki temel başlıklar halinde tanımlamışlardır.

Kendi kendine yetebilirlik: Web servisleri herhangi bir bileşeni istemci tarafında yüklü olmasına gerek olmadan kendine yetebilir. Sunucu üzerinde sadece bir Servlet motoru, bir EJB konteyneri veya .NET çalışma zamanı hizmeti dağıtmak için gereklidir. Servis dağıtıldığında ve çalışma için hazır olduğunda, bir istemci makine üzerinde herhangi bir yazılım yüklenmesine ihtiyaç duymadan servis kullanılabilir.

Kendi kendini tanımlayabilirlik: WS'ler kendi kendini tanımlayabilme özelliğine sahiptir. Bir WS için bir arayüz WSDL belge ile yayınlanmaktadır. Böyle bir WSDL belge, mesaj alışverişi ve mesajlarda kullanılacak veri türleri için biçimi tanımlar. Bir servisi kullanmak için, istemcinin(client) yalnızca bir istek ve yanıt mesajının formatı ve içeriği hakkında bilgi sahibi olması gerekir.

Modülerlik: WS'ler J2EE, CORBA, DCOM gibi mevcut bileşen teknolojileri üzerine dayalı bir soyutlama sağlar. Böyle çeşitli teknolojiler kullanarak, bileşenler oluşturulur. WS'ler istemciye bir hizmet sunmak için bu bileşenleri oluşturur. Bileşenlerin arayüzü istemci ile karşı karşıya gelmemektedir. Bir iş servisinin daha soyut bir görünüm oluşturması modüler bir yazılım geliştirmeyle sonuçlanır.

Web üzerinden erişilebilirlik: WS'ler yayınlanır ve web üzerinden çağrılır. WS'ler standart web protokollerini kullanırlar. Servis tanımı WSDL kullanılarak

yayımlanır, UDDI (Universal Description, Discovery, and Integration) yardımıyla bulunur ve SOAP kullanarak çağrılır. Bütün bu protokoller web tabanlıdır.

Dil, platform, protokol bağımsızlık: Web hizmetleri açık XML standartlarına dayalı olduğundan dil bağımsızdır. Herhangi bir dilde yazılmış bir istemci başka bir dilde yazılmış bir WS'ye erişebilir. WS'ler platform bağımsızdır; Tüketici ve servis iki bağımsız platformlarda çalışıyor olabilir. WS'ler herhangi bir standart ağ protokolü kullanılarak çağrılabilirler.

Açık ve standart tabanlılık: WS teknolojisi açık standartlara dayanmaktadır. Bu standartlar XML tabanlı SOAP, WSDL ve UDDI'dir.

Dinamiklik: WS'ler keşfedilebilir ve bunların herhangi bir derleme zamanı bilgisine sahip olmaya gerek kalmadan çalışma zamanında tüketilebilirler. Diğer birçok teknolojide istemci, bileşen arayüz derleme zamanı bilgisine ihtiyaç duyulur.

Birleştirilebilirlik: WS'ler daha geniş bir serviste toplanmış olabilir (Juric *et al*. 2007).

Web servis standartları aşağıdaki gibi sıralanabilir:

- XML (Extensible Markup Language)
- SOAP (Simple Object Access Protocol)
- UDDI (Universal Description, Discovery and Integration)
- WSDL (Web Services Description Language)

2.4.1. XML (*Extensible Markup Language*)

XML WS standartlarının temelidir. Tanımlama, keşfetme ve Web servislerinin yürütülmesi için tüm standartları XML'e dayanmaktadır. SOAP ve WSDL, XML şema kullanılarak tanımlanır (Remy and Rosenberg 2004).

XML Genişletilebilir İşaretleme Dilinin kısaltmasıdır. Standart Genelleştirilmiş İşaretleme Dili'nden (SGML) türetilen bir metin tabanlı işaretleme dilidir. XML etiketleri verilerin tespit edilmesi, depolanması ve düzenlenmesi için kullanılır. XML bir format içerisinde belgeleri kodlama için kurallar kümesi tanımlayayan hem insan

hem de makine tarafından okunabilen bir işaretleme dilidir. İşaretleme belirli şekillerde anlamını artıran ve bir belgeye eklenen bir bilgidir. XML'in özellikleri:

- Genişletilebilirdir, kendi kendini açıklayıcı etiketleri oluşturmak için olanak sağlar.
- Veri taşır ve nasıl sunulacağına bakılmaksızın veri depolamak için izin verir.
- World Wide Web Consortium (W3C) tarafından geliştirilmiştir ve açık bir standart olarak mevcuttur.
- Büyük web siteleri için HTML belgeleri oluşturulmasını basitleştirmek için sahne arkasında çalışabilir.
- Kuruluşlar ve sistemler arasında bilgi alışverişi için kullanılabilir.
- Veri tabanlarının boşaltılması ve yeniden yüklenmesi için kullanılabilir.
- XML her istenilen çıktıyı oluşturmak için kolayca hemen stil sayfaları ile birleştirilebilir.
- Her türlü veri bir XML belge olarak ifade edilebilir (Anonymous 2014).

2.4.2. SOAP (*Simple Object Access Protocol*)

SOAP, HTTP ve Genişletilebilir Biçimlendirme Dili (XML) kullanarak farklı işletim sistemleri üzerinde çalışan programlara birbirleriyle iletişim olanağı sağlayan bir mesajlaşma protokolüdür. SOAP WS'ler ile etkin uygulamalar arasında iletişim kurmak için XML tabanlı mesaj formatını tanımlar.

SOAP'ın avantajları :

- Platform ve dil bağımsız olması,
- Proxyler ve güvenlik duvarları arasında basitleştirilmiş iletişim sağlayabilmesi,
- HTTP, SMTP gibi farklı iletişim protokollerinden yararlanma yeteneğine sahip olması.

SOAP'ın dezavantajları:

- Ayrıntılı bir XML fotmatını kullanması gerçeği nedeniyle SOAP tipik olarak diğer katman standart türlerinden daha yavaştır. SOAP etrafında uygulamaları oluşturmadan önce tam performans sınırlamasını anlamak gereklidir.
- Desteklenen programlama diline bağlı olarak SOAP desteği farklı düzeylerdedir (Techtarget 2014).

```
<?xml version="1.0"?>
<soap:Envelope
Xmlns:soap=http://www.w3.org/2001/12/soap-encoding>
<soap:Header>
……
</soap:Header>
<soap:Body>
……..
        <soap:Fault>
        …….
        </soap:Body>
</soap:Envelope>
```

Şekil 2.7. SOAP mesaj yapısı iskeleti

SOAP üç temel bileşenden oluşur. Bunlar:

- SOAP zarf özellikleri (*SOAP Envelope Specification*)
- Veri çözümleme kuralları (*Data encoding rules*)
- RPC conventions (*Remote Procedure Calls conventions*)

2.4.2.a. SOAP zarf özellikleri

SOAP XML zarf, sistemler arasında transfer edilen veri için özel kurallar tanımlar. Bu uygulama metot adı çağırmak, metot parametreleri veya dönüş değerleri gibi özel bir veri içerir. Aynı zamanda zarf içeriğini kimlerin işleyebilmesi gerektiği, başarısızlık durumunda nasıl bir hata iletisi kodlamak gerektiği bilgisini içerir (Cerami 2002).

SOAP zarf elemanı SOAP mesajında kök öğedir. SOAP zarf içerisinde isteğe bağlı olarak bir 'header' elemanı ve bir de 'body' elemanı bulunur. Örnek bir SOAP zarf elemanı Şekil 2.8 deki gibidir.

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap=http://www.w3.org/2001/12/soap-envelope
soap:encodingStyle=http://w3.org/2001/12/soap-encoding>
……..
        Mesaj bilgisi buraya gelir
……..
</soap:Envelope>
```

Şekil 2.8. Örnek bir SOAP zarf elemanı

Burada zarf ad alanı beyanı 'xmlns:env="http://www.w3.org/2002/06/soap-envelope"' ile ifade edilir ve bulunması zorunludur. SOAP header elemanı zarf elemanının isteğe bağlı bir alt öğesidir. Zarfın içerisine koyulmazsa bir problem teşkil etmez. Header kısmına SOAP mesajının bir parçası olmayan istenilen bir bilgi koyulabilir. Body kısmı ise istemci veya herhangi bir web hizmeti tarafından işlenecek ana bölümü içeren, konulması zorunlu elemandır (Jenkov 2014).

2.4.2.b. Veri çözümleme kuralları

SOAP bir bütün olarak mesajın veya onun herhangi bir bölümü için geçerli çözümleme kurallarını belirtmek için bir mekanizma sağlar. Bu çözümleme SOAP zarf kısmında çözümleme stil niteliği ile yapılır. SOAP özel bir veri çözümleme kurallar kümesi tanımlar. Onlar SOAP mesajlarının içinde SOAP-ENV:encodingStyle = "http://schemas.xmlsoap.org/soap/encoding" ile belirtilir. Sık sık bir SOAP mesajında zarf elemanına doğrudan uygulanan bu özelliği görebilirsiniz. SOAP mesajında varsayılan çözümleme kavramı yoktur, çözümleme stili açıkça belirtilmelidir. Kuralları kodlayan SOAP verileri soyut veri modelleri ve XML söz dizimi arasında iyi tanımlanmış bir eşleştirme sağlamak için vardır. SOAP çözümleme kuralları aşağıdaki üç görevi yürütmek için algoritmalarını tanımlar:

- Soyut veri modellerine dair üst veri verildiğinde ondan bir XML şeması oluşturabilir.
- Veri modelinin bir örneği verildiğinde şemaya uyan XML üretibilir. Bu seri bir işlemdir.
- Şemaya uyan XML verildiğinde, soyut veri modelinin şemasına uyan örnek bir grafik oluşturabilir (Informit 2001).

2.4.2.c. RPC sözleşmeler

SOAP içerisinde uzak yordam çağrıları esasen HTTP üzerinde SOAP çözümleme kurallarına uyan istemci – sunucu etkileşimidir. HTTP içinde istek - URI (Universal Resource Identifier) tipik olarak bir sınıf veya bir nesneyi eşleştirmek için sunucu ucunda kullanılır (Extreme 2000).

SOAP amaçlarından biri de uzaktan yordam çağrısı (RPC) için basit, hafif bir mekanizma sağlamaktır. RPC tek yönlü veya iki yönlü iletimi içerebilir. SOAP vücut (body) elemanında çağrı ve cevap taşınabilmektedir. Aşağıdaki bilgiler RPC için gereklidir.

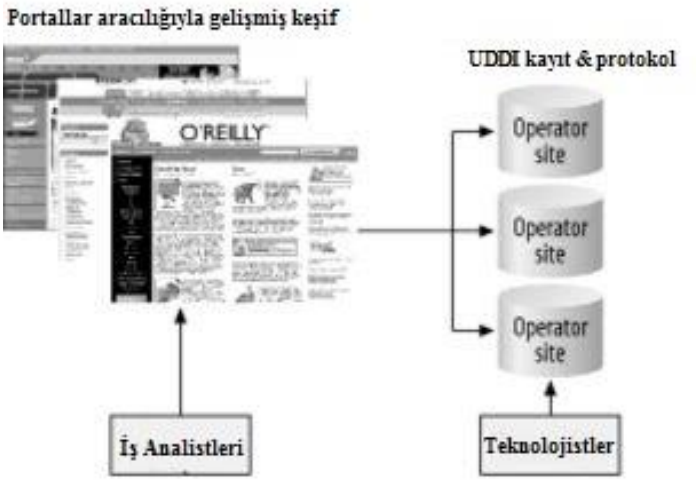
- Hedef URI (*Uniform Resourse Identifier*)
- Yordam veya yöntem adı
- Yordam veya yöntemi parametreleri
- İsteğe bağlı prosedür veya yöntem imzası
- İsteğe bağlı başlık verileri (Anonymous 2001)

2.4.3. UDDI (*Universal Description, Discovery, and Integration*)

UDDI 1.0 aslen Eylül 2000'de Microsoft, IBM ve Aruba tarafından açıklandı. İlk açıklanmasından bu yana Fujitsu, HP, Hitachi, IBM, Intel, Microsoft, Oracle, SAP, Paz ve Dell de dâhil olmak üzere 300 den fazla şirketi kapsayacak şekilde büyüdü. Mayıs 2001'de, Microsoft ve IBM ilk UDDI operatör siteleri başlattı. 2001 yılının Haziran ayında, UDDI Sürüm 2.0 duyuruldu. Şu anda UDDI OASIS tarafından desteklenmektedir (Anonymous)

UDDI internette kendilerini listelemek için dünya çapında işletmeler için bir XML tabanlı kayıttır. Onun nihai hedefi web üzerinde şirketlerin birbirlerini bulmasını ve e-ticaret için kendi sistemlerini birlikte çalışabilir hale getirebilmesini sağlayarak online işlemleri hızlandırabilmektir. Proje işletmelere ad, ürün, yer veya sundukları web servislerini birbirlerine listeleme olanağı sağlar (Techtarget 2005).

UDDI'nin kimin kullandığı perspektifine dayalı olarak çeşitli kullanımları vardır. Bir iş analistinin bakış açısıyla, UDDI iş süreçleri için bir internet arama motoruna benzer. Bir iş analisti farklı işletmelerin servis ve özellikleri için bir veya daha fazla UDDI kayıtlarına göz atabilir. Çeşitli kriterlere uyan servisleri keşfetmek için yazılım geliştiriciler servislerini yayımlamak ve kayıt sorgulamak için UDDI programcı API kullanır. Yazılım sonunda dinamik bir servis keşfedip ve insan etkileşimi gerektirmeden kullanabileceğimizi düşünebiliriz. İş analistleri ve yazılım geliştiricilerin her ikisi de yeni ticari kuruluşlar ve servislerini yayınlayabilirler (Anonymous 2016)



Şekil 2.9. İş analistleri ve teknolojistler arasındaki ilişki (Anonymous 2016b)

2.4.4. WSDL (*Web Services Description Language*)

WSDL belgesinin kendisi bir XML belgesidir. Bu yüzden herhangi iyi biçimli bir XML belgeden beklenen kurallara uygundur. Bu <definitions> elemanı ile WSDL belgesinde bir kök elemanı olarak yer alan şema ad tanımları ile başlar. Tipik bir WSDL belgesi bir kaç şema tanımını içerir, ancak en önemlisi "<definitions xmlns= "http://schemas.xmlsoap.org/wsdl/"> " dir. <definitions> kök elemanı WSDL belgenin içeriği tamamen çevreler. Aşağıda sunulan tüm unsurlar <definitions> kök öğesinin alt öğeleridir:

<types> : Bu eleman giriş parametreleri olarak xml mesajları ile alınan tüm veri tiplerini listeler ya da tipini döner. <types> öğesi gömülü XSD şema tanımı dosyası ile eşdeğerdir.

<message> : Bu eleman bir Web servisi işlemi için bir giriş, çıkış ya da hata mesajı gibi bir SOAP mesajı açıklar.

<operation> : Bu eleman, bir metot tanımına benzer. Ancak, yalnızca işlem ile ilişkili girdi, çıktı ve hata mesajları tanımlamanızı sağlar.

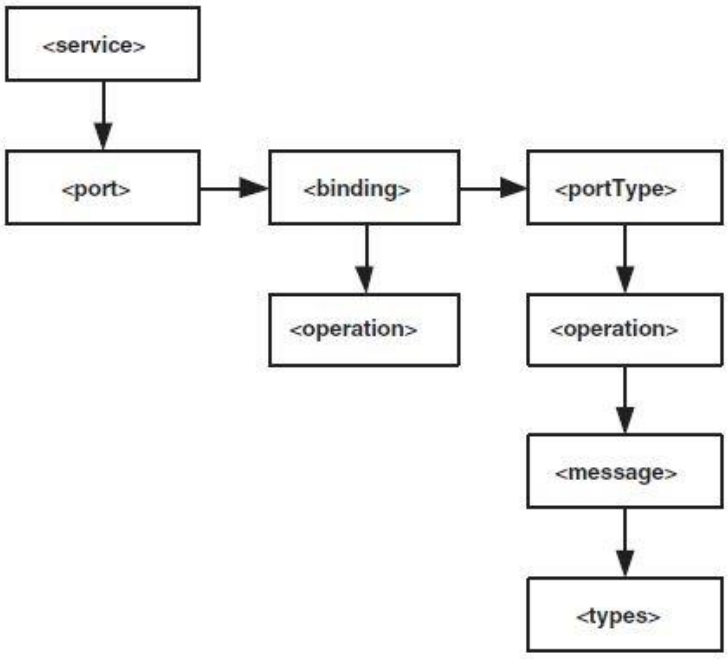
<portType> : Bu eleman bir Web servisinin desteklediği tüm işlemleri listeler. <PortType> öğesi mevcut işlemlerini açıklar iken <Port> elemanı tek bir Web servisine

karşılık gelir. <PortType> öğesi birçok alt düzey detayı önler. Bunun yerine Web servisin sağladığı işlemlerin (ve ilişkili girdi, çıktı ve hata mesajlarının) bir üst düzey özetini sağlar. <PortType> öğesi bir istemci için tek bir konum belirli bir web servis sunumlarına göz atmayı sağlar.

<binding> : Bu eleman soyut ve somut unsurları WSDL belgesi içinde birlikte bağlar. <Binding> elemanı, belirli bir <portType> elemanı ile ilişkilidir ve aynı zamanda <portType> elemanı ile ilişkili Web servis adresini listeler. <binding> elemanı Web servisi ile iletişim kurmak için kullanılan protokolü listelemektedir.

<port> : Bu eleman Web servisinin bulunduğu Tekdüze Kaynak Göstergesi (URI) tanımlar ve aynı zamanda bir <binding> elemanı uygular.

<service>: Bu element bir veya daha fazla <port> öğesini kapsamaktadır (Hasan 2004).



Şekil 2.10. WSDL Belge Yapısı (Hasan 2004).

2.5. Web Servislerinde Güvenlik Standartları

IBM ve Microsoft 2002 yılında "Bir Web Servis Dünyasında Güvenlik: Önerilen Mimari ve Yol Haritası" adında müşterek bir rapor yayınladı. Bu rapor, WS dünyasında güvenlikle ilgilenmek için birleştirici bir yaklaşım içeren bir kısım güvenlik standartları ve teknolojilerini tanımlar. Indrakanti (2012) WS güvenlik standartlarını aşağıdaki başlıklar halinde tanımlamıştır:

WS güvenliği (WS security): WS güvenliği güvenli mesajlaşma için SOAP a uzantılar tanımlar. SOAP mesajlarının güvenlik simgeleri ile ilişkilendirmek için genel amaçlı bir mekanizmadır. WS-Güvenliği, SSL, IPSec 13, XML Doğrulama ve Kriptolama gibi gelişmiş güvenlik teknolojiler üzerine kurulmuştur ve bunlarla uyumludur.

WS bildirimi (WS policy) :WS bildirimi, güvenlik bildirimlerinin sınırlılıklarını ve olanaklarının nasıl ifade edileceğini belirtir. WS sunan organizasyonlara, WS gerekliliklerini özelleştirmek için güvenlik ve gizlilik imkânı tanır. WS bildirimi belirli bir bildirim dili oluşturmak için gerekli olan temel yapıları sunan yüksek seviyede spesifikasyondur.

WS trust (WS güvencesi) : WS Trust aracılar gibi, direkt ve aracı güvencesi sunan modellerin kurulmasını belirtir. WS Güvence Dili (WS Trust), güvenlik sembollerinin çıkarımı, değişimi ve geçerliliği için ek temeller ve uzantılar sunan WS güvenliğinin güvenli mesajlaşma mekanizmalarını kullanır. WS güvencesi ayrıca farklı güvence alanları içerisindeki sertifikaların çıkarımına ve yayımlanmasına imkân tanır.

WS gizliliği (WS privacy) : Bu özellik kullanıcıların gizlilik tercihlerini belirtmesine ve web sitelerinin de gizlilik işlemlerini belirtmesine ve uygulamasına imkân tanır.

WS güvenli konuşma (WS secure conversation): WS güvenli konuşma partiler arasında mesaj değişiminin nasıl yönetilebileceği ve doğrulanabileceğini belirler ve güvenli yer değişimini, kurulumunu ve oturum anahtarları üretimini içerir.

WS federasyonu (WS federation) : WS federasyon heterojen birleşmiş kimlikler için destek gibi, birleşmiş ortamlarda güvenli iletişimin nasıl yönetileceği ve güvenli iletişime nasıl aracılık edileceğini belirler. WS federasyon özelliği farklı güvenlik

alanlarının iştiraki web servisleri arasında kimlik, nitelik, doğrulama güvencesine aracılık ederek ve izin vererek birleştirmesini sağlayacak mekanizmalar belirler.

WS yetkilendirme (WS authorisation): Bu özellik WS data ve poliçe yetkilendirmesinin nasıl yönetileceğini belirler.

WS denetimi (WS auditing) : WS için henüz var olmayan denetim standardıdır. Bir servis çalıştıktan sonra servisi kimin kullandığı ve isteğin nerenden geldiğini belirlemenin yolu yoktur. Dolayısıyla, güvenlikteki olası ihlalleri daha sonra araştırmaya yarayacak bir denetim mekanizması yoktur (Indrakanti 2012).

2.6. Servis Odaklı Mimari Dezavantajları

Herand (2015) servis odaklı mimarinin dezavantajlarından bazılarını aşağıdaki gibi sıralamıştır:

- Uygulamaya geçiş sürecinin uzun olması
- Eski sistemlerin SOA yaklaşımını olumsuz yönde etkilemesi
- IT personelinin SOA bilimsel yaklaşımını benimsemesinin zorluğu
- Türkiye'de yeni bir bilimsel yaklaşım olduğu için Türkçe kaynak edinme zorluğu
- Uygulamaya başlangıç aşaması için yeterli donanıma sahip uzman elaman sıkıntısının yaşanması
- Özellikle Türkiye'de uygulanmış proje sayısının az olması
- Servis sözleşmelerinin içerdiği servis seviye anlaşmaları için uluslararası bir standardın geliştirilmemiş olması
- Servis kaydı için uluslararası bir standardın geliştirilmemiş olması
- SOA'nın uygulanmaya başlanabilmesi için birlikte çalıştığı BPM, BPR, Process Mining gibi bilimsel yöntemlerin de bilinmesi zorunluluğu
- SOA'nın bilimsel bir yaklaşım olmasına rağmen günümüzde birçok kurum ve kişi tarafından salt bir teknoloji olarak algılanması
- Standart bir uygulama stratejisinin noktanlığı (Herand 2015).

2.7. Servis Odaklı Mimari Avantajları

SOA, geliştiricilerin servis katmanında oluşturduğu servislere sahiptir. Onlar geliştirdikleri servisleri ara yüzlerde yayınlarlar. Bu arayüzler farklı bir iş alanı destekliyor. Servislerin oluşturulması çabasında bulunan organizasyonlar bunun birçok faydasını görürler. Uygulama geliştirme organizasyonları için en yaygın senaryo bileşen tabanlı geliştirme ile bazı deneyimlere sahip olmaktır. Uygulamaların barındırılması için J2EE veya .NET gibi uygulama sunucuları kullanımı daha yaygın hale gelmektedir. Kuruluşların iş mantığı için bileşen tabanlı geliştirme uygulamaları ve uygulama sunucuları kullanıyorsa, o zaman zaten SOA kullanılıyor demektir. Stevens (2002) SAO'nın bazı faydalarını şu şekilde sıralamaktadır:

Yatırımdan daha iyi geri dönüş: Sağlam bir servis tabakasının oluşturulması yazılımın oluşturulmasında yapılan yatırımdan daha iyi bir geri dönüş yararı sağlar. Servisler farklı iş alanlarına eşlenir. Örneğin, bir şirket için envanter yönetmek için gerekli yöntemlerin tümü için bir envanter servisini oluşturabilirsiniz. Ayrı bir katman mantığını kullanarak, katmanın ömrünü içinde oluşan herhangi bir sistemin ömründen daha fazla olacak hale getirebilirsiniz. Örneğin, Kuruluşunuzda bir kredi kartı yetkilendirme hizmeti oluşturmak gerekiyorsa, temelde iki seçenek vardır. Geliştiriciler ya ihtiyaç duyulan bir uygulamanın parçası olarak işlevselliği geliştirecek ya da ayrı bileşen olarak geliştirecekler. Eğer kredi kartı yetkilendirmesi ayrı bir bileşen olarak geliştirilmiş ve servis olarak kullanılmış ise orjinal uygulamanın daha uzun ömürlü olması muhtemeldir.

Esneklik: SOA'nın bir özelliği de şeffaflık olduğundan kod hareketliliği bir gerçeklik haline gelir. Bir istemcinin servisi araması ve bağlanması için servisin nerede yerleşik olduğu önemli değildir. Bu nedenle, bir organizasyon farklı makinelere servis sunmak için ya da harici bir sağlayıcıya bir servis vermek için esnekliğe sahiptir.

Geliştirici odaklı roller: Bir SOA bir uygulamayı çoklu katmanlara sahip olması için zorlayacaktır. Her katmanda geliştiriciler için belirli roller dizisi bulunmaktadır. Örneğin, servis katmanı veri formatları, iş mantığı, süreklilik mekanizmaları ve işlem kontrol deneyime sahip geliştiricilere ihtiyaç duyar. Bir istemci geliştirici SWING, JSP

or MFC gibi teknolojileri bilmesi gerekir. Her katman becerileri karmaşık bir dizi gerektirir. Geliştiriciler uzmanlaşabildiği ölçüde, o uygulamanın belirli katmanları onlar için bir zanaat olacaktır. Böylece, şirketler uygulama geliştirme için son derece deneyimli personele bel bağlamak zorunda kalmadan ilgili katmanı bilen personelden yararlanabilecekler.

Daha fazla güvenlik: Tanımı gereği bir servis tabakasının oluşturulması geliştiricilerin birden çok uygulama tarafından kullanılan ek bir ağ arabirimi oluşturması anlamına gelir. Sistemler tek parça veya istemci sunucu yöntemleri kullanılarak inşa edildiğinde güvenlik normalde ön ucunda ele alınır. Birden fazla güvenlik listelerini korumak zor olduğu için şirketler veri tabanı güvenliğini bile hayata geçirmeyebilir. Öte yandan servisler birden fazla uygulama tarafından kullanılır, bu yüzden onların kendi güvenlik mekanizmaları vardır. Bir uygulama bu sebeple hem istemci seviyesinde hem de servis seviyesinde çok düzeyli bir kimlik denetimine sahip olacaktır.

Daha iyi test ve daha az kusur: Servislerin geliştiriciler tarafından birim testleri yazılan kolaylıkla test edilebilir ara yüzleri vardır. Geliştiriciler test paketleri olusturmak Için böyle JUnit gibi araçları kullanabilirler. Bu test suit servisi kullanan herhangi bir uygulamadan bağımsız olarak servis doğrulamak için çalıştırılabilir. Aynı zamanda otomatik oluşturma işlemi sırasında birim tesleri çalıştırmak için de iyi bir uygulamadır. Daha iyi ve daha çok test daha az kusur ve daha yüksek bir kalite anlamına gelir.

Çoklu müşteri tipleri için destek: SOA'nın bir yararı olarak kurumlar servise erişmek için birden çok istemci ve birden çok istemci tiplerini kullanabilir. J2ME kullanan bir PDA HTTP üzerinden bir servise erişebilir ve bir SWING istemcisi RMI üzerinden aynı servise erişebilir. Katmanlar istemci ve servis katmanlarına ayrıldığından beri farklı istemci tiplerinin uygulanması daha da kolaylaşmıştır.

Servis kurulu: Geliştiricilerin oluşturduğu servisler yeniden kullanılabilir bir servis kataloğu halinde gelecektir. Kullanıcılar yeniden kullanılabilir olan bu katoloğu, oluşturucularına mecbur kalmadan anlayabileceklerdir. Yeni uygulamalardan faydalanacak herkes yeniden kullanılabilir servis kataloğunun bir sonucu olarak daha hızlı geliştirilme yapabileceklerdir.

Bakım kolaylığı: Yazılım arkeolojisi kod hatalarını bulma ve düzeltme işidir. Geliştiricilerin daha kolay bulabildikleri için konum olarak servis katmanına odaklanması iş mantığının düzeltilebilirliğini artırır.

Tekrar kullanım: Kodun tekrar kullanımını dil ve platform uyumsuzlukları nedeniyle elde etmek zordur. Bileşeninin ya da servisin yeniden kullanımını sağlamak çok daha kolaydır. Çalışma zamanı servisini yeniden kullanma o dizinde bir servisi bulma ve ona bağlanmak kadar kolaydır.

Geliştirmede parelellik: Birden fazla katmanın yararı birçok gelişticinin bağımsız katmanlar üzerinde çalışabilmesidir. Geliştiriciler için bir projenin başında arayüz sözleşmelerini ve birbirinden bağımsız olarak kendi parçalarını oluşturabilmeleri mümkün olabilir.

İyi ölçeklenebilirlik: Bir SOA gereklerinden biri de yer şeffaflığıdır. Şeffaflığı elde etmek uygulamaların bir dizinde servisleri araması ve çalışma zamanında dinamik olarak bağlanmasıdır. Servis istemcisinin bilgisi olmadan birden fazla servis örneği için bir yük dengeleyici istekleri ilettiği için bu özellik ölçeklenebilirliği teşvik etmektedir.

Yüksek kullanılabilirlik: Birden çok sunucu üzerinde çalışan bir servisin birden çok örneği olabilir. Bir ağ kesimi veya bir makine kapatılırsa, bozulursa görev dağıtıcısı istemcinin bilgisi olmadan başka bir servis isteğini yönlendirebilir (Stevens 2002).

2.8. Servis Odaklı Mimari'de Güvenlik

SOA doğal bir istemci - sunucu bilgisayar evrimidir. Servisler herkes tarafından kolayca oluşturabilir ve bir bağlantı noktasında bir IP adresinden istemcilerin onları kullanılabilmesi sağlanır. Güvenlik alanında, kimlik doğrulama servisleri ve denetim izi toplama gibi şeyler için gelişmekte olan SOA yetenekleri vardır. SOA için, yeni yararlı bir uygulama veya bir iç web sitesinin aksine güvenlik elamanı hızla kurumsal işlemlerin önemli bir elamanı haline gelir (Karimi 2011).

Büyük işletmeler için temel uygulama geliştirme ve entegrasyon modeli olarak servis odaklı mimarideki gelişim çeviklik için büyük kazanımları, maliyet tasarrufunu

vadediyor. Ancak bu kazanımlarla birlikte çeşitli alanlarda artan güvenlik riskleri de mevcuttur. Bunlar:

Mesaj güvenliği: Standartlara dayalı servis etkileşimleri SOA'nın temel faydalarından birisidir. İyi mimarili sistemlerde belirsizliğe yol açacak güvenlik açığına yer yoktur. Fakat servis etkileşimleri de risk taşıyabilir. Gönderici ve sağlayıcıya özgünlük, mesaj gizliliği ve yetkilendirmenin sağlanması için standartlar topluluğu büyük atılımlar yapmıştır.

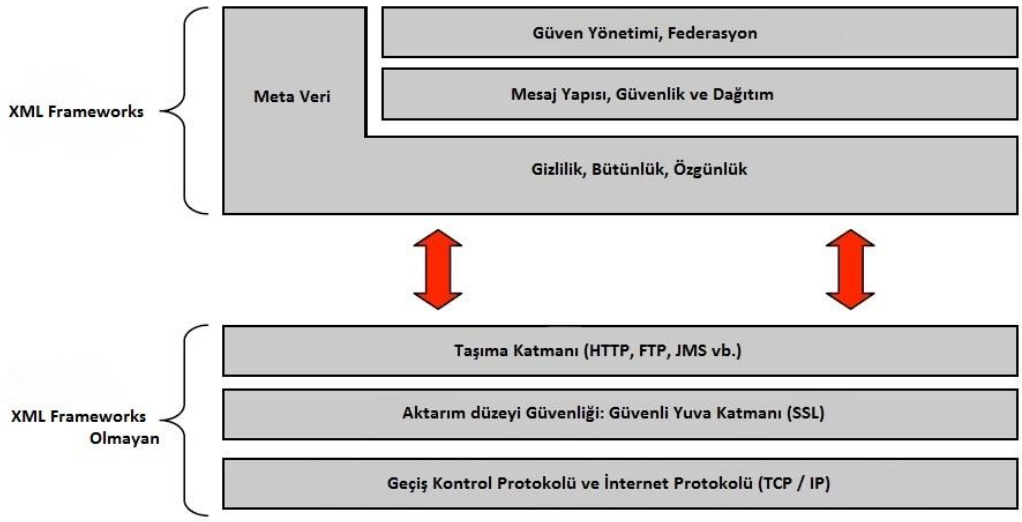
Arayüz güvenliği: SOA amaçlarından birisi de yeniden kullanılabilir ticari servisler oluşturmaktır. Bu servisler sıklıkla mevcut uygulamalardan alınan veri veya iş mantığı ile oluşturulur ve yayınlanır. Bunun anlamı bir uygulama içerisine veri veya mantığın gömülmesi ve erişebilir hale getirilmesi ve bu süreçte potansiyel tehditlere maruz kalabilmesidir.

Güvenlik altyapısı: Kurumsal SOA'ya yönelme, kayıt / havuz, politika yönetimi ve hizmet yönetimi de dâhil olmak üzere yeni altyapı çözümleri dağıtımını içerir. Bu çözümlerin her biri mevcut kurumsal güvenlik politikalarına uygun olmalıdır yoksa kurumsal uygulamaların güvenliğini sağlamak üzere tasarlanmış çözümler kendileri potansiyel saldırı noktaları olabilir (Anonymous 2015)

2.8.1. Servis odaklı mimaride güvenlik standartları

SOA güvenlik standartlarının temel amacı heterojen müşteri ortamlarında kullanılan çoklu ürünler arasında birlikte çalışabilirlik için bir temel sağlamaktır. Standartlara dayalı uygulama stratejileri gelişimini hızlandırır, entegrasyonu kolaylaştırır ve zamanla yönetici maliyetlerini azaltır.

Güvenlik standartları taşıma seviyesinde XML framework olmayan ve uygulama seviyesinde XML framework olanlarda Şekil 2.11'de gösterildiği gibi uygulanmaktadır.



Şekil 2.11. Güvenlik standartlarının uygulanması (Oracle 2008)

Gizlilik, bütünlük, özgünlük : XML şifreleme, XML İmzası
Mesajı düzeyinde güvenlik : WS Güvenlik.
Güvenli ileti Dağıtımı: WS Adresleme, WS Güvenilir Mesajlaşma.
Meta veri: WS Politikası, WS Güvenlik Politikası.
Güven yönetimi: SAML, WS Güven, WS Güvenli Konuşma, WS Federasyonu
Genel anahtar altyapısı: PKCS, PKIX, XKMS (Oracle 2008).

SOA güvenlik standartlarını Indrakanti (2012) aşağıdaki başlıklar halinde tanımlamıştır:

XML imza (XML signature) : XML imza (doğrulama), WS güvelik spesifikasyonu ve WS güvenliği için temel bir teknolojidir. WS güvenliği, XML anahtar yönetim spesifikasyonu (XKMS) ve diğer WS güvenlikleri için çekirdek niteliğindedir. Ayrıca, XML şifreleme için gerekli olan paylaşılmış gizli anahtarların transfer edilmesinde de kullanışlı bir araçtır.
XML şifreleme (XML encryption): XML imza doğrulamaya benzeyen, XML kriptolama da paylaşılmış anahtar kriptolama teknolojisinin kullanımı üzerine kurulmuştur ve bir W3C önerisidir. Herhangi bir XML mesajını etkili bir şekilde şifreleyebilme XML kriptolama için ana gerekliliktir. Mesajın hedefine giderken

yolundaki duraklardan gizli bir şekilde geçip devam edebilmesi için SSL gibi transfer seviyesi üzerinde şifreleme XML kriptolama için gereklidir. Bu da paylaşılan servisler kullanıldığında sıklıkla görülür. XML kriptolama, XML mesajı depolandığında hatta hedefine ulaştığında bile mesajın gizliliğini korur. Dataya bazı standart algoritmalar uygular ve ardından XML' in içindeki şifrelenen bu sonuçları depolar.

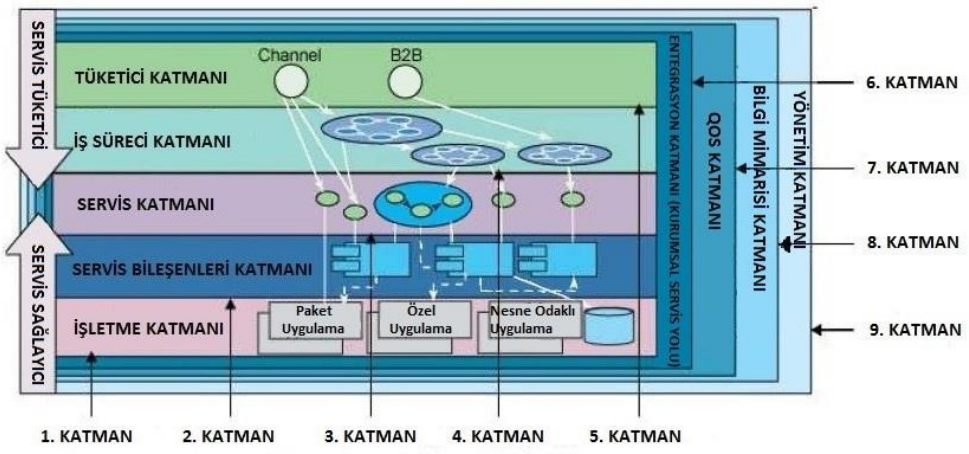
XML anahtar yönetim özellikleri (XML Key Management Specification): Aynı zamanda bir XML standardı bileşeni olan XKMS, dijital imzalama ve kriptolama açısından XML standartlarının tepesinde yer almaktadır. XML doğrulama ve XML kriptolama standartlarıyla uyumlu kullanılabilen genel anahtarları dağıtmak ve kaydetmek için protokoller sunmak XKMS'ın asıl amacıdır.

SAML: OASIS tarafından sunulan SAML, kullanıcılar ya da nesneler hakkında bilgi doğrulaması, yetkilendirme değişimi yapan güvenlik spesifikasyon tabanlı bir XML dir.

XACML: XACML, OASIS tarafından erişim kontrolü için standardize edilmiş ilke tabanlı bir XML dilidir. XACML hem erişim kontrolü ilke dilidir hem de XACML istek/cevap mesajları için bir SAML profili destekler (Indrakanti 2012).

2.9. Servis Odaklı Mimari Katmanları

Servis odaklı mimari genellikle her bir tabakanın kendi amaçları olan dokuz tabaka olarak sunulur.



Şekil 2.12. Servis odaklı mimari katmanları (Flurry 2007)

Servis tabanlı mimarinin katmanlarını Arsanjani ve diğerleri (2014) aşağıdaki başlıklar şeklinde tanımlamıştır:

Operasyonel (işletme) Katman: Operasyonel katmanın görevi IT işletim ortamında çalışan uygulamaları idare etmektir. Operasyonel katman uygulama yazılım sistemlerinin inşa edilmesidir. Bu SOA çözüm uygulamasındaki toplam maliyeti doğrudan etkiler. Tek parça özel uygulamalar (J2EE, .NET), eski uygulamalar, mevcut işletim sistemleri ve veritabanı gibi mevcut yazılım sistemleri bu katmana dâhildir.

Servis bileşenleri katmanı: Servis bileşenleri katmanı bir servis kullanımı için yazılım bileşenlerini içerir. Bu yazılım bileşenleri SOA'ya dayalı servis bileşen mimarisi olarak ortaya çıkar. Servis bileşen mimarisi web hizmetleri, kurumsal bilgi sistemi hizmeti varlıkları, iş akışları ve veri tabanları gibi tüm iş süreçlerini sunar.

Servis katmanı: Servis katmanı bir SOA içinde tanımlanan tüm servislerin bulunduğu katmandır. Bu referans mimarisi bir kolleksiyonun soyut özelliğinin bir mimarisi olarak kabul edilir. Bu özellik istemcilere bir bağımsız platform ile işletme fonksiyonlarını çağırma imkânı sunar. Bu genellikle bir web hizmetleri tanımlama dili (WSDL) soyut aşaması ile yapılır. Bu katmandaki servislerin ortak özelliği fonksiyonları iyi tanımlanmış arayüzler aracılığıyla bir ağ üzerinden erişilebilir olmasıdır.

İş süreç katmanı: Bu katmanda bileşimler ve servislerin koreografileri tanımlanır. Servisler koreografi aracılığıyla akış halinde paketlenmiştir, bu nedenle tek bir uygulama olarak birlikte hareket ederler. Bu uygulamalar, belirli kullanım durumları ve iş süreçlerini desteklerler.

Tüketici katmanı: Sunum katmanı olarak da adlandırılan tüketici katmanı belirli kullanım tercihlerini karşılamak için son kullanıcılara BT fonksiyonları ve veri sunmak için gerekli yetenekleri sağlar. Bu katman aynı zamanda uygulama iletişimleri için bir arayüz sağlar.

Entegrasyon katmanı (kurumsal servis yolu): Entegrasyon katmanı ST'lerin SS'leri bulmasını ve servis etkinleştirilmesini başlatma yeteneğini sağlar. Entegrasyon katmanı arabuluculuk, yönlendirme ve veri, protokol dönümü olmak üzere üç temel yeteneğe sahiptir. Bu yeteneklere dayalı olarak servisler bir iş sürecinin bir parçası olurken birbirleri ile de iletişim kurabilirler. Güvenlik, gecikme ve referans mimarisi, bitişik

katmanlar arasında servis kalitesi gibi fonksiyonel olmayan gereksinimler bu katmanda mimari yapı blokları tarafından uygulanmaktadır.

QAS katmanı (quality of service layer): Bu katman fonksiyonel olmayan gereksinimleri yönetmeye odaklanır. Bunlar güvenlik, kullanılabilirlik, ölçeklenebilirlik ve güvenilirlik olabilir.

Bilgi mimarisi ve iş zekâsı katmanı: Bilgi mimarisi ve iş zekâsı katmanı SOA için gerekli veri ve bilgilerin doğru bir temsilini sağlar. Bu tabakanın veri mimarisi ve her spesifik yatay katmanda bilgi mimarisini temsil etmek gibi sorumlulukları vardır. Ayrıca veri madenciliği ve iş zekâsı için gerekli meta veri saklar.

Yönetim Katmanı: Yönetim tabakası servislerin tüm yaşam döngüsünün doğru yönetimi olmasını sağlar. Hangi değeri yüksek servislerin mimaride katmanların her biri için uygulanması ve servis istemcilerinin bir iş ve IT hedefini karşılamaya dayalı nasıl bir rasyonalizasyon sağlaması gerektiği bu katmanın öncelikli sorumlulukları arasında yer almaktadır. Bu katman çeşitli işletmeler ve BT düzenleyici politikalar ve gereksinimleri ile uyumlu verimli servislerin tasarım ve uygulanmasını denetleyen bir çerçeve sağlar (Arsanjani *et al*. 2014).

2.10. WCF (*Windows Communication Foundation*)

SOA ve servislerinden yararlanılmak isteniyorsa sadece web servislerine takılıp kalmadan daha da ötesine gidilmek zorunlulu vardır. Microsoft .Net Framework 4 ve sonrasında yer alan WCF farklı türlerde ihtiyaç duyulan tüm servislerin inşa edilmesinde karşımıza çıkan engelleri azaltan kapsamlı bir araç kümesini sağlar. SOA sadece web servisleri aracılığıyla birlikte çalışabilirliği sağlamak demek değildir. SOA aynı zamanda duyarlı uygulamalar oluşturabilmek için kabul edilebilir bir performans, web uygulamaları için AJAX desteği, saatler hatta haftalar boyunca uzanan iş süreçlerini etkinleştirmenin de sağlanmasını amaçlar. WCF'in her sürümü tüm bu ihtiyaçları karşılıyor durumdadır (Vogel 2011).

Patil (2010) WCF'i servis odaklı uygulamaları oluşturmak için bir çerçeve olarak tanımlamaktadır. WCF kullanarak, başka bir servisin uç noktasından asenkron mesaj

olarak veri gönderilebilir. Bir servisin uç noktası IIS tarafından barındırılan bir sürekli mevcut bir servisin veya bir uygulamada barındırılan bir servisin bir parçası olabilir. WCF'deki mesajlar XML olarak gönderilen tek bir karakter veya basit bir kelime olabileceği gibi ikili veri akışı kadar karmaşık da olabilir. WCF' in özelliklerini şu şekilde sıralayabiliriz:

- Servis yönlendirmesi
- Birlikte çalışabilirlik
- Çoklu mesaj modelleri
- Servis meta verileri
- Veri sözleşmeleri (Contracts)
- Güvenlik
- Çoklu taşıma ve çözümleme
- Güvenilir ve sıraya alınmış mesajlar
- AJAX and REST desteği (Patil 2010).

SOA uygulamaları oluşturmak için birleşik programlama modeli olan WCF, dağıtık bir ortamda uygulamaların birbirleriyle iletişim kurmasını sağlayan Microsoft'un bir teknolojidir. Farklı platformlar arasında entegre, güvenli, güvenilir işlem çözümleri oluşturur ve geliştiricilere mevcut uygulamalarla birlikte çalışabilme imkânı sağlar. WCF dağıtılmış bilgi işlem, geniş birlikte çalışabilirlik ve servis yönlendirmede doğrudan destek için yönetilebilir bir yaklaşım sunmak üzere tasarlanmıştır. WCF katmanlı bir mimari sunarak dağıtılan uygulama geliştirmenin birçok tipini destekler (Lui 2009).

Dağıtık uygulama geliştirme için Microsoft XML web servisleri, .Net Remoting, COM+, MSMQ gibi sistemleri geliştirmiştir. WCF tam olarak bu sistemlerin hepsinin kabiliyetlerine sahiptir ve tam SOA desteği sağlamaktadır. WCF'in iki önemli özelliği bulunmaktadır. Birincisi Microsoft tarafındaki servislerin farklı platformlar tarafından ele alınabilmesidir. Böylece karmaşık .Net tipleri Java, Com gibi modelleri destekleyen platformlara dağıtılabilirler. Bunun sonucu olarak Linux, Unix gibi sistemler

servislerimizin birer tüketicisi olabilirler. Bu SOA'nın genel yeteneklerinden birisi olan birlikte çalışabilirliği karşılamaktadır. İkinci özellik de Windows üzerinde yapılan dağıtık modeller arasında gerçekleştirilecek entegrasyonların tek bir merkezde toplanmasıdır. Ayrıca WCF service, CLR tiplerini birer servis olarak sunabilmemizi ve servisleri birer CLR tipi olarak kullanabilmeyi sağlayan bir mimari sunmaktadır. WCF'in addresses, contracts ve bindings isimlerinde üç temel bileşeni vardır. Bunlar:

Addresses: Her servis kendine has benzeri başka bir yerde olmayan bir adrese sahip olmalıdır. Bu adres bilgisayar adı, site adı, URL adresleri gibi servisin bulunduğu yer bilgilerini de içerebilir.

Contracts: Bir servisin ne iş yaptığı özellikle istemcilerin ihtiyaç duyduğu proxy sınıflarının yazılması sürecinde önemlidir. WCF servis üzerinde yer alan tüm servisler dış ortama bir contract sunmaktadır.

Bindings: Servisler ile nasıl bağlantı kurulacağını tanımlamaya yarar. Bir bingings tip, taşıma tipi (Transporter type), protokol ve veri çözümlemesi (data encoding) bilgilerini bildirir. Bu bilgiler SOA mimarisinde kullanılacak senaryolar düşünülerek oluşturulur. WCF'deki bağlayıcı tipleri Çizelge 2.2. de gösterildiği gibidir (Osmancık 2013).

Çizelge 2.2. WCF'deki bağlayıcı tipleri (Osmancık 2013)

Binding Tipi	**Konfigürasyon Elementi**	**Taşıma Tipi**	**Veri Çözümlemesi**	**Platform Desteği**
BasicHttpBinding	<basicHttpBinding>	http/https	Text	Var
NetTcpBinding	<netTcpBinding>	TCP	Binary	Yok
NetPeerTcpBinding	<netPeerTcpBinding>	P2P	Binary	Yok
NetNamedPipeBinding	<netNamedPipeBinding>	IPC	Binary	Yok
WSHttpBinding	<wsHttpBinding>	http/https	Text/MTOM	Var
WSFedarationBinding	<wsFedarationBinding>	http/https	Text/MTOM	Var
NetMsmqBinding	<netMsmqBinding>	MSMQ	Binary	Yok
MsmqIntegrationBinding	<msmqIntegrationBinding>	MSMQ	Binary	Var
WSDualHttpBinding	<wsDualHttpBinding>	http	Text/MTOM	Var

2.10.1. WCF mimarisi

Şekil 2.13. WCF mimarisinin gösterilmesi (Anonymous 2016a)

Sözleşmeler: Sözleşmeler mesaj sisteminin çeşitli yönlerini tanımlar. Veri sözleşmesi servisi oluşturmak veya tüketebilmek için gerekli olan her mesajı oluşturan parametreleri açıklar. Mesaj parametreleri XML şema tanımlama dili (XSD) belgeleri ile tanımlanır. Mesaj sözleşmeleri SOAP protokollerini kullanarak belirli mesaj parçaları tanımlar ve birlikte çalışabilirlik talepleri gibi hassas parçaları üzerinde ince taneli kontrolü sağlar. Servis sözleşmesi hizmet asıl yöntem imzalarını belirtir ve Visual Basic veya Visual C # olarak desteklenen programlama dillerinden birinde bir ara birim olarak dağıtılır. Politikalar ve bağlantı bir servis ile iletişim kurmak için gerekli koşulları şart koşmaktadır. Politikalar güvenlik şartlarını ve servis ile iletişim kurmak için yerine getirilmesi gereken diğer durumlardır.

Servis Çalışma Zamanı: Servis çalışma zamanı katmanı servisin yalnızca fiili çalışma sırasında ortaya çıkan davranışları içerir, diğer bir deyişle servis zamanı davranışlarıdır. Daraltma davranışı servis için yapılan talep önceden belirlenen bir sınıra kadar büyümüşse kaç mesajın işleneceğini belirler. Bir hata davranışı, servis üzerinde bir iç hata oluştuğunda ne olacağını belirtir. Meta veri davranışı meta verinin dış dünyaya nasıl sunulduğunu yönetir. Örnek davranış servisin birçok örneklerinin nasıl çalıştırılabildiğini belirtir. İşlem davranışı bir hata oluştuğunda işlem operasyonlarının geri alınmasını sağlar. Sevk davranışı bunun kontrolü bir mesajın WCF altyapısı tarafından nasıl işlendiğini kontrol eder. Genişletilebilirlik çalışma zamanı süreçlerinin özelleştirilmesini sağlar. Parametre filtreleme ileti başlıklarına etki eden filtrelere göre gerçekleşmesi önceden belirlenmiş eylemleri sağlar.

Mesajlaşma: Mesajlar tabakası kanallardan oluşur. Bir kanal, bir şekilde bir ileti işleyen bir bileşendir. Örnek olarak bir ileti kimliğinin doğrulanması verilebilir. Kanal kümesi, aynı zamanda, bir kanal yığın olarak da bilinir. Kanallar mesajlar ve mesaj başlıkları üzerinde çalışır. Bu mesajın organlarının içeriğini işleme konusuyla ilgilenen servis çalışma zamanı katmanından farklıdır. Taşıma kanalları ve protokol kanalları olmak üzere iki türlü kanal vardır. Taşıma kanalları ağdan, dış dünya veya başka bir iletişim noktasından gelen mesajları okur ve yazar. Protokol kanalları sık sık okuyarak ya da mesaja ek başlıklarını yazarak, mesaj işleme protokollerini uygular. Bu tür protokollerin örnekleri WS güvenlik ve WS güvenilirlik içermektedir. Mesajlaşma katmanı olası biçimleri ve veri alışverişi modellerini göstermektedir. WS güvenlik mesaj katmanında güvenliği sağlayan WS güvenlik şartnamesinin bir uygulamasıdır. WS güvenilirlik mesajlaşma kanalı mesaj teslimat garantisini sağlar. Çözümleyici mesajın ihtiyaçlarına uygun olarak kullanılabilen çeşitli çözümlemeler sunar. HTTP HyperText Aktarım Protokolü mesaj teslimi için kullanılan kanalı belirtir. TCP kanalı TCP protokolünü belirtir. İşlem akış kanalı, mesaj desenleri işlemini yönetir. Adlandırılmış yöneltme kanalı kanallar arası iletişimi sağlar. MSMQ kanalı MSMQ uygulamalarının birbirleri ile aralarındaki faaliyetleri sağlar.

Aktivasyon ve barındırma: Servis nihayetinde bir program gibidir. Diğer programlar gibi yürütülebilmesi için çalıştırılması gerekir. Servisler IIS veya windows etkinleştirme hizmeti gibi (WAS) dışarıdan bir etkinin oluşmasıyla bir yerde barındırılır veya çalıştırılır. Servisler manuel olarak da .exe dosyaları olarak çalıştırılabilirler. Bir servis

bir windows servisi olarak otomatik olarak da çalıştırılabilir. COM + bileşenleri de WCF hizmetleri olarak barındırılmış olabilir (Anonymous 2016a)

3. MATERYAL ve YÖNTEM

3.1. Uygulamanın Geliştirildiği Platform ve Programlama Dilleri

Bu çalışma kapsamında yer alan servis ve windows uygulaması Microsoft firmasına ait olan Microsoft Visual Studio Premium 2013 ortamında c# programlama dili kullanılarak geliştirilmiştir. Visual Studio Premium 2013 ortamı, güçlü bir uygulama geliştirme ortamı sağlaması, farklı teknolojileri içerisinde barındırması ve kodlama işleminin kolay olması nedenleriyle tercih edilmiştir. Projede VM için ise yine Microsoft firmasına ait olan Microsoft SQL Server 2014 Management Studio kullanılmıştır.

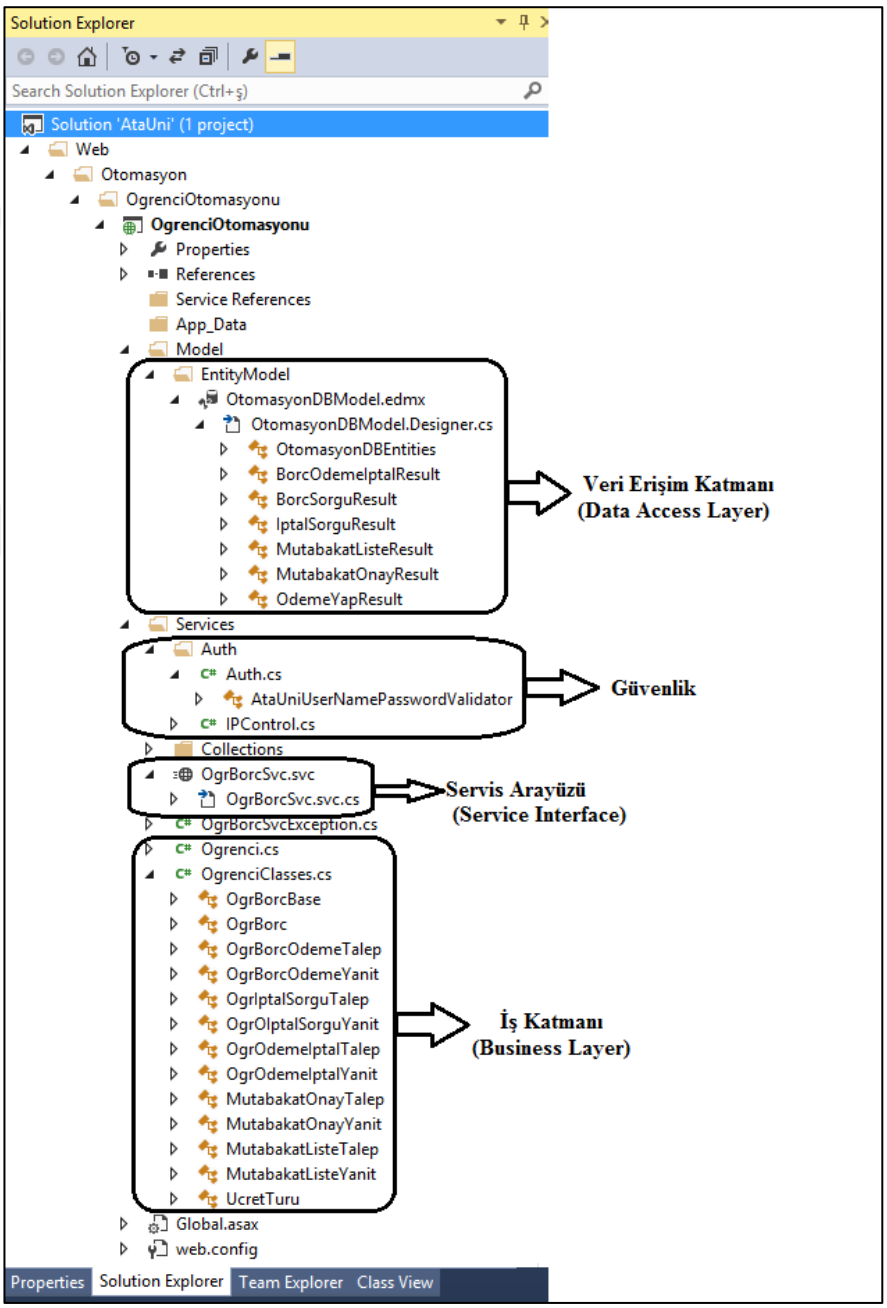
3.2. Projenin Tanıtımı

Proje SOA yaklaşımı benimsenerek inşa edilmiştir. Proje kapsamında ÜBS'de öğrencilerden alınması gereken katkı payı, kart ücreti gibi ödemelerin çevrimiçi (online) olarak alınabilmesi sağlanmıştır. Proje sonucunda öğrencilerin atmler, banka şubeleri, internet bankacılığı aracılığı ile yapacakları ödemeler anlık olarak ilgili veri tabanlarına düşmektedir. Bunun sonucunda öğrenciler yatırdıkları ücretlerin bankalardan üniversitelere excel ya da Word ortamında gönderilmesini ve üniversitelerin de gelen bu listeleri kendi sistemlerine eklemesini günlerce beklemeden ders almalarını ve ücret yatırmalarının ön koşul olduğu diğer işlemlerini hemen yapabileceklerdir. Projenin üç önemli ayağı vardır. Bunlar:

A) WCF teknolojisiyle geliştirilmiş ödeme servisi
B) Kullanıcıların uygulamayı kullanabilmelerini sağlayan windows arayüzü
C) Tablolar, stored procedurlerin yer aldığı VM kısmı

3.3. WCF Teknolojisiyle Geliştirilmiş Ödeme Servisi

WCF teknolojisiyle ödeme işlemleri için geliştirilen servisin genel görünümü Şekil 3.1'de görüldüğü gibidir.



Şekil 3.1. WCF ödeme servisinin genel görünümü

3.3.1. Ödeme servisindeki metotlar

Microsoft WCF teknolojisi kullanarak oluşturulan servisimizde borç sorgulama, borç ödeme, ödemeyi iptal etme, mutabakat işlemleri gibi kriterler göz önünde bulundurularak aşağıdaki metotlar tanımlanmıştır:

BorcSorgula() : Bankalardan, kredi kartı ödeme kısmından, Windows uygulamalardan servise borç sorgu için gelecek sorgularda bu metot (Şekil 3.2) kullanılmaktadır.

```
    [OperationContract(Name = "BorcSorgula"),
    FaultContract(typeof(OgrBorcSvcException)),
    WebGet(ResponseFormat = WebMessageFormat.Xml, UriTemplate =
"BorcSorgula/{OgrNo}")]

     public Ogrenci BorcSorgula(string OgrNo, int BankaKodu)
   {
     if (ServiceSecurityContext.Current.IsAnonymous)
       throw new FaultException<OgrBorcSvcException>(new OgrBorcSvcException(1));

     return fxBorcSorgula(OgrNo, BankaKodu );    }
```

Şekil 3.2. BorcSorgula () metodu

BorcOdeme() : Borç sorgulama yapıldıktan sonra ilgili borcun ödenmesi için BorcOdeme() metodunun çağrılması (Şekil 3.3) gerekmektedir.

```
[OperationContract(Name = "BorcOdeme"),
   FaultContract(typeof(OgrBorcSvcException)),
   WebInvoke(ResponseFormat = WebMessageFormat.Xml, UriTemplate =
"BorcOdeme/{OgrNo}/", Method = "POST")]
   public Ogrenci BorcOdemeYap(BorcOdemeTalepCollection odemeTalep)
   {
      if (ServiceSecurityContext.Current.IsAnonymous)
         throw new FaultException<OgrBorcSvcException>(new OgrBorcSvcException(1));
      if (odemeTalep == null) throw new ArgumentException("odemeTalep parametresi
girilmemis", "odemeTalep");
      return fxBorcOdemeYap(borcOdemeTalepColl: odemeTalep);    }
```

Şekil 3.3. BorcOdeme () metodu

IptalSorgu() : Alınan ödemenin herhangi bir nedenden dolayı iptal edilmek istenmesi durumunda bu metot (Şekil 3.4) çağrılır. Metoddan iptalin gerçekleştirilip gerçekleştirilemeyeceği değeri döndürülür.

```
[OperationContract(Name = "IptalSorgu"),
    FaultContract(typeof(OgrBorcSvcException)),
    WebInvoke(ResponseFormat = WebMessageFormat.Xml, UriTemplate =
"IptalSorgu/{OgrNo}/", Method = "POST")]
public Ogrenci IptalSorgu(IptalSorguTalepCollection iptalSorguTalep)
 {
   if (ServiceSecurityContext.Current.IsAnonymous)
      throw new FaultException<OgrBorcSvcException>(new OgrBorcSvcException(1));
   if (iptalSorguTalep == null)
      throw new ArgumentException("IptalSorguTalep parametresi girilmemis",
"sorguIptalTalep");

      return fxIptalSorgu(iptalSorguTalepColl: iptalSorguTalep);
   }
```

Şekil 3.4. IptalSorgu () metodu

OdemeIptal() : İptal sorgu sonucunda ödeme iptal edilebilir diye bir cevap dönmüşse, ödemenin iptali için bu metot (Şekil 3.5) çağrılır.

```
[OperationContract(Name = "OdemeIptal"),
    FaultContract(typeof(OgrBorcSvcException)),
    WebInvoke(ResponseFormat = WebMessageFormat.Xml, UriTemplate =
"OdemeIptal/{OgrNo}/", Method = "POST")]
   public Ogrenci OdemeIptal(OdemeIptalTalepCollection odemeIptalTalep)
   {
      if (ServiceSecurityContext.Current.IsAnonymous)
         throw new FaultException<OgrBorcSvcException>(new OgrBorcSvcException(1));

      if (odemeIptalTalep == null) throw new ArgumentException("odemeIptalTalep parametresi
girilmemis", "odemeIptalTalep");

      return fxOdemeIptalYap(odemeIptalTalepColl: odemeIptalTalep);
   }
```

Şekil 3.5. OdemeIptal () metodu

MutabakatOnay() : Bankalarla üniversite arasında toplam tahsilat sayısı, toplam tahsilat miktarı, iptal edilen ödeme sayısı ve miktarları göz önünde bulundurularak mutabakatın sağlanıp sağlanmadığı bilgisi için bu metot (Şekil 3.6) kullanılır.

```
[OperationContract(Name = "MutabakatOnay"),
 FaultContract(typeof(OgrBorcSvcException)),
 WebInvoke(ResponseFormat = WebMessageFormat.Xml, UriTemplate =
"MutabakatOnay/{OgrNo}/", Method = "POST")]
   public Ogrenci MutabakatOnay(MutabakatOnayTalepCollection mutabakatOnayTalep)
   {
      if (ServiceSecurityContext.Current.IsAnonymous)
         throw new FaultException<OgrBorcSvcException>(new OgrBorcSvcException(1));

      if (mutabakatOnayTalep == null) throw new ArgumentException("MutabakatOnay
parametresi girilmemis", "mutabakatOnayTalep");

      return fxMutabakatOnay(mutabakatTalepColl: mutabakatOnayTalep);
   }
```

Şekil 3.6. MutabakatOnay () metodu

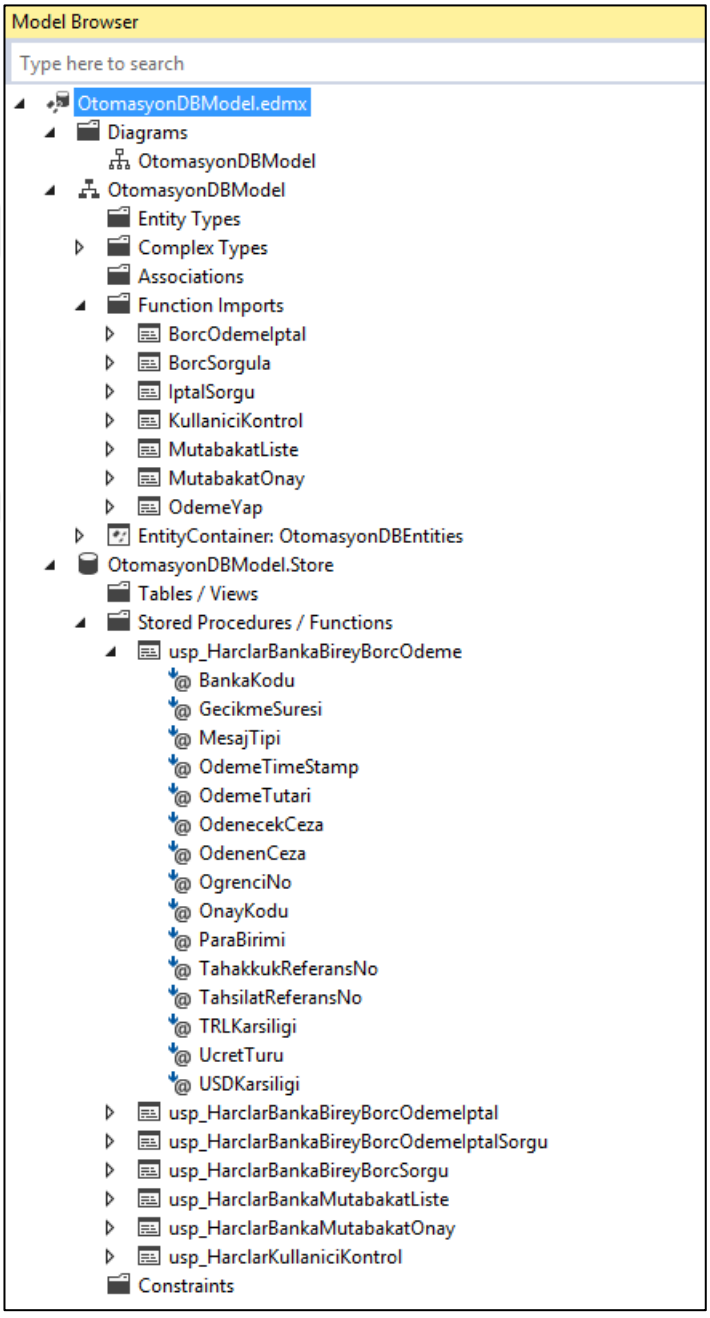
MutabakatListe() : Mutabakat sağlanamaması durumunda mutabakat listesi için bu metot (Şekil 3.7) çağrılır.

```
  [OperationContract(Name = "MutabakatListe"),
 FaultContract(typeof(OgrBorcSvcException)),
 WebInvoke(ResponseFormat = WebMessageFormat.Xml, UriTemplate =
"MutabakatListe/{OgrNo}/", Method = "POST")]
   public Ogrenci MutabakatListe(MutabakatListeTalepCollection mutabakatListeTalep)
   {
      if (ServiceSecurityContext.Current.IsAnonymous)
         throw new FaultException<OgrBorcSvcException>(new OgrBorcSvcException(1));

      if (mutabakatListeTalep == null) throw new ArgumentException("MutabakatListe
parametresi girilmemis", "mutabakatListeTalep");

      return fxMutabakatListe(mutabakatListTalepColl: mutabakatListeTalep);
   }
```

Şekil 3.7. MutabakatListe () metodu

3.3.2. Ödeme servisinde entity model

Entity model geliştirilen WCF servisinin veri erişim (Data Access Layer) katmanıdır. Sadece veri tabanına erişimi sağlamakla sorumlu olan bu katman (Şekil 3.8) veri tabanıyla iletişimi gerçekleştiren temel bileşenleri çağırmaktadır.



Şekil 3.8. Ödeme servisi entity modeli

3.3.3. Servis Yetki Kontrolü

Bankalar için daha önceden tanımlanmış ve veri tabanında ilgili tabloda tutulan kullanıcı adı ve şifre bilgileri serviste Auth kısmında Auth.cs ‘de (Şekil 3.9) kontrol edilir. İstenirse ipcontrol.cs de ip kontolü de yapılabilir.

```
 public override void Validate(string userName, string password)
{   if (null == userName || null == password)
          throw new ArgumentNullException();
      var ctx = new Model.EntityModel.OtomasyonDBEntities();
  if (!ctx.KullaniciKontrol(userName, password).Single().Value)
          throw new FaultException("Hatalı Kullanıcı Adı veya Parolası!", new
FaultCode("932"));   }
```

Şekil 3.9. Kullanıcı adı ve şifre kontrolü yapan kod

3.3.4. Serviste bulunan sınıflar (class)

ogrBorc () : Öğrenci borçlarını listeyelen sınıftır (Şekil 3.10).

```
[DataContract]
  public class OgrBorc : OgrBorcBase
{  [DataMember]
    public string OgrenciNo { get; set; }
    [DataMember]
    public string Ad { get; set; }
    [DataMember]
    public string Soyad { get; set; }
    [DataMember]
    public decimal Borc { get; set; }
    [DataMember]
    public string SonOdemeTarihi { get; set; }
    [DataMember]
    public decimal MinimumOdemeMiktari { get; set; }
    [DataMember]
    public string BorcAciklama { get; set; }; }
```

Şekil 3.10. ogrBorc () sınıfı

OgrBorcOdemeTalep () : Banka tarafından ödeme yapılmak istendiğinde çağrılan sınıftır (Şekil 3.11).

```
public class OgrBorcOdemeTalep : OgrBorcBase
 { public OgrBorcOdemeTalep(): base()    {     }
   [DataMember]
   public string OgrenciNo { get; set; }
   [DataMember]
   public string MesajTipi { get; set; }
   [DataMember]
   public string TahsilatReferansNo { get; set; }
   [DataMember]
   public string OdemeTimeStamp { get; set; }
   [DataMember]
   public decimal OdemeTutari { get; set; }
   [DataMember]
   public int? GecikmeSuresi { get; set; }
   [DataMember]
   public decimal? OdenenCeza { get; set; }
   [DataMember]
   public string OnayKodu { get; set; }
   [DataMember]
   public int BankaKodu { get; set; }  }
```

Şekil 3.11. OgrBorcOdemeTalep () sınıfı

OgrBorcOdemeYanit (): Bankaların borç ödeme isteklerine verilen yanıtı belirleyen sınıftır (Şekil 3.12).

```
public class OgrBorcOdemeYanit : OgrBorcBase
 { public OgrBorcOdemeYanit() : base()    {    }
   [DataMember]
   public string OgrenciNo { get; set; }
   [DataMember]
   public string MesajStatusu { get; set; }
   [DataMember]
   public string TahsilatReferansNo { get; set; }
   [DataMember]
   public string OdemeTimeStamp { get; set; }
   [DataMember]
   public decimal OdemeTutari { get; set;  }
```

Şekil 3.12. OgrBorcOdeme () sınıfı

OgrIptalSorguTalep () : Ödemesi gerçekleştirilen bir tahsilatın iptal edilip edilemeyeceğinin sorgulanması için kullanılan sınıftır (Şekil 3.13).

```
[DataContract]
 public class OgrIptalSorguTalep : OgrBorcBase
 {
    public OgrIptalSorguTalep()
       : base()
    {     }

    [DataMember]
    public string OgrenciNo { get; set; }
    [DataMember]
    public string TahsilatReferansNo { get; set; }
    [DataMember]
    public string OdemeTimeStamp { get; set; }
    [DataMember]
    public decimal OdemeTutari { get; set; }
    [DataMember]
    public int BankaKodu { get; set; }
 }
```

Şekil 3.13. OgrIptalSorguTalep () sınıfı

OgrOIptalSorguYanit () : İptal sorguları için bankalara verilen yanıt için kullanılan sınıftır. (Şekil 3.14)

```
 public class OgrOIptalSorguYanit : OgrBorcBase
 {    public OgrOIptalSorguYanit()
       : base()
    {  }
    [DataMember]
    public string MesajStatusu { get; set; }
    [DataMember]
    public string TahsilatReferansNo { get; set; }
    [DataMember]
    public string OdemeTimeStamp { get; set; }
    [DataMember]
    public decimal OdemeTutari { get; set; }
    [DataMember]
    public string OnayKodu { get; set; }
    [DataMember]
    public string OgrenciNo { get; set; }   }
```

Şekil 3.14. OgrOIptalSorguYanit () sınıfı

OgrOdemeIptalTalep () : Ödemesi gerçekleştirilen bir borcun iptal edilmek istendiğinde kullanılan sınıftır (Şekil 3.15).

```
public class OgrOdemeIptalTalep : OgrBorcBase
 {  public OgrOdemeIptalTalep()
        : base()      {      }
    [DataMember]
    public string OgrenciNo { get; set; }
    [DataMember]
    public string MesajTipi { get; set; }
    [DataMember]
    public string TahsilatReferansNo { get; set; }
    [DataMember]
    public string OdemeTimeStamp { get; set; }
    [DataMember]
    public decimal OdemeTutari { get; set; }
    [DataMember]
    public int? GecikmeSuresi { get; set; }
    [DataMember]
    public string OnayKodu { get; set; }
    [DataMember]
    public int BankaKodu { get; set; } }
```

Şekil 3.15. OgrOdemeIptalTalep () sınıfı

OgrOdemeIptalYanit () : Ödeme iptal talebine verilecek yanıt için kullanılan sınıftır (Şekil 3.16).

```
public class OgrOdemeIptalYanit : OgrBorcBase
 {  public OgrOdemeIptalYanit()
        : base()
    {      }
    [DataMember]
    public string OgrenciNo { get; set; }
    [DataMember]
    public string MesajStatusu { get; set; }
    [DataMember]
    public string TahsilatReferansNo { get; set; }
    [DataMember]
    public string OdemeTimeStamp { get; set; }
    [DataMember]
    public decimal OdemeTutari { get; set; }  }
```

Şekil 3.16. OgrOdemeIptalYanit () sınıfı

MutabakatOnayTalep () : Mutabakatın gerçekleştirilip gerçekleştirilmediğinin bankalar tarafından sorgulanması için kullanılan sınıftır (Şekil 3.17).

```
public class MutabakatOnayTalep
{  public MutabakatOnayTalep()    : base()
   {     }
   [DataMember]
   public string BankaMutabakatTarihi { get; set; }
   [DataMember]
   public int BankaTopIptalAdet { get; set; }
   [DataMember]
   public decimal BankaTopIptalTutar { get; set; }
   [DataMember]
   public int BankaTopTahsilatAdet { get; set; }
   [DataMember]
   public decimal BankaTopTahsilatTutar { get; set; }
   [DataMember]
   public int BankaKodu { get; set; }  }
```

Şekil 3.17. MutabakatOnayTalep () sınıfı

MutabakatOnayYanit () : Mutabakat onay sorgusuna verilecek yanıt için kullanılan sınıftır (Şekil 3.18).

```
[DataContract]
 public class MutabakatOnayYanit
 {
    public MutabakatOnayYanit()
       : base()
    {     }

    [DataMember]
    public string UnivMutabakatTarihi { get; set; }
    [DataMember]
    public int UnivTopIptalAdet { get; set; }
    [DataMember]
    public decimal UnivTopIptalTutar { get; set; }
    [DataMember]
    public int UnivTopTahsilatAdet { get; set; }
    [DataMember]
    public decimal UnivTopTahsilatTutar { get; set; }  }
```

Şekil 3.18. MutabakatOnayYanit () sınıfı

MutabakatListeTalep () : Mutabakatın gerçekleştirilememesi durumunda bankalar tarafından mutabakat listesi istemek için kullanılan sınıftır (Şekil 3.19).

```
[DataContract]
public class MutabakatListeTalep
{   public MutabakatListeTalep()
      : base()
    {     }
    [DataMember]
    public string BankaMutabakatTarihi { get; set; }  }
```

Şekil 3.19. MutabakatListeTalep () sınıfı

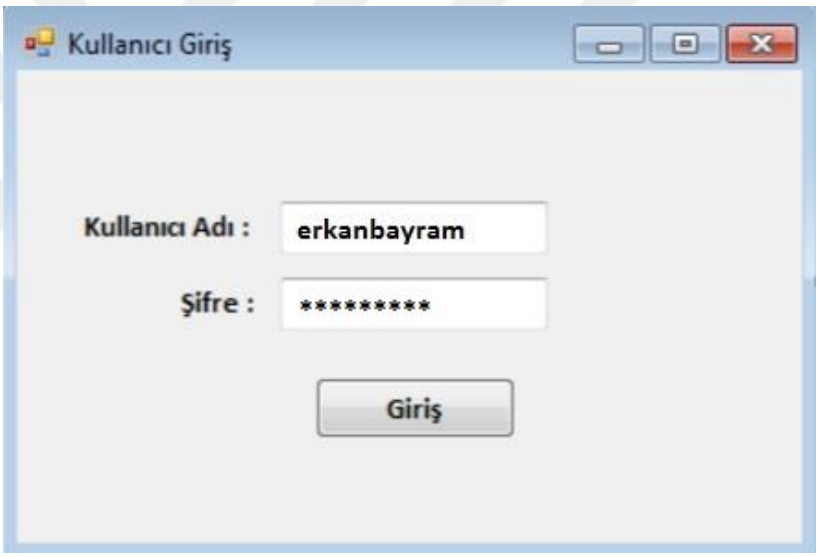
MutabakatListeYanit () : Mutabakat liste isteğine verilecek yanıt için kullanılan sınıftır (Şekil 3.20).

```
[DataContract]
 public class MutabakatListeYanit
 {  public MutabakatListeYanit()
      : base()
    {     }
    [DataMember]
    public string Ad { get; set; }
    [DataMember]
    public string Soyad { get; set; }
    [DataMember]
    public string OgrenciNo { get; set; }
    [DataMember]
    public string TahakkukReferansNo { get; set; }
    [DataMember]
    public string TahsilatReferansNo { get; set; }
    [DataMember]
    public string UcretTuru { get; set; }
    [DataMember]
    public string IslemZamani { get; set; }
    [DataMember]
    public decimal OdemeTutari { get; set; }
    [DataMember]
    public string OnayKodu { get; set; }
    [DataMember]
    public string IslemTipi { get; set; }  }
```

Şekil 3.20. MutabakatListeYanit () sınıfı

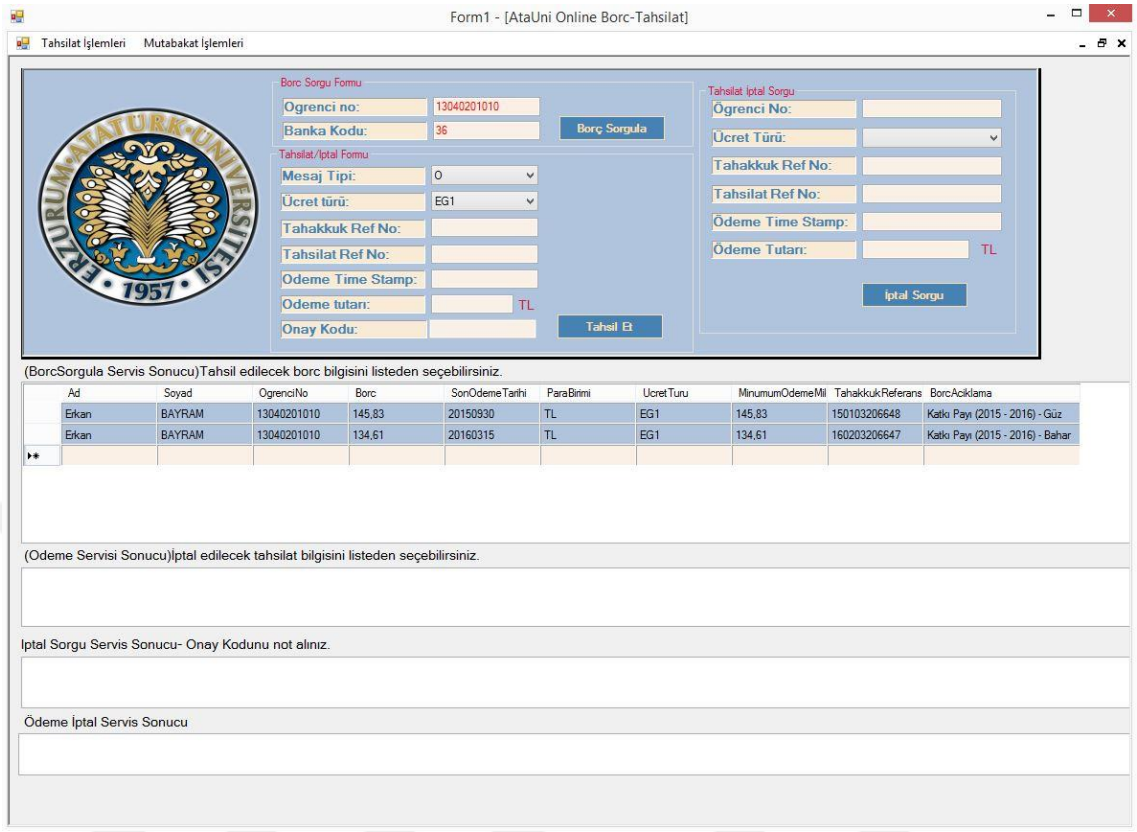
3.4. Windows Uygulaması

Ödeme işlemlerini bankalar almaktadır, her bankanın kendine has bir arayüzü mevcuttur. Oluşturulan bu windows uygulaması sanki herhangi bir bankadan ödeme alınıyormuş gibi düşünülerek tasarlandı. Windows uygulamasına giriş yapılabilmesi için öncelikli olarak daha önceden kullanıcılara verilmiş olan kullanıcı adı ve şifre bilgilerinin login ekranında (Şekil 3.21) doğru bir şekilde girilmesi gerekmektedir. Doğru değerleri giremeyen kullanıcılara yanlış kullanıcı adı veya şifre uyarısı verilmektedir.



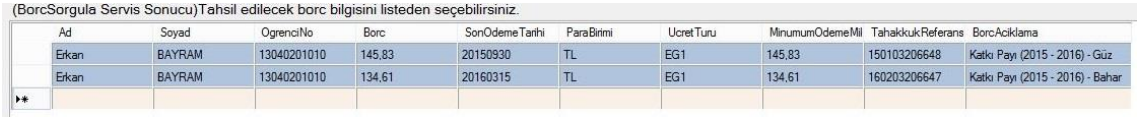
Şekil 3.21. Uygulama arayüzü için login ekranı

Kullanıcı adı ve şifre doğru girildikten sonra uygulamaya giriş yapılmış olacaktır. Uygulama ekranı (Şekil 3.22) tahsilat işlemleri ve mutabakat işlemleri diye iki kısma ayrılmıştır. Tahsilat işlemleri kısmında borç sorgulama, ödeme gerçekleştirme ve ödeme iptal işlemleri yapılabilmektedir.



Şekil 3.22. Borç sorgulama - ödeme gerçekleştirme - ödeme iptal ekranı

Ödeme ekranında öğrenci no yazan kutucuğa öğrencinin numarası, banka kodu yazan kutucuğa da daha önce bankalara verilen banka kodlarına göre uygun banka kodu girilir. Banka kodunu ödemenin hangi bankadan gerçekleştirildiği bilgisini tespit edebilmek için kullanıyoruz. Bu iki veri girildikten sonra borç sorgula butonu tıklanınca koddan servise öğrenci numarası ve banka kodu parametre olarak gönderilir ve daha sonra servisden dönen öğrenciye ait borç bilgileri ekrana basılır (Şekil 3.23).



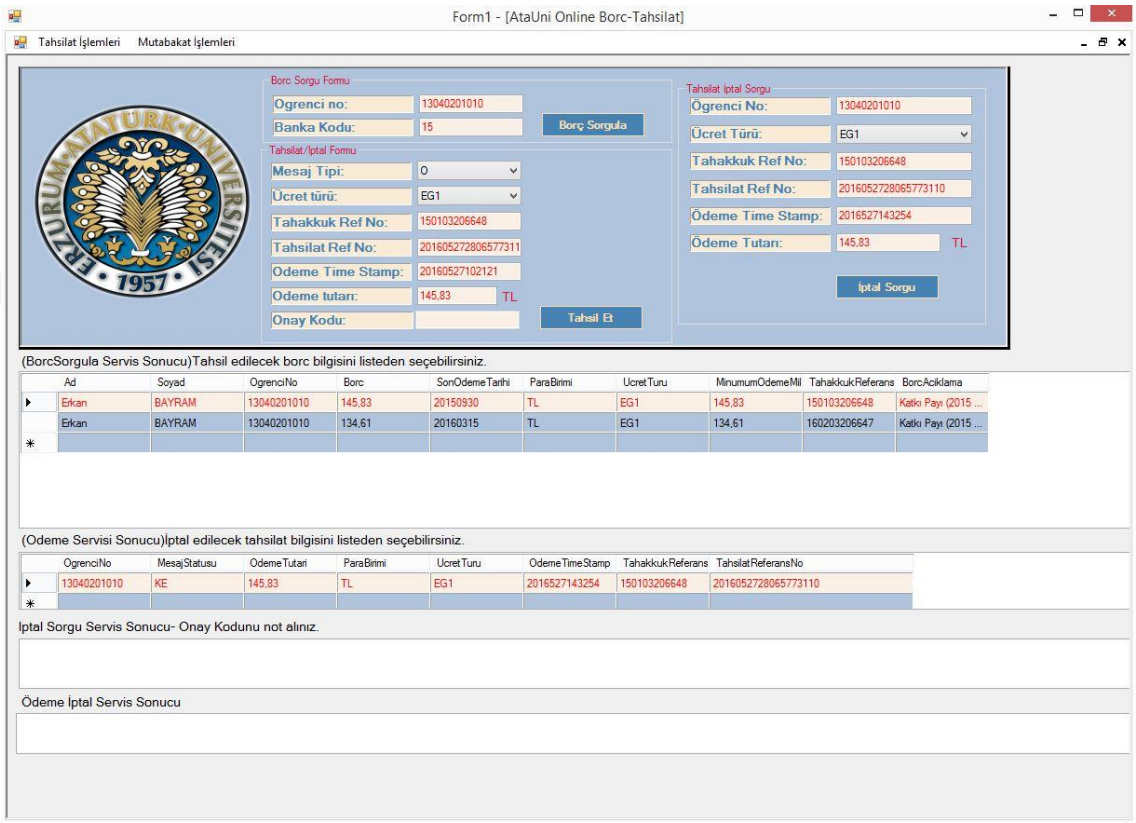
Şekil 3.23. Borçların listelenmesi

Listelenen borçlardan ödemesi yapılmak istenenin üzerinde çift tıklanınca yukarıdaki ilgili alanlar otomatik olarak doldurulacaktır. Borç sorgula butonunun arkasında çalışan ***private void fn_BorcSorgula()*** fonksiyonu içerisinde (Şekil 3.24) öğrenci no ve banka kodu alanlarının girilmesi zorunlu tutulmuştur.

```
private void fn_BorcSorgula()
{  dataGridBorc.DataSource = null;
if (txtOgrNo.Text == "") MessageBox.Show("Öğrenci numarasını boş geçmeyiniz");
 else
 {  ogrSvc.Ogrenci ogr = client.BorcSorgula(txtOgrNo.Text.Trim()
,Convert.ToInt32(txtBankaKodu.Text.Trim()));
  DataTable dt = new DataTable();
  DataRow dr;
  dt.Columns.Add(new DataColumn("Ad", typeof(string)));
  dt.Columns.Add(new DataColumn("Soyad", typeof(string)));
  dt.Columns.Add(new DataColumn("OgrenciNo", typeof(string)));
  dt.Columns.Add(new DataColumn("Borc", typeof(string)));
  dt.Columns.Add(new DataColumn("SonOdemeTarihi", typeof(string)));
  dt.Columns.Add(new DataColumn("ParaBirimi", typeof(string)));
  dt.Columns.Add(new DataColumn("UcretTuru", typeof(string)));
  dt.Columns.Add(new DataColumn("MinumumOdemeMiktari", typeof(string)));
  dt.Columns.Add(new DataColumn("TahakkukReferansNo", typeof(string)));
  dt.Columns.Add(new DataColumn("BorcAciklama", typeof(string)));
if (ogr.HataKodu != 0)
MessageBox.Show("Hata Kodu:"+ogr.HataKodu.ToString()+" - "+ogr.HataKodAciklama);
 for (int i = 0; i < ogr.Borclar.Count; i++)
      {
   dr = dt.NewRow();
   dr[0] = ogr.Borclar[i].Ad;
   dr[1] = ogr.Borclar[i].Soyad;
   dr[2] = ogr.Borclar[i].OgrenciNo;
   dr[3] = ogr.Borclar[i].Borc;
   dr[4] = ogr.Borclar[i].SonOdemeTarihi;
   dr[5] = ogr.Borclar[i].ParaBirimi;
   dr[6] = ogr.Borclar[i].UcretTuru;
   dr[7] = ogr.Borclar[i].MinimumOdemeMiktari;
   dr[8] = ogr.Borclar[i].TahakkukReferansNo;
   dr[9] = ogr.Borclar[i].BorcAciklama;
   dt.Rows.Add(dr);
      }
      DataView dv = new DataView(dt);
      dataGridBorc.DataSource = dv;
      }
  }
```

Şekil 3.24. Borç sorgu yapan kod kısmı

Ödemeyi alacak olan personel tahsilat referans numarasını, ödemetimestamp bilgilerini girdikten sonra tahsil et butonu tıklayacaktır. Bu işlemlerden sonra ekran Şekil 3.25'teki gibi dolacaktır.



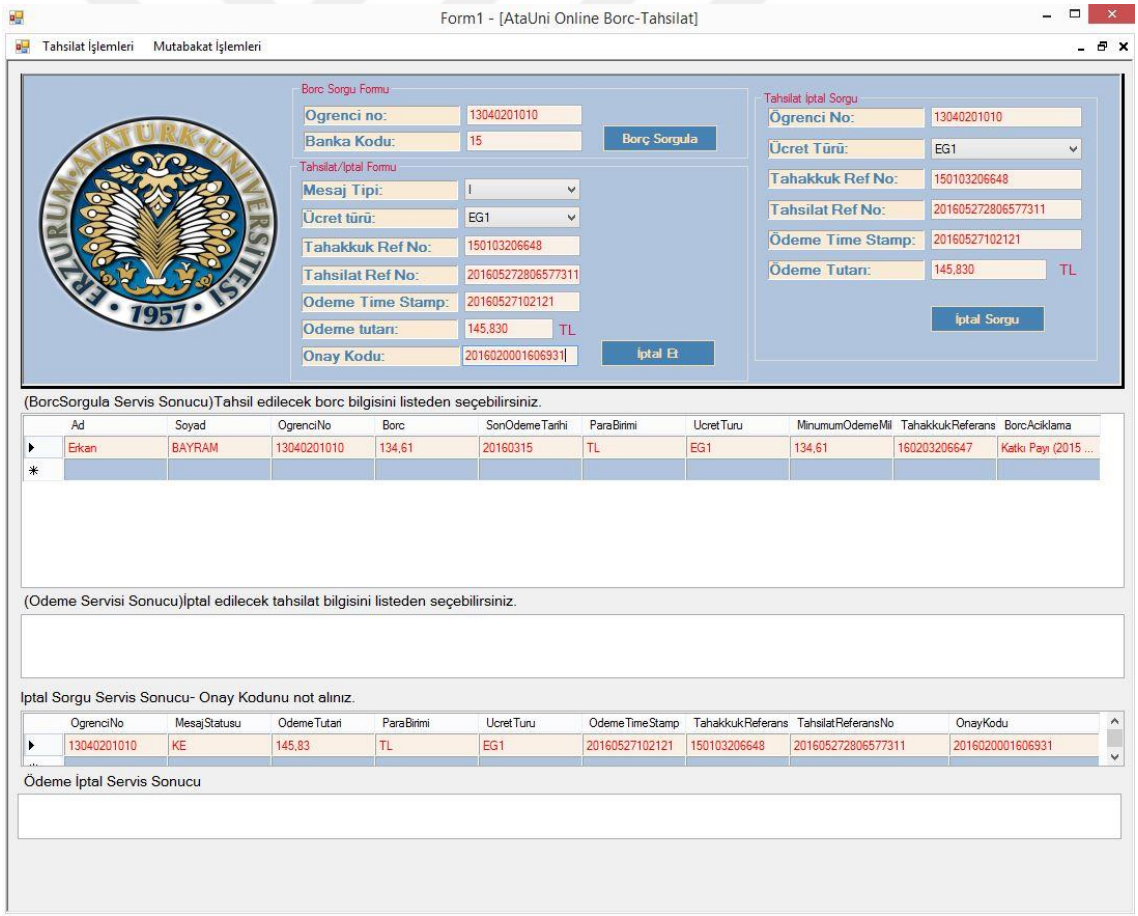
Şekil 3.25. Tahsilat alma durumu

Yukarıda da görülebileceği gibi ödeme alındıktan sonra alınan ödemenin bilgileri 'Ödeme servisi sonucu' alanına düşmüştür. Dikkat edilirse mesaj tipi alanında ödemenin gerçekleşeceğini belirten 'O' harfi ve istenirse gerçekleşen ödemenin iptal edilmesini sağlayacak 'I' harfi yer almaktadır. Hangi işlemi yapmak istiyorsak bu alandan ilgili harfi seçmeliyiz. Ödeme gerçekleşmesi durumunda kod tarafında ***private void fn_BorcOdeme()*** fonksiyonunun içerisinde (Şekil 3.26) ödemeyle ilgili kod parçası çalışacaktır ve ödeme gerçekleştirilecektir.

```
private void fn_BorcOdeme(){
  if (OdemeTutari.Text == "") OdemeTutari.Text = "0";
  if (OdemeTutari.Text == "0") MessageBox.Show("Ödeme tutarını boş geçmeyiniz");
  else if (txtTahkRefNo.Text=="")
    MessageBox.Show("Tahakkuk referans numarasını boş geçmeyiniz");
  else if (txtTahsilatRefNo.Text == "")
    MessageBox.Show("Tahsilat referans numarasını boş geçmeyiniz");
  Else
  { if (cmbMsj.Text.Trim() == "O")  {
  testHarcServis.ogrSvc.BorcOdemeTalepCollection   odemeler = new
testHarcServis.ogrSvc.BorcOdemeTalepCollection();
testHarcServis.ogrSvc.OgrBorcOdemeTalep odeme = new
testHarcServis.ogrSvc.OgrBorcOdemeTalep()
  { OgrenciNo = txtOgrNo.Text.Trim(),
    MesajTipi = cmbMsj.Text.Trim(),
    OdemeTimeStamp = txtOdemeTimeStamp.Text.Trim(),
    OdemeTutari = Convert.ToDecimal(OdemeTutari.Text.Trim()),
    ParaBirimi = "TL",
    TahakkukReferansNo = txtTahkRefNo.Text.Trim(),
    TahsilatReferansNo = txtTahsilatRefNo.Text.Trim(),
    UcretTuru = cmbUcret.Text.Trim(),
    OnayKodu = txtOnayKod.Text.Trim(),
    BankaKodu = Convert.ToInt32(txtBankaKodu.Text.Trim())
  };  odemeler.Add(odeme);
  ogrSvc.Ogrenci ogr = client.BorcOdeme(odemeler);
  DataTable dt = new DataTable();  DataRow dr;
  dt.Columns.Add(new DataColumn("OgrenciNo", typeof(string)));
  dt.Columns.Add(new DataColumn("MesajStatusu", typeof(string)));
  dt.Columns.Add(new DataColumn("OdemeTutari", typeof(string)));
  dt.Columns.Add(new DataColumn("ParaBirimi", typeof(string)));
  dt.Columns.Add(new DataColumn("UcretTuru", typeof(string)));
  dt.Columns.Add(new DataColumn("OdemeTimeStamp", typeof(string)));
  dt.Columns.Add(new DataColumn("TahakkukReferansNo", typeof(string)));
  dt.Columns.Add(new DataColumn("TahsilatReferansNo", typeof(string)));
  if (ogr.HataKodu != 0)
  MessageBox.Show("Hata Kodu: " + ogr.HataKodu.ToString() + " - " +
ogr.HataKodAciklama);
   for (int i = 0; i < ogr.Odemeler.BorcTahsilatlari.Count; i++)
  { dr = dt.NewRow();
     dr[0] = ogr.Odemeler.BorcTahsilatlari[i].OgrenciNo;
     dr[1] = ogr.Odemeler.BorcTahsilatlari[i].MesajStatusu;
     dr[2] = ogr.Odemeler.BorcTahsilatlari[i].OdemeTutari;
     dr[3] = ogr.Odemeler.BorcTahsilatlari[i].ParaBirimi;
     dr[4] = ogr.Odemeler.BorcTahsilatlari[i].UcretTuru;
     dr[5] = ogr.Odemeler.BorcTahsilatlari[i].OdemeTimeStamp;
     dr[6] = ogr.Odemeler.BorcTahsilatlari[i].TahakkukReferansNo;
     dr[7] = ogr.Odemeler.BorcTahsilatlari[i].TahsilatReferansNo;
     dt.Rows.Add(dr);
    } DataView dv = new DataView(dt); dataGridViewOdeme.DataSource = dv;   }
```

Şekil 3.26. Ödemeyi gerçekleştiren kod kısmı

Kodda da görüleceği üzere servise ilgili parametreler gönderiliyor ve servisten hata kodu olarak sıfır değeri dönüyorsa ödeme gerçekleştirilmiş oluyor. Eğer yapılan ödeme işleminin yanlış olduğu düşünülüyorsa ya da ödemeyi yapan kişi ücreti geri isteyip ödeme işleminin iptal edilmesini istiyorsa, iptal işlemini gerçekleştirebilmek için iki yöntem vardır. Ödemeden hemen sonra sayfayı yenilemeden iptal işlemi gerçekleştirilecekse mesaj tipi alanında 'I' harfi seçilir ve İptal Et butonuna basılır. Diğer bir yöntem ise sayfa yenilenmiş ve 'ödeme servis sonucu' alanı boşaltılmış ise sağ tarafta yer alan 'Tahsilat iptal Sorgu' alanındaki ilgili alanlar doldurulur ve İptal Sorgu butonuna basılarak iptal edilmek istenen tahsilat bulunur. İptal sorgu butonu tıklanınca ekran Şekil 3.27'deki gibi doldurulacaktır.



Şekil 3.27. İptal sorgu sonucu ekran görüntüsü

Görüleceği üzere iptal sorgu servis sonucu alanına iptal edilebilecek kaydın bilgileri düşmüştür. Öğrenci ücreti yatırıp ders alma yapmışsa ve ücretini de ders alma yaptıktan sonra geri almak istiyorsa, sistem iptal işlemine izin vermeyecek ve öğrenci ders alma yapmış diye hata kodu dönecektir. İptal sorgusunu gerçekleştirip sonucu ekrana basan kod kısmı Şekil 3.28'deki gibidir.

```
private void fn_IptalSorgu() {
  if (IOdemeTutari.Text == "") IOdemeTutari.Text = "0";
  if (IOdemeTutari.Text=="0") MessageBox.Show("Ödeme tutarını boş geçilemez");
   else if (txtITahkRefNo.Text == "")
      MessageBox.Show("Tahakkuk referans numarasını boş geçmeyiniz");
   else if (txtITahsilatRefNo.Text == "")
      MessageBox.Show("Tahsilat referans numarasını boş geçmeyiniz");
   else {
   testHarcServis.ogrSvc.IptalSorguTalepCollection
                iptalSorguTalep = new testHarcServis.ogrSvc.IptalSorguTalepCollection();
    testHarcServis.ogrSvc.OgrIptalSorguTalep
                iptalSorgu = new testHarcServis.ogrSvc.OgrIptalSorguTalep()
   { OgrenciNo = txtIOgrNo.Text.Trim(),
     OdemeTimeStamp = txtIOdemeTimeStamp.Text.Trim(),
     OdemeTutari = Convert.ToDecimal(IOdemeTutari.Text.Trim()),
     ParaBirimi = "TL",
     TahakkukReferansNo = txtITahkRefNo.Text.Trim(),
     TahsilatReferansNo = txtITahsilatRefNo.Text.Trim(),
     UcretTuru = cmbIUcret.Text.Trim(),
     BankaKodu =  Convert.ToInt32(txtBankaKodu.Text.Trim())
    };
    iptalSorguTalep.Add(iptalSorgu);
    ogrSvc.Ogrenci ogr = client.IptalSorgu(iptalSorguTalep);
    DataTable dt = new DataTable();    DataRow dr;
    dt.Columns.Add(new DataColumn("OgrenciNo", typeof(string)));
    dt.Columns.Add(new DataColumn("MesajStatusu", typeof(string)));
    dt.Columns.Add(new DataColumn("OdemeTutari", typeof(string)));
    dt.Columns.Add(new DataColumn("ParaBirimi", typeof(string)));
    dt.Columns.Add(new DataColumn("UcretTuru", typeof(string)));
    dt.Columns.Add(new DataColumn("OdemeTimeStamp", typeof(string)));
    dt.Columns.Add(new DataColumn("TahakkukReferansNo", typeof(string)));
    dt.Columns.Add(new DataColumn("TahsilatReferansNo", typeof(string)));
    dt.Columns.Add(new DataColumn("OnayKodu", typeof(string)));
 if (ogr.HataKodu != 0)
    MessageBox.Show("Hata Kodu: " + ogr.HataKodu.ToString()+" - +ogr.HataKodAciklama);
  for (int i = 0; i < ogr.IptalSorgular.IptalSorgu.Count; i++)
```

Şekil 3.28. İptal sorgusunu yapan kod parçacığı

```
{ dr = dt.NewRow();
    dr[0] = ogr.IptalSorgular.IptalSorgu[i].OgrenciNo;
    dr[1] = ogr.IptalSorgular.IptalSorgu[i].MesajStatusu;
    dr[2] = ogr.IptalSorgular.IptalSorgu[i].OdemeTutari;
    dr[3] = ogr.IptalSorgular.IptalSorgu[i].ParaBirimi;
    dr[4] = ogr.IptalSorgular.IptalSorgu[i].UcretTuru;
    dr[5] = ogr.IptalSorgular.IptalSorgu[i].OdemeTimeStamp;
    dr[6] = ogr.IptalSorgular.IptalSorgu[i].TahakkukReferansNo;
    dr[7] = ogr.IptalSorgular.IptalSorgu[i].TahsilatReferansNo;
    dr[8] = ogr.IptalSorgular.IptalSorgu[i].OnayKodu;
    dt.Rows.Add(dr);   }
   DataView dv = new DataView(dt);   dataGridViewIptalSorgu.DataSource = dv;   }  }
```

Şekil 3.28. (devam)

Eğer ödemenin iptal edilmesi için bir hata kodu almamışsak sistemimiz karşı taraf için bir onay kodu oluşturur ve servisle bunu karşı tarafa bildirir. İptal edilirken bu onay kodu ile doğrulama işlemi yapılmaktadır. İptal sorgu sonucu bir hata almamışsak ekrandaki ilgili alanları doldurup İptal Et butonuna basarak iptal işlemini gerçekleştiriyoruz. İptal işlemini gerçekleştiren kod Şekil 3.29 daki gibidir.

```
else if (cmbMsj.Text.Trim() == "I")
{
 testHarcServis.ogrSvc.OdemeIptalTalepCollection odemeIptalTalep = new
 testHarcServis.ogrSvc.OdemeIptalTalepCollection();
   testHarcServis.ogrSvc.OgrOdemeIptalTalep
     odemeIptal = new testHarcServis.ogrSvc.OgrOdemeIptalTalep()
   {
       OgrenciNo = txtOgrNo.Text.Trim(),
       MesajTipi = cmbMsj.Text.Trim(),
       OdemeTimeStamp = txtOdemeTimeStamp.Text.Trim(),
       OdemeTutari = Convert.ToDecimal(OdemeTutari.Text.Trim()),
       ParaBirimi = "TL",
       TahakkukReferansNo = txtTahkRefNo.Text.Trim(),
       TahsilatReferansNo = txtTahsilatRefNo.Text.Trim(),
       UcretTuru = cmbUcret.Text.Trim(),
       OnayKodu = txtOnayKod.Text.Trim(),
```

Şekil 3.29. İptal İşlemini yapan kod parçası

```
    BankaKodu = Convert.ToInt32(txtBankaKodu.Text.Trim())
 };

 odemeIptalTalep.Add(odemeIptal);
 ogrSvc.Ogrenci ogr = client.OdemeIptal(odemeIptalTalep);
 DataTable dt = new DataTable(); DataRow dr;
 dt.Columns.Add(new DataColumn("OgrenciNo", typeof(string)));
 dt.Columns.Add(new DataColumn("MesajStatusu", typeof(string)));
 dt.Columns.Add(new DataColumn("OdemeTutari", typeof(string)));
 dt.Columns.Add(new DataColumn("ParaBirimi", typeof(string)));
 dt.Columns.Add(new DataColumn("UcretTuru", typeof(string)));
 dt.Columns.Add(new DataColumn("OdemeTimeStamp", typeof(string)));
 dt.Columns.Add(new DataColumn("TahakkukReferansNo", typeof(string)));
 dt.Columns.Add(new DataColumn("TahsilatReferansNo", typeof(string)));
if (ogr.HataKodu != 0)

 MessageBox.Show("Hata Kodu: "+ogr.HataKodu.ToString()+ " - "+ogr.HataKodAciklama);

for (int i = 0; i < ogr.IptalOdemeler.OdemeIptal.Count; i++)
  {
     dr = dt.NewRow();
     dr[0] = ogr.IptalOdemeler.OdemeIptal[i].OgrenciNo;
     dr[1] = ogr.IptalOdemeler.OdemeIptal[i].MesajStatusu;
     dr[2] = ogr.IptalOdemeler.OdemeIptal[i].OdemeTutari;
     dr[3] = ogr.IptalOdemeler.OdemeIptal[i].ParaBirimi;
     dr[4] = ogr.IptalOdemeler.OdemeIptal[i].UcretTuru;
     dr[5] = ogr.IptalOdemeler.OdemeIptal[i].OdemeTimeStamp;
     dr[6] = ogr.IptalOdemeler.OdemeIptal[i].TahakkukReferansNo;
     dr[7] = ogr.IptalOdemeler.OdemeIptal[i].TahsilatReferansNo;
     dt.Rows.Add(dr);
  }
   DataView dv = new DataView(dt);
   dataGridViewOdemeIptal.DataSource = dv;
 }
```

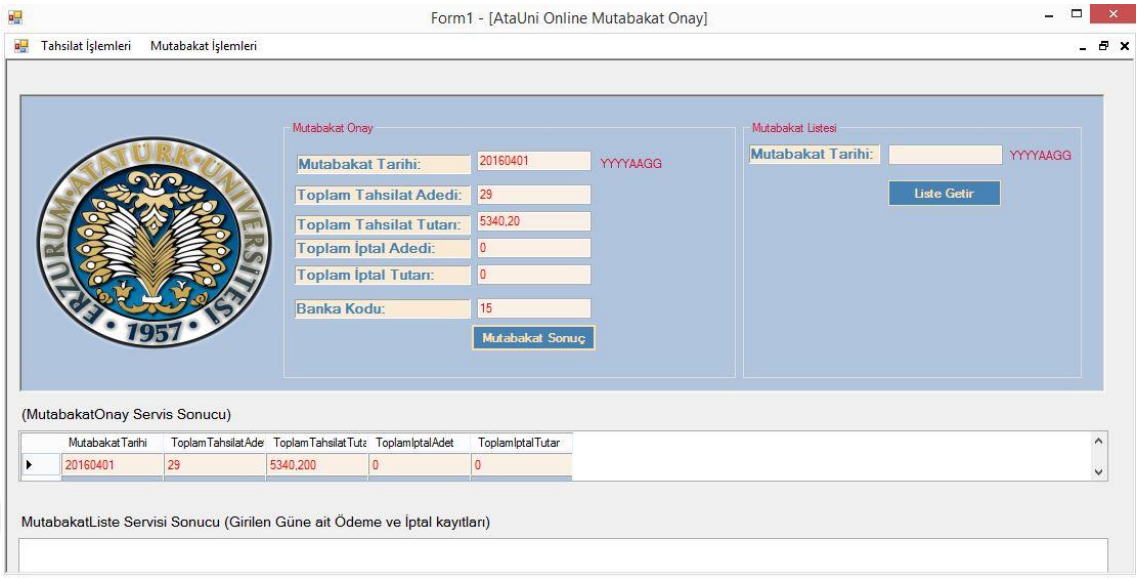
Şekil 3.29. (devam)

Uygulamanın diğer bir kısmı da Mutabakat İşlemleri sekmesidir. Mutabakat işlemleri seçilen herhangi bir tarihte ilgili banka ve kurum arasında toplam alınan ödeme ve iptal edilen ödeme sayısı ve miktarları girilerek sorgulanıyor. Şekil 3.30. da mutabakat onay işlemini gerçekleştiren kod kısmı görülmektedir.

```
private void btnOnaySonuc_Click(object sender, EventArgs e)

   testHarcServis.ogrSvc.MutabakatOnayTalepCollection
         mutabakat = new testHarcServis.ogrSvc.MutabakatOnayTalepCollection();
   testHarcServis.ogrSvc.MutabakatOnayTalep
         mutabakatOnay = new testHarcServis.ogrSvc.MutabakatOnayTalep()
   {
      BankaMutabakatTarihi = txtMutTarOnay.Text.Trim(),
      BankaTopIptalAdet = Convert.ToInt32(txtToplamIptalAdet.Text.Trim()),
      BankaTopIptalTutar = Convert.ToDecimal(txtToplamIptalTutar.Text.Trim()),
      BankaTopTahsilatAdet =  Convert.ToInt32(txtTopTahAdet.Text.Trim()),
      BankaTopTahsilatTutar = Convert.ToDecimal(txtTopTahTutar.Text.Trim()),
      BankaKodu = Convert.ToInt32(txtMutBankaKodu.Text.Trim())
   };
   mutabakat.Add(mutabakatOnay);  ogrSvc.Ogrenci ogr = client.MutabakatOnay(mutabakat);
   DataTable dt = new DataTable();
   DataRow dr;
   dt.Columns.Add(new DataColumn("MutabakatTarihi", typeof(string)));
   dt.Columns.Add(new DataColumn("ToplamTahsilatAdet", typeof(string)));
   dt.Columns.Add(new DataColumn("ToplamTahsilatTutar", typeof(string)));
   dt.Columns.Add(new DataColumn("ToplamIptalAdet", typeof(string)));
   dt.Columns.Add(new DataColumn("ToplamIptalTutar", typeof(string)));
 if (ogr.HataKodu != 0)   MessageBox.Show("Hata Kodu: "+ogr.HataKodu.ToString()+" -
"+ogr.HataKodAciklama);
       for (int i = 0; i < ogr.Mutabakat.MutabakatOnay.Count; i++)
       {
          dr = dt.NewRow();
          dr[0] = ogr.Mutabakat.MutabakatOnay[i].UnivMutabakatTarihi;
          dr[1] = ogr.Mutabakat.MutabakatOnay[i].UnivTopTahsilatAdet;
          dr[2] = ogr.Mutabakat.MutabakatOnay[i].UnivTopTahsilatTutar;
          dr[3] = ogr.Mutabakat.MutabakatOnay[i].UnivTopIptalAdet;
          dr[4] = ogr.Mutabakat.MutabakatOnay[i].UnivTopIptalTutar;
          dt.Rows.Add(dr);
       }
       DataView dv = new DataView(dt);    dataGridOnay.DataSource = dv;
   }
```

Şekil 3.30. Mutabakat sorgusunu yapan kod parçası

Eğer girilen tarih için mutabakat sağlanmışsa bir problem yok denilip o günün hesap işlemleri kapatılır. Şekil 3.31'de görüleceği üzere, girilen bir tarihte bir banka için mutabakat sorgusu yapılmış ve mutabakat sağlandığı için herhangi bir uyarı alınmadı ve liste Mutabakat Onay Servis Sonucu kısmında listelendi.



Şekil 3.31. Mutabakat sorgulama

Mutabakatın sağlanamaması durumunda ekranda 'Bankanın gönderdiği mutabakat tutarı ile kurumdaki kayıtlar uyuşmamaktadır' diye uyarı verilmektedir (Şekil 3.32). Mutabakat tarihi kısmına ilgili tarihi girerek listeyi alan kurum kendi tarafıyla bu listeyi karşılaştırmaktadır. Eğer düşmeyen tahsilat varsa o tahsilat bilgileri o tarih için yeniden kuruma gönderilmektedir.

Şekil 3.32. Mutabakatın sağlanamaması durumu

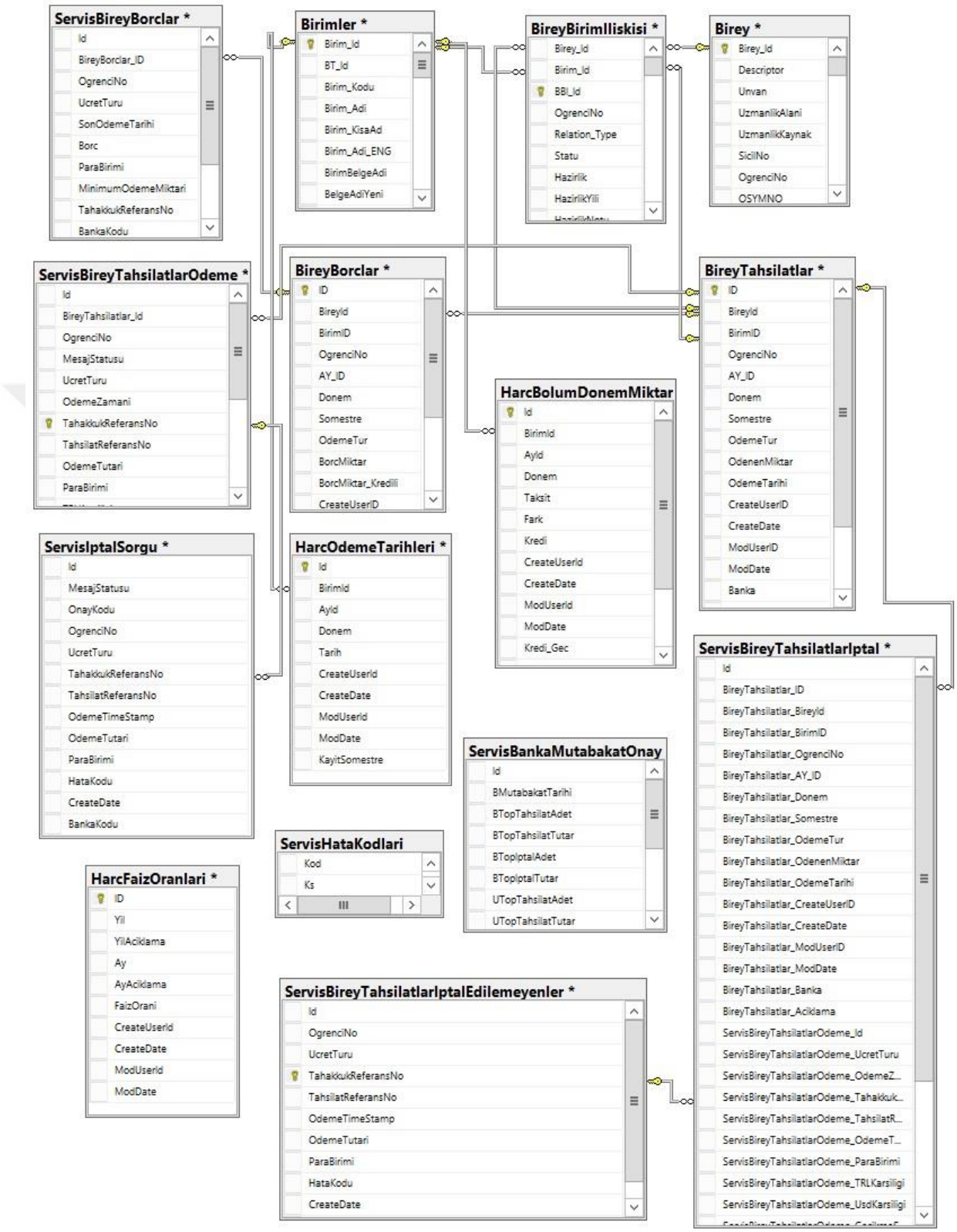
Mutabakat sağlanamadığı durumda, o tarih için liste çekilip kontrol edilmesi gerekmektedir. Şekil 3.33 de listeyi çeken kod parçası görülmektedir.

```
private void btnListe_Click(object sender, EventArgs e)
   {
    testHarcServis.ogrSvc.MutabakatListeTalepCollection
          mutabakatListe = new testHarcServis.ogrSvc.MutabakatListeTalepCollection();
    testHarcServis.ogrSvc.MutabakatListeTalep
          mutabakatListeTalep = new testHarcServis.ogrSvc.MutabakatListeTalep()
    {
       BankaMutabakatTarihi = txtMutTarList.Text.Trim()
    };
    mutabakatListe.Add(mutabakatListeTalep);
    ogrSvc.Ogrenci ogr = client.MutabakatListe(mutabakatListe);
    DataTable dt = new DataTable();
    DataRow dr;
      dt.Columns.Add(new DataColumn("Ad", typeof(string)));
      dt.Columns.Add(new DataColumn("Soyad", typeof(string)));
      dt.Columns.Add(new DataColumn("OgrenciNo", typeof(string)));
      dt.Columns.Add(new DataColumn("TahakkukReferansNo", typeof(string)));
      dt.Columns.Add(new DataColumn("TahsilatReferansNo", typeof(string)));
      dt.Columns.Add(new DataColumn("UcretTuru", typeof(string)));
      dt.Columns.Add(new DataColumn("IslemZamani", typeof(string)));
      dt.Columns.Add(new DataColumn("OdemeTutari", typeof(string)));
      dt.Columns.Add(new DataColumn("ParaBirimi", typeof(string)));
      dt.Columns.Add(new DataColumn("OnayKodu", typeof(string)));
      dt.Columns.Add(new DataColumn("IslemTipi", typeof(string)));
    if (ogr.HataKodu != 0)
      MessageBox.Show("Hata Kodu: "+ogr.HataKodu.ToString()+" -
"+ogr.HataKodAciklama);
    for (int i = 0; i < ogr.MutabakatListeler.MutabakatListe.Count; i++)
    {
      dr = dt.NewRow();
      dr[0] = ogr.MutabakatListeler.MutabakatListe[i].Ad;
      dr[1] = ogr.MutabakatListeler.MutabakatListe[i].Soyad;
      dr[2] = ogr.MutabakatListeler.MutabakatListe[i].OgrenciNo;
      dr[3] = ogr.MutabakatListeler.MutabakatListe[i].TahakkukReferansNo;
      dr[4] = ogr.MutabakatListeler.MutabakatListe[i].TahsilatReferansNo;
      dr[5] = ogr.MutabakatListeler.MutabakatListe[i].UcretTuru;
      dr[6] = ogr.MutabakatListeler.MutabakatListe[i].IslemZamani;
      dr[7] = ogr.MutabakatListeler.MutabakatListe[i].OdemeTutari;
      dr[8] = ogr.MutabakatListeler.MutabakatListe[i].ParaBirimi;
      dr[9] = ogr.MutabakatListeler.MutabakatListe[i].OnayKodu;
      dr[10] = ogr.MutabakatListeler.MutabakatListe[i].IslemTipi;
      dt.Rows.Add(dr);
    }
    DataView dv = new DataView(dt);
    dataGridViewMutabakatListe.DataSource = dv;      }     }
```

Şekil 3.33. Mutabakat listesini çeken kod parçası

3.5. Veri Modeli

VM kısmında tablolar ve SP'ler yer almaktadır. Bu projede asıl işi yapan SP'lerdir. SP'lere asıl yükü yüklememizin sebebi, SP'lerle işlem yapmanın daha performanslı sonuçlar vermesidir. Örnek vermek gerekirse bir faiz hesabı, arayüz kod kısmında değil SP tarafında yaptırıldı. Bunun gibi hesap işlemleri ya da ödemeye yönelik tüm işlemler SP'lere yaptırılmıştır. Projemizde arayüz ve servisler iletişimi sağlayacak birer aracı olarak düşünülmüştür. Bu proje kapsamında veri tabanında birçok tablo tanımlanmıştır. Tabloların her birisine ayrı bir görev tanımlanmıştır. Örnek vermek gerekirse borç sorgularının her birisinin eklendiği tablolarımız, ödemelerin tutulduğu tablolarımız, tüm loğların alındığı tablolarımız gibi birçok tablo VM'de yer almaktadır. Öğrencilerin ödemelerinde bir problemle karşılaşıldığında tüm bu tablolardaki veriler incelenerek problem varsa giderilmesi sağlanmaktadır. Şekil 3.34'de görüleceği üzere bu tablolar birbirleriyle karmaşık ilişki içerisindedir.



Şekil 3.34. Veri tabanında yer alan tablolar ve ilişkileri

VM'de proje kapsamında her 'tablo'ya ve SP'ye önemli bir misyon yüklenmiştir. Şimdi kısaca bu 'tablo'lar ve SP'lerden ve işlevlerinden bahsedelim.

3.5.1. Veri tabanında bulunan tablolar

ServisBireyBorclar: Bu tabloya borç sorgulama log tablosu denilebilir. Herhangi bir öğrenci için banka şubesi, atm, internet bankacılığı ya da kredi kartı ödeme kısmından yapılan tüm borç sorgulamaları bu tabloya insert edilmektedir. Örnek verilecek olursa Windows arayüzünden Erkan Bayram öğrencisi için yaptığımız borç sorguları (Şekil 3.35) bu tabloya insert edilmiştir.

Results | Messages

	Id	BireyBorclar_ID	OgrenciNo	UcretTuru	SonOdemeTarihi	Borc	ParaBirimi	MinimumOdemeMiktari	TahakkukReferansNo	BankaKodu	HataKodu	CreateDate	HesapKodu
1	6655829	3206647	13040201010	EG1	2016-03-15	134.61	TL	129.00	160203206647	15	0	2016-05-27 15:19:19.110	1
2	6655828	3206647	13040201010	EG1	2016-03-15	134.61	TL	129.00	160203206647	15	0	2016-05-27 15:12:37.220	1
3	6655827	3206648	13040201010	EG1	2015-09-30	145.83	TL	129.00	150103206648	15	0	2016-05-27 15:08:20.710	1
4	6655826	3206647	13040201010	EG1	2016-03-15	134.61	TL	129.00	160203206647	15	0	2016-05-27 15:08:20.710	1

Şekil 3.35. Borç sorgularınının tablolara loglanması

Birimler: Öğrencilerin kayıtlı bulundukları birimlerin bilgilerinin tutulduğu tablodur. Bu tabloda (Şekil 3.36) öğrencinin birinci öğretim, ikinci öğretim, açık öğretim gibi öğretim tiplerinin yanı sıra birimin alt üst ilişkisi, bitirilme süresi vb. bilgilerin hepsi bu tabloda tutulmaktadır.

Results | Messages

	birim_Id	Birim_Kodu	Birim_Adi	Ogretim_Dili	Ogretim_Tipi	donem	AzamiDonem	BirimTip	OsymKodu	yoksisKodu	Statu	PuanTuru
1	661	040201	Bilgisayar Mühendisliği - Bilim Dalı - Tezli Yükseklisans Programı	4	1	4	6	1	101400012	237006	1	33

Şekil 3.36. Öğrenci birim bilgileri

BireybirimIliskisi: Öğrencinin hangi birimlerde kaydının bulunduğu, giriş yılı,kayıt tarihi, statüsü, kaçıncı dönemini okuduğu, kaçıncı sınıfta olduğu vb. bilgilerinin tutulduğu tablodur.

Birey: Öğrencinin T.C. kimlik no, doğum tarihi, doğum yeri vb. nufüs bilgilerinin tutulduğu tablodur. Her öğrenci için bu tabloda sadece bir tane veri bulunabilir yani Birey tablosunda aynı kişiye ait iki adet satır bulunamaz.

ServisBireyTahsilatlarOdeme: Ödemesi alınan tahsilat ilk olarak bu tabloya insert edilir. Bu tablo ödeme bilgilerinin tutulduğu ve üniversite personeline hiçbir şekilde müdahale izni verilmeyen her durumda öğrencinin değiştirilmeyen tahsilat bilgilerinin sorgulanabileceği yedek tablodur. Bu tabloya insert edilen veriler hemen sonra bireytahsilatlar tablosuna da eklenmektedir.

BireyBorclar: Öğrenci borç sorgulama sonuncunda hesaplanan borçlar bu tabloya (Şekil 3.37) yazılır. Eğer borç sorgulaması yapılan ödeme yapılmayıp başka bir zamanda yeniden borç sorgulaması yapılırsa, ilgili borç yeni baştan hesaplanmaz ve bu tablodaki daha önce hesaplanmış olan borç gösterilir. Böyle bir durumda her seferinde yeniden borç hesaplaması yapılmadığından borç sorguya servisten daha kısa sürede cevap dönecektir.

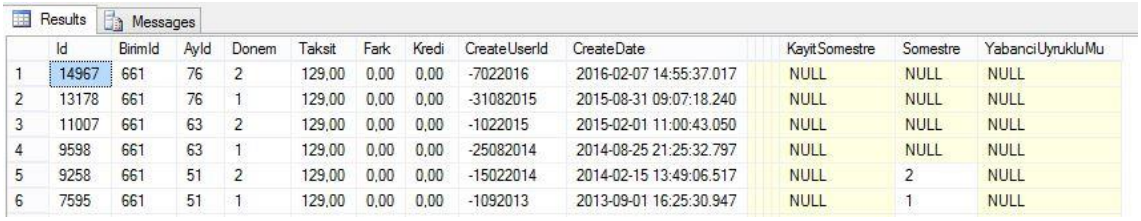
Results | Messages

	ID	BireyId	BirimID	OgrenciNo	AY_ID	Donem	Somestre	OdemeTur	BorcMiktar	CreateUserID	CreateDate	ModUserID	ModDate
1	3206647	386111	661	13040201010	76	2	2	Katkı Payı (2015 - 2016) - Bahar	129,00	1	2016-05-25 16:07:33.327	1	2016-05-25 16:13:56.703
2	3206648	386111	661	13040201010	76	1	1	Katkı Payı (2015 - 2016) - Güz	129,00	219096	2016-05-27 10:58:53.123	NULL	NULL

Şekil 3.37. Borçların kaydedildiği tablo

BireyTahsilatlar: İlk olarak ServisbireyTahsilatlarOdeme tablosuna insert edilen ödemeler, hemen sonra bu tabloya insert edilir. BireyTahsilatlar tablosu üniversitede ödemelerle ilgili olan personelin müdahalesine açıktır. Personel ihtiyaç duyulması durumunda bu tablodaki bazı verileri değiştirebilir.

HarcBolumDonemMiktar: Öğrencinin birimine bağlı olarak hangi akademik yıl ve hangi dönemde ne kadar ücret yatırması gerektiği bilgisini tutan tablodur (Şekil 3.38). Bu tabloda bir birim için aynı akademik yıl ve dönem için birden fazla borç bilgisi tutulabilir. Bunun için tablonun içerisinde yer alan kayıtsomestre alanını kullanıyoruz.

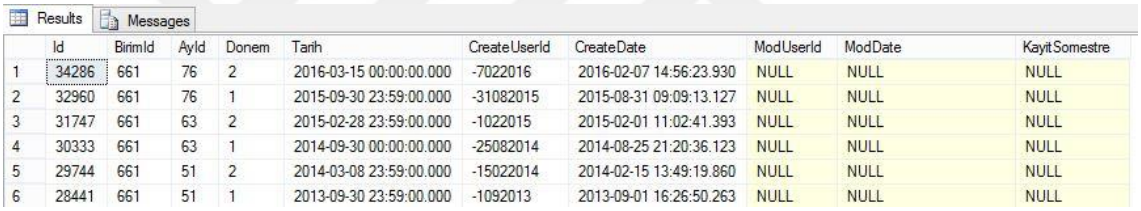


Results | Messages

	Id	BirimId	AyId	Donem	Taksit	Fark	Kredi	CreateUserId	CreateDate	KayitSomestre	Somestre	YabanciUyrukluMu
1	14967	661	76	2	129,00	0,00	0,00	-7022016	2016-02-07 14:55:37.017	NULL	NULL	NULL
2	13178	661	76	1	129,00	0,00	0,00	-31082015	2015-08-31 09:07:18.240	NULL	NULL	NULL
3	11007	661	63	2	129,00	0,00	0,00	-1022015	2015-02-01 11:00:43.050	NULL	NULL	NULL
4	9598	661	63	1	129,00	0,00	0,00	-25082014	2014-08-25 21:25:32.797	NULL	NULL	NULL
5	9258	661	51	2	129,00	0,00	0,00	-15022014	2014-02-15 13:49:06.517	NULL	2	NULL
6	7595	661	51	1	129,00	0,00	0,00	-1092013	2013-09-01 16:25:30.947	NULL	1	NULL

Şekil 3.38. Birime göre ödeme miktarları

HarcOdemeTarihleri: Birime bağlı olarak akademik yıl ve dönem göz önünde bulundurularak borcun son ödeme tarihi bilgisinin tutulduğu tablodur (Şekil 3.39.). Eğer öğrenci bu tablodaki son ödeme tarihinden sonra ödemesini gerçekleştirmişse faiz uygulanacaktır.



Results | Messages

	Id	BirimId	AyId	Donem	Tarih	CreateUserId	CreateDate	ModUserId	ModDate	KayitSomestre
1	34286	661	76	2	2016-03-15 00:00:00.000	-7022016	2016-02-07 14:56:23.930	NULL	NULL	NULL
2	32960	661	76	1	2015-09-30 23:59:00.000	-31082015	2015-08-31 09:09:13.127	NULL	NULL	NULL
3	31747	661	63	2	2015-02-28 23:59:00.000	-1022015	2015-02-01 11:02:41.393	NULL	NULL	NULL
4	30333	661	63	1	2014-09-30 00:00:00.000	-25082014	2014-08-25 21:20:36.123	NULL	NULL	NULL
5	29744	661	51	2	2014-03-08 23:59:00.000	-15022014	2014-02-15 13:49:19.860	NULL	NULL	NULL
6	28441	661	51	1	2013-09-30 23:59:00.000	-1092013	2013-09-01 16:26:50.263	NULL	NULL	NULL

Şekil 3.39. Borçların son ödeme tarihleri

ServisIptalSorgu: Ödemesi iptal edilmek istenen tahsilatların, iptal edilip edilemeyeceği durumu için yapılan her sorguyu bu tabloda logluyoruz. Bu tabloyu hangi ödeme için, ne zaman hangi banka tarafından iptal sorgusu yapıldığı bilgisine ihtiyacımız olduğu durumlarda kullanıyoruz.

ServisBireyTahsilatIptal: Ödemesi iptal edilen tahsilatlar bu tabloya insert edilir ve servisBireyTahsilatOdeme, bireytahsilatlar tablolarından silinir. Silme işleminden sonra bireytahsilatlar_log tablosuna silinen tahsilat bilgileri insert edilir.

ServisBireyTahsilatlarIptalEdilemeyenler: Ödemesi iptal edilmek istenen fakat ders alma yaptığı için ödeme iptal işlemine izin verilmeyen ödemeler bu tabloda loglanır. Ödemesi iptal edilmek istenen fakat edilemeyen tahsilatların listesine ihtiyaç duyduğumuz zaman bu tablodan çekeceğiz.

ServisBankaMutabakatOnay: Bankalar tarafından mutabakatın gerçekleşip gerçekleşmediğinin tespiti için servis üzerinden yapılan her sorgu bu tabloda loglanır. Tabloda mutabakat sağlansın ya da sağlanmasın tüm bilgiler yer almaktadır.

HarcFaizOranları: Bu tablo **(**Şekil 3.40**)** üniversitenin ödemelerle ilgili personeli tarafından bir arayüz aracılığıyla yıl ve aya bağlı olarak girilen ceza oranları bilgilerinin tutulması için kullanılır. Son ödeme tarihinden sonra yatırılan ücretlere eklenecek gecikme cezası miktarının hesaplanmasında bu tablodaki veriler kullanılır.

Results | Messages

	ID	Yil	YilAciklama	Ay	AyAciklama	FaizOrani	CreateUserId	CreateDate	ModUserId	ModDate
1	972	2016	2016 Harç Yılı	12	Aralık	1,45	219096	2015-07-10 13:38:00	NULL	NULL
2	971	2016	2016 Harç Yılı	11	Kasım	1,45	219096	2015-07-10 13:38:00	NULL	NULL
3	970	2016	2016 Harç Yılı	10	Ekim	1,45	219096	2015-07-10 13:38:00	NULL	NULL
4	969	2016	2016 Harç Yılı	9	Eylül	1,45	219096	2015-07-10 13:38:00	NULL	NULL
5	968	2016	2016 Harç Yılı	8	Ağustos	1,45	219096	2015-07-10 13:38:00	NULL	NULL
6	967	2016	2016 Harç Yılı	7	Temmuz	1,45	219096	2015-07-10 13:38:00	NULL	NULL
7	966	2016	2016 Harç Yılı	6	Haziran	1,45	219096	2015-07-10 13:38:00	NULL	NULL
8	965	2016	2016 Harç Yılı	5	Mayıs	1,45	219096	2015-07-10 13:38:00	NULL	NULL
9	964	2016	2016 Harç Yılı	4	Nisan	1,45	219096	2015-07-10 13:38:00	NULL	NULL
10	963	2016	2016 Harç Yılı	3	Mart	1,45	219096	2015-07-10 13:38:00	NULL	NULL
11	962	2016	2016 Harç Yılı	2	Şubat	1,45	219096	2015-07-10 13:38:00	NULL	NULL
12	961	2016	2016 Harç Yılı	1	Ocak	1,45	219096	2015-07-10 13:38:00	NULL	NULL
13	960	2015	2015 Harç Yılı	12	Aralık	1,45	219096	2014-02-05 13:52:00	NULL	NULL
14	959	2015	2015 Harç Yılı	11	Kasım	1,45	219096	2014-02-05 13:52:00	NULL	NULL
15	958	2015	2015 Harç Yılı	10	Ekim	1,45	219096	2014-02-05 13:52:00	NULL	NULL
16	957	2015	2015 Harç Yılı	9	Eylül	1,45	219096	2014-02-05 13:52:00	NULL	NULL
17	956	2015	2015 Harç Yılı	8	Ağustos	1,45	219096	2014-02-05 13:52:00	NULL	NULL
18	955	2015	2015 Harç Yılı	7	Temmuz	1,45	219096	2014-02-05 13:52:00	NULL	NULL
19	954	2015	2015 Harç Yılı	6	Haziran	1,45	219096	2014-02-05 13:52:00	NULL	NULL
20	953	2015	2015 Harç Yılı	5	Mayıs	1,45	219096	2014-02-05 13:52:00	NULL	NULL
21	952	2015	2015 Harç Yılı	4	Nisan	1,45	219096	2014-02-05 13:52:00	NULL	NULL
22	951	2015	2015 Harç Yılı	3	Mart	1,45	219096	2014-02-05 13:52:00	NULL	NULL
23	950	2015	2015 Harç Yılı	2	Şubat	1,45	219096	2014-02-05 13:52:00	NULL	NULL
24	949	2015	2015 Harç Yılı	1	Ocak	1,45	219096	2014-02-05 13:52:00	NULL	NULL

Şekil 3.40. Ceza oranları bilgileri

ServisHataKodları: Servis üzerinden gelen taleplere dönülecek cevaplarda hata kodları açıklama olarak gönderilmez. Daha önceden ilan edilen hangi hata kodunun açıklamasının ne olduğu bilgisi tüm bankalar tarafından bilindiği için hata kodları integer olarak servisi sorgulayan bankalara gönderilir. Onlar da bu kodun ne manaya geldiğini bilirler. Örneğin HataKodu=0 gönderilmişse işlem başarılı hata yok demektir.

Bu hata kodlarının ne olduğu ne manaya geldiği sistemimizde bu tabloda (Şekil 3.41) tutulmaktadır.

Results | Messages

	Kod	Ks	Aciklama
1	911	OT	Öğrenci Tanımsız
2	912	BT	Borç Tanımsız
3	913	TR	Tahakkuk Referans tanımsız
4	915	CT	Çift Tahsilat
5	916	TT	Tahsilat Tanımsız
6	919	CI	Çift odeme İptal isteği
7	926	IO	IptalOnayKodu gönderilmemiş ya da hatalı.
8	940	MB	Bankanın gönderdiği mutabakat tutarı ile kurumdaki kayıtlar uyuşmamaktadır.
9	942	DK	Ders kayıt yaptırmış.
10	999	BH	Banka kodu Hatalı.

Şekil 3.41. Servis hata kodları ve açıklamaları

3.5.2. Veri tabanında bulunan saklı yordamlar (*Stored Procedure*)

usp_HarclarBankaBireyBorcSorgu: Servis borç sorgu için her çağrıldığında bu SP çalışır. '@No varchar(11)' ve '@BankaKodu int' olmak üzere iki parametre alır. Buradaki @No parametresi öğrencinin öğretim tipine göre değişkenlik gösterebilir. Örneğin açıköğretim fakültesi öğrencisi için @No parametresi için T.C. öğrenci no ya da T.C. kimlik no gönderilir. Normal eğitim öğrencileri için sadece öğrenci no parametre olarak gönderilir. Bu SP projemizin en büyük yükünü sırtlanmıştır. Bu SP'de öğretim tiplerine göre borç hesabı, faiz hesabı, hangi ücretin hangi hesaba yatacağı, tahakkuk referans numarası oluşturma gibi ödeme sisteminde olmazsa olmaz ana işlemler yapılmaktadır. Bu ana işlemlerin nasıl yapıldığını kısaca inceleyelim:

Gecikme Cezası Hesabı: Son ödeme tarihi geçmiş olan ödemelere gecikme süresine bağlı olarak ceza miktarı eklenir. Bu miktarı *fnHarclarOdemeCezaMiktari* fonksiyonu aracılığıyla hesaplanır. Fonksiyon *@SonOdemTar varchar(20), @OdemeTar varchar(20), @harcMiktar Money* olmak üzere üç adet parametre alır ve geriye

hesaplanan miktarı döner. Fonksiyon içerisinde gecikme cezası hesaplanırken son ödeme tarihi ile aynı yıl içerisindeki ödemeye getirilen ceza hesabı ve son ödeme tarihi ile farklı yıllardaki ödemelere getirilen ceza miktarının hesabı diye iki kriter göz önünde bulunduruldu. Fonksiyon içerisinde aynı yıl içerisindeki geciken ödemeye getirilen ceza miktarının hesabını yapan kod parçası Şekil 3.42 de görülmektedir.

```
Declare @OdemeGun int, @SonGun int, @SonAy int
Declare @SonYil int, @OdemeAy int, @OdemeYil int

set @SonGun= cast(substring (@SonOdemeTar, 1 ,2 ) as int)
set @OdemeGun = cast(substring (@OdemeTar, 1 ,2 ) as int)
set @SonAy= cast(substring (@SonOdemeTar, 4 ,2 ) as int)
set @SonYil= cast(substring (@SonOdemeTar, 7 ,4 ) as int)
set @OdemeAy= cast(substring (@OdemeTar, 4 ,2  ) as int)
set @OdemeYil= cast(substring (@OdemeTar, 7 ,4 ) as int)

Declare @toplam Money, @Ay1 int, @Yil1 int, @Ay2 int, @Yil2 int

Set @Ay1 = @SonAy
Set @Yil1 = @SonYil
Set @Ay2 = @OdemeAy
Set @Yil2 = @OdemeYil

Declare @tempAy int, @faizOran float, @toplamFaiz float

Set @toplamFaiz = 0

If @Yil1 = @Yil2
  Begin
  While (@Ay2 > @Ay1) or ((@Ay1=@Ay2) and (@SonGun < @OdemeGun))
     Begin
      Select top 1  @faizOran= FaizOrani from HarcFaizOranlari
                    where Yil = @Yil1 And Ay =  @Ay1

          Set @toplamFaiz = @toplamFaiz + (@harcMiktar * @faizOran) /100
          Set @Ay1 = @Ay1 +1
     End
End
```

Şekil 3.42. Son ödeme tarihi yılıyla aynı yılda yapılan ödemeye gecikme cezası hesabı

Son ödeme tarihinden farklı yıllarda ödenen borçlar için ceza hesabında Şekil 3.43 de görülebiliceği gibi ‘while döngüsü’ kullanılarak hesap yapılmıştır.

```
Else If @Yil1 < @Yil2
   Begin
      While @Yil2 >= @Yil1
        Begin
          if @Yil1 < @Yil2
            Begin
              While @Ay1 <=12
                Begin
                     Select top 1  @faizOran= FaizOrani
                     from HarcFaizOranlari where Yil = @Yil1 And Ay =  @Ay1
                     Set @toplamFaiz = @toplamFaiz + (@harcMiktar * @faizOran) /100
                     Set @Ay1 = @Ay1 +1
                End
           if @Ay1>12 Set @Ay1 =1
          End
           Else if @Yil1 = @Yil2
           Begin
            Set @tempAy = 1
           While @Ay2 > = @tempAy
           Begin
              Select top 1  @faizOran= FaizOrani
              from HarcFaizOranlari where Yil = @Yil1 And Ay =  @Ay1
            Set @toplamFaiz = @toplamFaiz + (@harcMiktar * @faizOran) /100
            Set @tempAy = @tempAy +1
           End
          End
          Set @Yil1 = @Yil1 +1
  End
 End
          Set @Yil1 = @Yil1 +1
 End   End
```

Şekil 3.43. Son ödeme tarihinden farklı yıllardaki ödemelere gecikme cezası hesabı

Ceza miktarı hesaplandıktan sonra son olarak harç miktarına ceza miktarı da eklenerek fonksiyondan geriye ilgili borç için ödenmesi gereken toplam miktar dönüş yapılır(Şekil 3.44)

```
Declare @ToplamMiktar money
Set @ToplamMiktar = Round(@harcMiktar + @toplamFaiz, 2)
  RETURN(isnull(@ToplamMiktar,0))
END
```

Şekil 3.44. Hesaplanan toplam miktarın return edilmesi

Hangi Ücretin Hangi Hesap Numarasına Yatırılacağının Bankalara Bildirilmesi: *usp_HarclarBankaBireyBorcSorgu* SP'si içerisinde yer alan ücret türü değişkenine bağlı olarak bankalara hangi ücretin hangi hesaba yatması gerektiği bilgisi bildirilir. Bankalarla online ödemeler için entegrasyon sağlanmadan önce ücret türlerine bağlı olarak kurumun hesap numaraları tanımlanır. Yani bankalar hangi ücret türünün hangi hesapla eşleştiği bilgisini ödeme almadan önce bilmek zorundadır. Servise gelen sorgularda hangi ödemenin hangi hesaba yatacağı bilgisi, ücret türü alanını servis cevabında bankalara göndererek belirliyoruz. SP içerisinde bunu belirleyen kod parçacığı Şekil 3.45 de belirtildiği gibidir.

```
If (@OgretimTipi =1) – 1.Öğretim Hesapları
        Begin
              Set @UcretTuru= 'EG1'
              Update #Sonuc Set UcretTuru = @UcretTuru
        End
If (@OgretimTipi =2) – 2.Öğretim Hesapları
        Begin
             Set @UcretTuru= 'EG2'
             Update #Sonuc Set UcretTuru = @UcretTuru
        End
If (@OgretimTipi =3) -- Uzaktan Egitim Hesapları
        Begin
        Update  #Sonuc
        Set UcretTuru= case
                          when Donem in(1,2) then 'UE1'
                          when Donem in(26,27,28,29,41) then 'UES'
                          else 'UE2'
                         end
        End
If (@OgretimTipi =5) -- AOF Hesapları
        Begin
        Update  #Sonuc
        Set UcretTuru= case Donem
                          when 1 then 'AO1'
                          when 2 then 'AO1'
                          else 'AO2'
                         end
        End
If (@Donem =3 ) -- Yaz Okulu
                Begin
                 Update  #Sonuc Set UcretTuru='YAZ' Where Donem =3
                End
```

Şekil 3.45. Hangi ödemenin hangi hesaba yatacağının belirlenmesi

Servise Gelen Tüm Sorguların Loglanması: Bankalar, kredi kartı ya da Windows uygulamalar üzerinden ödeme servise yapılan her sorgu loglanır. Loglama işlemi (Şekil 3.46) borç hesabı bittikten sonra borçları listelemeden hemen önce ServisBireyBorclar tablosuna yapılır.

```
insert into  ServisBireyBorclar
            [BireyBorclar_ID]
           ,[OgrenciNo]
           ,[UcretTuru]
           ,[SonOdemeTarihi]
           ,[Borc]
           ,[ParaBirimi]
           ,[MinimumOdemeMiktari]
           ,[TahakkukReferansNo]
           ,[BankaKodu]
           ,[HataKodu]
           ,[CreateDate]
           ,[HesapKodu]
           )
 Select
            BireyBorcId
           ,@No
           ,UcretTuru
           ,BorcHarcTarih
           ,TahsilatHarcMiktar
           ,'TL'
           ,BorcHarcMiktar
           ,'TahakkukReferansNo' =  substring( convert (varchar (4),year (BorcHarcTarih)),3,2 )+
                      dbo.[FnKarakterEkle] ( convert (varchar (2),Donem ),2,'0',1)
                       + dbo.[FnKarakterEkle] (convert (varchar (8),BireyBorcId ),8,'0',1)
           ,@BankaKodu
           ,@HataKodu_
           ,GetDate()
           ,case Donem
                      when 1 then 1
                      when 2 then 1
                      else 2
           End
from #Sonuc where BireyTahsilatId =0 And @BankaKodu in (10,15,25)
```

Şekil 3.46. Tüm borç sorgularının loglanması

Borç Hesabı: Öğrenciye ait borçların hesabını yapabilmek için her öğretim tipine göre ayrı bir SP 'usp_HarclarBankaBireyBorcSorgu' nun içerisinde çağrılmaktadır. Bunları inceleyecek olursak:

usp_HarclarBorclandirmaUzakOgrenci: Uzaktan eğitim öğrencilerinin borçlarını hesaplar. @BireyId int, @BirimId int, @BankaKodu int olmak üzere üç tane parametre alır. Bu kısımda #tempHarcDonemleri adlı bir temp tablo (Şekil 3.47) oluşturulur. Öğrencinin ödeme yapması gereken akademik yıl ve dönem bilgileri bu temp tablosuna atılır.

```
Create Table #tempBireyBorclar
        (
          BireyId int,
          BirimId int,
          Statu int,
          OgrenciNo varchar(20),
          Ad varchar(100),
          Soyad varchar(100),
          BirimAdi varchar(200),
          OgrTip int,
          BrmDonem int,
          OkuduguDonem int,
          Miktar money
         )
Insert Into #tempHarcDonemleri
          Select ht.AyId, ht.Donem,ht.KayitSomestre
                    From HarcOdemeTarihleri HT
                     inner join HarcBolumDonemMiktar HM
                              on HT.BirimId= HM.BirimId
                                  And HT.AyId = HM.AyId
                                  And HT.Donem= HM.Donem
                                  And HT.KayitSomestre= HM.KayitSomestre
                     inner join BireyBirimIliskisi bbi
                              on HT.BirimId =bbi.Birim_Id
                    Where   bbi.Birey_Id =@BireyId
                              and bbi.Birim_Id =@BirimId
                              and ((ht.AyId = bbi.KayitAY_Id
                              and ht.KayitSomestre=bbi.KayitSomestre)
                              or  (ht.AyId > bbi.KayitAY_Id
                              and ht.KayitSomestre=bbi.KayitSomestre))
                              and bbi.Statu in (75,11,50,7,80)
```

Şekil 3.47. Uzaktan eğitim öğrencisinin borçlu olduğu dönemler

Daha sonra Şekil 3.48 de görüleceği gibi sonra bu temp tablosuna bağlı olarak ilgili tablolarla join işlemleri yapılarak bireyborclar tablosuna öğrencinin hesaplanan borçları atılır.

```
INSERT INTO [BireyBorclar]
        ([BireyId]
        ,[BirimID]
        ,[OgrenciNo]
        ,[AY_ID]
        ,[Donem]
        ,[Somestre]
        ,[OdemeTur]
        ,[BorcMiktar]
        ,[BorcMiktar_Kredili]
        ,[CreateUserID]
        ,[CreateDate]
        ,[ModUserID]
        ,[ModDate]
        )
 Select
          @BireyId, @BirimId, bbi.OgrenciNo, t.AyId, t.Donem , null
          ,'Katkı Payı (' + substring(Descriptor,1,11) +')' +  Deger
          ,HM.Taksit,  NULL, @BankaKodu, GETDATE(), NULL, NULL
 from #tempHarcDonemleri t
          inner join AkademikYil  a
                    on a.AY_Id = t.AyId
          inner join Sabitler s
                    on s.KullanildigiYer like 'HarcDonem'
                              and s.Sabit = t.donem
          inner join BireyBirimIliskisi bbi
                    on bbi.Birey_Id=@BireyId and bbi.Birim_Id =@BirimId
                              and Statu in (75,11,50,7,80)
          inner join HarcBolumDonemMiktar HM
                    on  HM.BirimId=@BirimId and HM.AyId=t.AyId
                              and HM.Donem = t.donem and hm.KayitSomestre =t.Kayitsomestre
          left join BireyBorclar b
                    on b.AY_ID =t.AyId and b.Donem = t.Donem
                              and b.BireyId =@BireyId and b.BirimID=@BirimId
where b.ID is null
```

Şekil 3.48. Uzaktan eğitim öğrencisinin borçlarının bireyborclar tablosuna eklenmesi

exec usp_HarclarBorclandirmaAOFOgrenci: Açıköğretim öğrencilerinin borçlarını hesaplayan SP'dir. @BireyId int, @BirimId int, @BankaKodu int adlarında üç tane

parametre alır. Bu SP'nin içerisinde #tempBireyBorclar ve #tempHarcDonemleri adlı iki tane temp tablo oluşturulur(Şekil 3.49).

```
Create Table #tempHarcDonemleri
   (        Id int identity(1,1),
            AyId int,
            Donem int,
            Kayitsomestre int     )
Create Table #tempBireyBorclar
  (         [BireyId] [int] NULL,
            [BirimID] [int] NULL,
            [OgrenciNo] [varchar](50) NULL,
            [AY_ID] [int] NULL,
            [Donem] [int] NULL,
            [Somestre] [int] NULL,
            [OdemeTur] [varchar](50) NULL,
            [BorcMiktar] [money] NULL,
            [BorcMiktar_Kredili] [money] NULL,
            [CreateUserID] [int] NULL,
            [CreateDate] [datetime] NULL,
            [ModUserID] [int] NULL,
            [ModDate] [datetime] NULL    )
```

Şekil 3.49. Açıköğretim öğrencilerinin ödemesi gereken dönem ve borç bilgileri

Şekil 3.50 de görüleceği gibi oluşturulan bu iki tabloya öğrencinin hangi dönemler için ne kadar ücret yatırması gerektiği bilgisini ekleme işlemi yapılır.

```
Insert Into #tempHarcDonemleri
  Select ht.AyId, ht.Donem,ht.KayitSomestre
  from HarcOdemeTarihleri ht, BireyBirimIliskisi bbi
  where ht.BirimId = bbi.Birim_Id
            and  bbi.Birey_Id =@BireyId
            and bbi.Birim_Id =@BirimId
            and ((ht.AyId = bbi.KayitAY_Id and ht. KayitSomestre =bbi. KayitSomestre)
            or  (ht.AyId > bbi.KayitAY_Id and ht. KayitSomestre =bbi. KayitSomestre))
                     and  Statu in(50,1000,80)
INSERT INTO  #tempBireyBorclar
 select @BireyId,@BirimId, bbi.OgrenciNo,t.AyId ,t.Donem  ,null,
            'Katkı Payı (' + substring(Descriptor,1,11) +')-'+ Deger
            ,HM.Taksit,NULL,@BankaKodu,GETDATE(),NULL,NULL
```

Şekil 3.50. Açık öğretim temp tablolarının doldurulması

```
from #tempHarcDonemleri t
   inner join AkademikYil a  on a.AY_Id = t.AyId
   inner join Sabitler s  on s.KullanildigiYer like 'HarcDonem' and s.Sabit = t.donem
   inner join BireyBirimIliskisi bbi on bbi.Birey_Id=@BireyId and bbi.Birim_Id =@BirimId
                and Statu in (9,11,50,1000,80)
   inner join HarcBolumDonemMiktar HM on  HM.BirimId=@BirimId and HM.AyId=t.AyId
                and HM.Donem = t.donem and hm. KayitSomestre =t. KayitSomestre
   left join BireyBorclar b on b.AY_ID =t.AyId and b.Donem = t.Donem
                and b.BireyId =@BireyId and b.BirimID=@BirimId
where b.ID is null
```

Şekil 3.50. (devam)

Daha sonra ihtiyaç durulması halinde #tempBireyBorclar tablosu farklı statüdeki öğrencilerin durumlarına göre güncellenebilir. Örneğin uzatan öğrencilerin borcu normalin iki katı gibi güncellemeler yapılabilir. Son olarak da Şekil 3.51 deki gibi bireyborclar tablosuna öğrencinin borçları insert edilir.

```
INSERT INTO [BireyBorclar]
        ([BireyId]
        ,[BirimID]
        ,[OgrenciNo]
        ,[AY_ID]
        ,[Donem]
        ,[Somestre]
        ,[OdemeTur]
        ,[BorcMiktar]
        ,[BorcMiktar_Kredili]
        ,[CreateUserID]
        ,[CreateDate]
        ,[ModUserID]
        ,[ModDate]      )
Select  [BireyId]
        ,[BirimID]
        ,[OgrenciNo]
        ,[AY_ID]
        ,[Donem]
        ,[Somestre]
        ,[OdemeTur]
        ,[BorcMiktar]
        ,[BorcMiktar_Kredili]
        ,[CreateUserID]
        ,[CreateDate]
        ,[ModUserID]
        ,[ModDate] from #tempBireyBorclar  where #tempBireyBorclar.BorcMiktar <>0
```

Şekil 3.51. Açıköğretim öğrencilerinin borçlarının bireyborclar'a yazılması

usp_HarclarBorclandirmaLisansOgrenciGuzDonemV2: Lisans- Önlisans öğrencisi olup, öğretim tipi de birinci ve ikinci öğretim olan öğrencilerin güz dönemlerindeki borç hesabında bu SP kullanılır. @BireyId int, @BirimId int, @AyId int, @Donem int, @BankaKodu int olmak üzere beş tane parametre alır. Burada da ilk olarak #tempBireyBorclar adında bir temp tablo oluşturulur. Öğrencinin gore yeni kayıt öğrenci mi ve eski öğrenci mi diye durumuna bakılarak iki farklı durum için #tempBireyBorclar tablosuna farklı insertler yapılır. Öğrenci yeni kayıt öğrencisi ise Şekil 3.52 deki kod kısmı çalışacaktır. Görüleceği üzere öğrencinin statusüne, giriş türüne bağlı olarak temp tablosu güncellenecektir.

```
if exists (select 1 from BireyBirimIliskisi(nolock)
                        where Birey_Id =@BireyId and Birim_Id =@BirimId
                            and Statu in(11,13,50) and Ay_Id =@AyId )
Begin
  insert into #tempBireyBorclar
    Select BBI.Birey_Id, BBI.Birim_Id, BBI.Statu,BBI.OgrenciNo,BRY.Descriptor,
           Brm.Ogretim_Tipi, Brm.Donem, BBI.Okudugu_Donem ,0, BBI.GelmeSekli
    from BireyBirimIliskisi BBI(nolock)
           inner join Birey BRY (nolock)
                 on BRY.Birey_Id = BBI.Birey_Id
           inner join Birimler Brm (nolock)
                 on Brm.Birim_Id = BBI.Birim_Id
   Where BBI.Birey_Id = @BireyId  And BBI.Birim_Id = @BirimId
           And BBI.Ay_Id = @AyId And BBI.Statu in (11,13,50)
           And BBI.GelmeSekli not in(7,9,10) And Brm.BT_Id in (21,33)
           and Brm.Ogretim_Tipi in(1,2)

Update #tempBireyBorclar Set HarcMiktar = isnull(HRC.Taksit,0)
From #tempBireyBorclar T
          inner join HarcBolumDonemMiktar HRC(nolock) ON HRC.BirimId = T.BirimId
Where   HRC.Donem = @Donem  And HRC.AyId = @AyId

Update #tempBireyBorclar Set HarcMiktar = 0
From #tempBireyBorclar T inner  join HarcBolumDonemMiktar HRC(nolock)
          ON HRC.BirimId = T.BirimId
Where   HRC.Donem = @Donem  And HRC.AyId = @AyId
          and T.OgrTip =1 and t.GelmeSekli not in (6,42,73)


Update #tempBireyBorclar –Yabancı Uyruklu 1.Öğretim Öğrencileri
          Set HarcMiktar = 4*isnull(HRC.Taksit,0)
From #tempBireyBorclar T inner join HarcBolumDonemMiktar HRC(nolock)
          ON HRC.BirimId = T.BirimId
```

Şekil 3.52. Yeni kayıt lisans-önlisans öğrenci güz dönemi borçlandırılması

```
Where   HRC.Donem = @Donem  And
        HRC.AyId = @AyId And T.GelmeSekli in (6,42,73)
        And T.OgrTip =1 and T.BirimId !=1366

Update #tempBireyBorclar  --Yabancı Uyruklu 2.Öğretim öğrencileri
        Set HarcMiktar = 2*isnull(HRC.Taksit,0)
From #tempBireyBorclar T inner join HarcBolumDonemMiktar HRC (nolock)
        ON HRC.BirimId = T.BirimId
Where   HRC.Donem = @Donem  And HRC.AyId = @AyId
        And T.GelmeSekli in(6,42,73) And T.OgrTip =2
End
```

Şekil 3.52. (devam)

Öğrenci yeni kayıt olan bir öğrencisi değilse yüzde ona girip girmediği, uzatıp uzatmadığı gibi bilgilerine bağlı olarak temp tablosu (Şekil 3.53) güncellenmektedir.

```
Begin
select @kacDonemHazirlikOkudu =isnull(KacDonemHazOkudu,0)  from BireyBirimIliskisi
   where Birey_Id =@BireyId and Birim_Id =@BirimId
insert into #tempBireyBorclar
  Select  BBI.Birey_Id, BBI.Birim_Id, BBI.Statu, BBI.OgrenciNo,BRY.Descriptor,
          Brm.Ogretim_Tipi, Brm.Donem, (BBI.Okudugu_Donem) ,0, BBI.GelmeSekli
   from BireyBirimIliskisi BBI(nolock)
        inner join Birey BRY(nolock) on BRY.Birey_Id = BBI.Birey_Id
        inner join Birimler Brm (nolock)on Brm.Birim_Id = BBI.Birim_Id
  Where  BBI.Birey_Id = @BireyId  And BBI.Birim_Id = @BirimId
         AND BBI.Statu in (13,14,50) And Brm.BT_Id in (21,33)
         And BBI.GelmeSekli not in (13,15)
         And BBI.GelmeSekli not in(7,9,10
         And (BBI.OgrenciNo<>" or BBI.OgrenciNo is not null)
insert into #tempBireyBorclar  -- CAP ve YanDal Ogrencileri gozetmek icin
   Select BBI.Birey_Id , BBI.Birim_Id, BBI.Statu, BBI.OgrenciNo, BRY.Descriptor,
          Brm.Ogretim_Tipi, Brm.Donem, (BBI.Okudugu_Donem) ,0, BBI.GelmeSekli
   from BireyBirimIliskisi BBI(nolock)
      inner join Birey BRY(nolock) on BRY.Birey_Id = BBI.Birey_Id
      inner join Birimler Brm (nolock)on Brm.Birim_Id = BBI.Birim_Id

   Where  BBI.Birey_Id = @BireyId  And BBI.Birim_Id = @BirimId annd BBI.Statu in
(13,14,50)  And Brm.BT_Id =33  And BBI.GelmeSekli in (13,15) And BBI.GelmeSekli not
in(7,9,10  And (BBI.OgrenciNo<>" or BBI.OgrenciNo is not null)
```

Şekil 3.53. Önlisans-lisans ara sınıf öğrencilerinin güz dönemi borçlandırılması

```
Update #tempBireyBorclar Set HarcMiktar = isnull(HRC.Taksit,0)
From #tempBireyBorclar T inner join HarcBolumDonemMiktar HRC(nolock)
  ON HRC.BirimId = T.BirimId Where HRC.Donem = @Donem  And HRC.AyId = @AyId
Update #tempBireyBorclar –Yabancı Uyruklu 1.Öğretim Öğrencileri
        Set HarcMiktar = 4*isnull(HRC.Taksit,0)
From #tempBireyBorclar T   inner join HarcBolumDonemMiktar HRC(nolock)
        ON HRC.BirimId = T.BirimId
Where HRC.Donem = @Donem  And HRC.AyId = @AyId
        And T.GelmeSekli in (6,42,73) And T.OgrTip =1 and T.BirimId !=1366
Update #tempBireyBorclar
        Set HarcMiktar = 2*isnull(HRC.Taksit,0) –Yabancı Uyruklu 2.Öğretim Öğrencileri
From #tempBireyBorclar T  inner join HarcBolumDonemMiktar HRC (nolock)
        ON HRC.BirimId = T.BirimId
Where   HRC.Donem = @Donem  And HRC.AyId = @AyId
        And T.GelmeSekli in(6,42,73) And T.OgrTip =2
Update #tempBireyBorclar Set HarcMiktar =HarcMiktar * 0
From #tempBireyBorclar BRY inner join HarcBolumDonemMiktar HRC(nolock)
        ON HRC.BirimId = BRY.BirimId
Where   BRY.OgrTip =1 And HRC.Donem =@Donem  And HRC.AyId = @AyId And
         (BRY.OkuduguDonem - @kacDonemHazirlikOkudu)<=BRY.BrmDonem
        and bry.GelmeSekli not in (6,42,73)
Update #tempBireyBorclar --Yuzde Ona Girenler kendi bölümünün gündüz parasını öder.
        Set HarcMiktar = dbo.fnIOBiriminOOHarcMiktari(T.BirimId,@AyId,1)
From #tempBireyBorclar T inner join YuzdeOnaGirenOgrenciler  Y(nolock)
        On Y.BireyID  = T.BireyId AND Y.BirimId = T.BirimId
```

Şekil 3.53. (devam)

Son olarak Şekil 3.54 deki gibi bireyborclar tablosuna #tempbireyborclar tablosundaki verilere bağlı olarak öğrencinin borçları insert edilir.

```
INSERT INTO BireyBorclar
      ([BireyId], [BirimID], [OgrenciNo] ,[AY_ID], [Donem], [Somestre]
       ,[OdemeTur], [BorcMiktar] ,[BorcMiktar_Kredili], [CreateUserID]
      ,[CreateDate], [ModUserID], [ModDate]     )
Select  T.BireyId, T.BirimId, T.OgrenciNo, @AyId, @Donem, @Donem,
          'Katkı Payı (' + substring(Descriptor,1,11) +')  -  ' + Deger,
          T.HarcMiktar, NULL, @BankaKodu, GETDATE(), NULL, NULL
from #tempBireyBorclar T
    inner join AkademikYil (nolock)        on AY_Id = @AyId
    inner join Sabitler (nolock) on KullanildigiYer like 'HarcDonem' and Sabit = @Donem
    left join BireyBorclar bb (nolock)
          on bb.BireyId =T.BireyId and bb.BirimID =T.BirimId
              and bb.AY_ID =@AyId and bb.Donem =@Donem
where T.HarcMiktar!=0  And bb.ID is null
```

Şekil 3.54. Önlisans-lisans öğrenci borçlarının bireyborclar tablosuna eklenmesi

usp_HarclarBorclandirmaLisansOgrenciBaharDonemV2: Önlisans lisans öğrencisi olup birinci ve ikinci öğretim öğrencilerinin bahar dönemi borçlarının hesaplanmasında kullanılır. Mantık ve işleyiş olarak güz dönemi borcunu hesaplayan SP ile aynıdır fakat istisnai birkaç durum sebebiyle bahar dönemi için bu SP'yi oluşturduk.

usp_HarclarBorclandirmaEnstituOgrenciV2: Lisansüstü öğrencilerinin borçlarının hesaplanmasında kullanılır. @BireyId int, @BirimId int, @AyId int, @Donem int, @BankaKodu int parametrelerini alır. Burada da geçici temp tablo kullanılır. Belirli kurallara göre temp tablo doldurulur, güncellenir. Son olarak temp tablodaki verilere göre bireyborclar tablosuna öğrencinin borçları aktarılır.

Usp_HarclarBorclandirmaYazOkulu: Bu SP yaz okuluna kalan ve derslere ön kayıt yaptıran öğrencilerin seçtikleri derslere göre yatırması gereken ücretleri hesaplar. *@BireyId int, @BirimId int, @AyId int, @Donem int, @BankaKodu* parametrelerini alır. Şekil 3.55. de görülebileceği gibi öğrencilerin ön kayıt yaptırdıkları derslere bağlı olarak hesap yapılmaktadır. Buradaki hesapta her fakültenin YÖK tarafından belirlenmiş olan bir hesaplama katsayısı kullanılır.

```
INSERT INTO [BireyBorclar]
        ([BireyId] ,[BirimID] ,[OgrenciNo] ,[AY_ID] ,[Donem]
        ,[Somestre] ,[OdemeTur] ,[BorcMiktar]
        ,[CreateUserID] ,[CreateDate] ,[ApId])
Select  BBI.Birey_Id,BBI.Birim_Id,BBI.OgrenciNo
          ,@AyId,@Donem,@Donem
          ,case  when LEN(AP.Ders_Kodu)>20 then substring(AP.Ders_Kodu,0,20) +'.'
                 else  isnull(AP.Ders_Kodu,'')
                 end  +' / '+
          case  when len(AP.DersAdi)>22 then SUBSTRING(AP.DersAdi,0,22)+'.'
                 else isnull(AP.DersAdi,'')
                 end
          ,dbo.ufn_DersUcretiHesapla(AP.ID, @BireyId, @BirimId)
          ,@BankaKodu, GETDATE(), AP.ID
From      DersAlma DA inner join BireyBirimIliskisi BBI  on BBI.Birey_Id = DA.Ogrenci_Id
                 and BBI.Birim_Id = DA.Birim_Id
          inner join AkademikProgram AP  on DA.AP_Id = AP.ID
          left join BireyBorclar BB  on BB.ApId= DA.AP_Id And BB.BireyId =DA.Ogrenci_Id
                  And BB.BirimID = DA.Birim_Id
Where   DA.Ogrenci_Id = @BireyId and DA.Birim_Id in(@BirimId, @ikinciBirimId)
          and DA.AY_Id = @AyId and DA.Somestre = @Donem and DA.Durumu = 1 and
BB.ID is null
```

Şekil 3.55. Yaz okulu için seçilen derslerin ücretlerinin bireyborclara eklenmesi

Usp_HarclarBorclandirmaYazOkulu SP'sinin içerisinde yer alan *dbo.ufn_DersUcretiHesapla* fonksiyonu (Şekil 3.56) dersin ücretini hesaplama işlevini yerine getirir. Fonksiyon *@ApId int, @OgrId int, @OgrBirimId int* adlı üç adet parametre alır. Fonksiyon içerisinde gerekli hesaplamalar yapıldıktan sonra hesaplanan miktar return edilir.

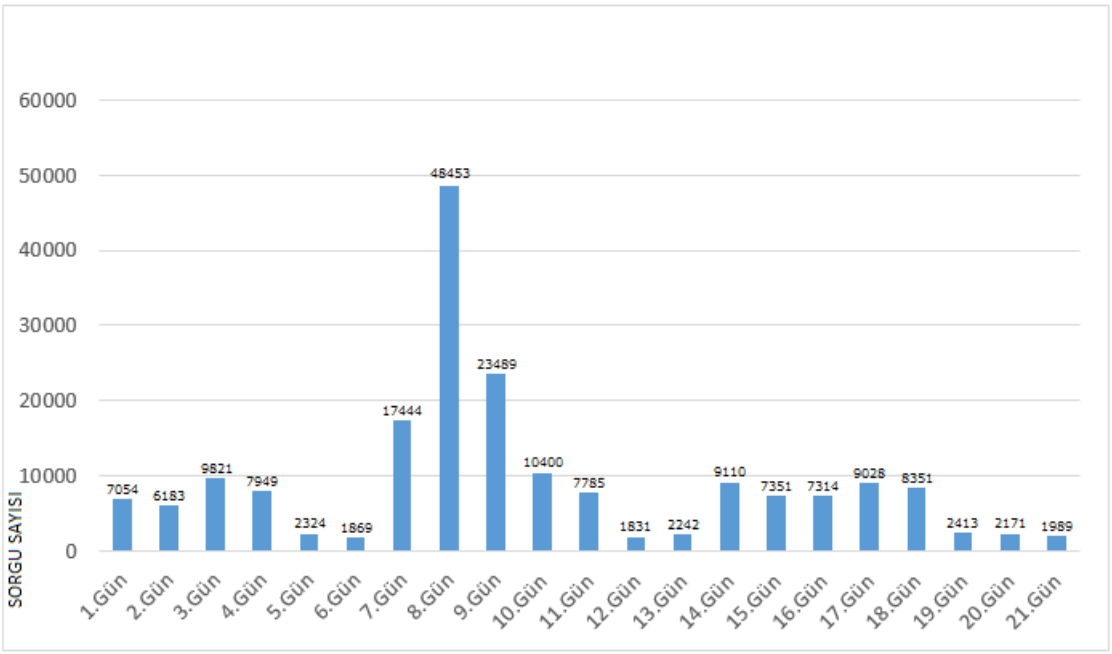
```
Declare
        @Tutar float
        ,@DersSaat float
        ,@DersSaatUcreti float
        ,@OgrTip int
        ,@DersBirimId int
        ,@OgrKatsayi int
        ,@DersKredisi float
Select
         @DersSaat = DersTeoriKredi + DersPratikKredi + DersLabKredi
        ,@DersBirimId = Birim_ID
        ,@DersKredisi=DersKredi
From
        AkademikProgram (nolock)
Where
        ID = @APId

if @DersSaat < @DersKredisi
        begin
                set @DersSaat =@DersKredisi
        end

Select  @DersSaatUcreti = H.Taksit*14
From    HarcBolumDonemMiktar H (nolock)
        inner join Birim_Iliskileri BI1 (nolock)
                on BI1.AB1_Id = H.BirimId
                    and BI1.Relation_Type = 0
        inner join Birim_Iliskileri BI2 (nolock)
                on BI2.AB1_Id = BI1.AB2_Id
                    and BI2.Relation_Type = 0
                    and BI2.AB2_Id = @DersBirimId
Where   BBI.Birey_Id = @OgrId
        and BBI.Birim_Id = @OgrBirimId

Set @Tutar = isnull((@DersSaat * @DersSaatUcreti * @OgrKatsayi),-1)
RETURN @Tutar
```

Şekil 3.56. Yaz okulu ders ücretinin hesaplanması

4. ARAŞTIRMA BULGULARI ve TARTIŞMA

4.1. WCF Servis İstatistikleri

Bu projede Microsoft tarafından SOA yaklaşımına bir çözüm olarak ortaya çıkarılan WCF teknolojisi kullanılarak ödeme servisi oluşturuldu. Projenin Uygulama arayüzü, Veri tabanı kısmı ve WCF servisi olmak üzere üç temel bileşeni vardır. Bu temel bileşenlerden SOA yaklaşımı için şüphesiz en önemli bileşen WCF servisidir. SOA yaklaşımı için kullandığımız WCF servisinin belirli zaman aralıklarında bankalardan gelen borç sorgu isteklerine verdiği cevap sayısını (Şekil 4.1) incelediğimizde servise belirli dönemlerde yoğun bir şekilde sorgu geldiği görülmektedir.

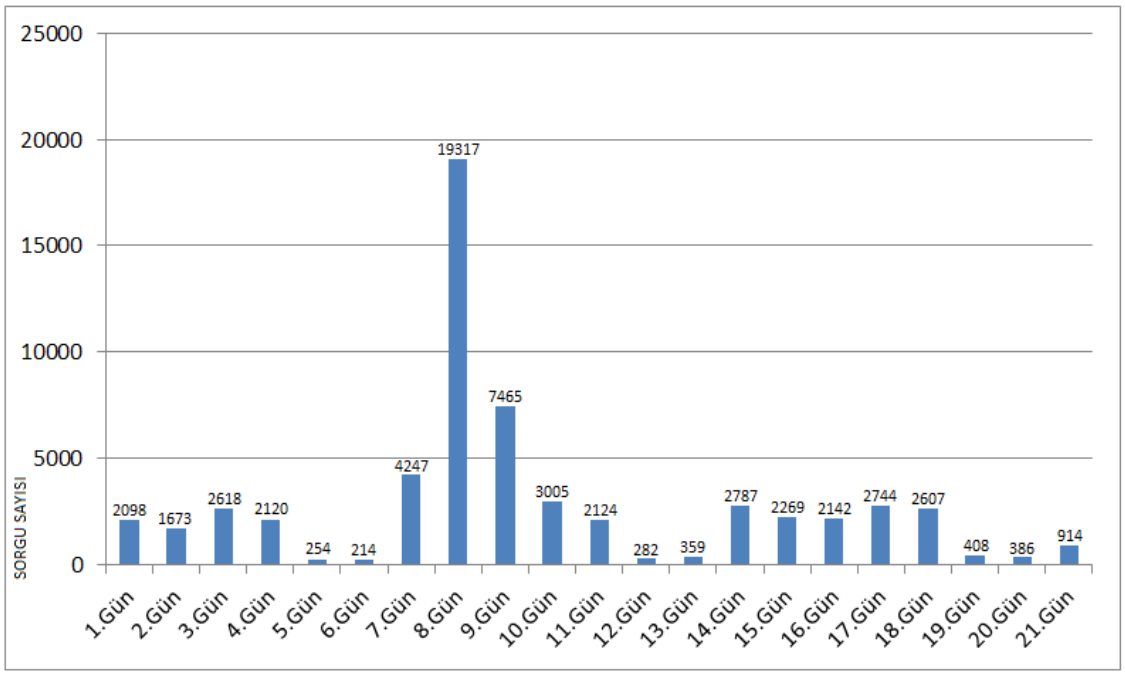


Şekil 4.1. Üç haftalık süreçte borç sorguya dönülen cevapların istatistiği

2015 yılı Eylül ayının ilk üç haftalık sürecinde borç sorgu için alınan istatistik verilerini incelediğimizde, servisin en çok yedinci, sekizinci, dokuzuncu günlerde çağrıldığını ve gelen isteklere en çok cevap bu günlerde dönüldüğü görülmüştür. Bu aralıklarda

servisin yoğun olarak çağrılmasının sebebi öğrencilerin ders alma döneminin bu tarihlerde başlamış olmasıdır. Ders alma yapabilmek için ödeme yapması gereken öğrencilerin, borç sorgu yapmaları bu tarihlerde servisin yoğunluğunun artmasına sebep olmuştur. Öğrenciler ödeme yapmasalar bile ATM'ler, internet bankacılığı ve banka şubeleri aracılığıyla her borç sorgusu yaptırdığında servisimizin ilgili metotları çağrılır ve o metotlara da cevap dönmesi gereken metotlar cevaplarını dönerler. Her sorgu yapıldığında ödeme yapılacak diye bir şey söz konusu değildir. Öğrenciler internet bankacılığı ve ATM'lerden borcunu öğrenmek için de borç sorgu yaptırdıklarından borç sorgu için servisimiz belirli dönemlerde yoğun olarak çağrılmaktadır. Şekil 4.1. de görülebileceği gibi WCF servisimiz üç haftalık süreçte toplam 163564 sorguya cevap dönmüştür. Toplam dönülen cevap sayısı olan 163564 'ü 21 güne bölecek olursak bu üç haftalık süreçte bankalardan gelen isteklere günlük ortalama 7788 cevap dönülmüştür.

Borç sorgu yapıldıktan sonra ödeme için servise gelen istekleri yine 2015 Eylül ayının ilk üç haftasındaki veriler (Şekil 4.2) baz alarak incelendi.



Şekil 4.2. Üç haftalık ödeme sorguya dönülen cevapların istatistiği

Sayısal veriler incelendiğinde üç haftalık süreçte ödeme için gelen isteklere servis 60033 cevap dönmüştür. Buradan şu sonuç çıkarılabilir ki WCF servisiyle üç haftalık süreçte yaklaşık olarak altmış bin ödeme gerçekleştirilmiş. Bu dönem ders alma dönemine denk geldiği için sayı oldukça fazladır. Servis bu kadar yoğun ödeme isteği ve çok daha fazla sayıda olan borç sorgu isteğine kısa sürelerde cevap dönmüş bir problem yaşanmamıştır.

4.2. Projenin Tamamlanmasının Getirdiği Yararlar

Online ödeme sistemi tasarımı gerçekleştirilmeden önce öğrenciler bankalardaki üniversitenin ilgili hesap numaralarını söyleyerek T.C. numaraları ya da öğrenci numaraları üzerinden ödemelerini gerçekleştirebiliyorlardı. Hesaba yatırılan ücretin üniversite bilgi sistemine düşebilmesi için çeşitli yollar vardı. Bunlardan birincisi her öğrencinin ödeme yaptığına dair elinde olan dekontunu üniversite ilgili personeline götürüp ödemeyi sisteme işletmesidir. Üniversitede yüzbinlerce öğrenci olduğunu düşünürsek bir personelin bu kadar ödemeyi sisteme işlemesi ve hatasız olarak bu işlemleri gerçekleştirmesi mümkün değildir. Hesaba yatırılan ücretin üniversite bilgi sistemlerine düşmesi için ikinci yöntem ise bankalardaki hesaplara yatırılan ücretlerin bankalar tarafından üniversiteye haftalık ya da aylık excel listeleri halinde gönderilmesiydi. Bu yöntemdeki sıkıntı ise bankaya yatırılan ücretlerin üniversiteye eksik ve geç gönderilmesi durumudur. Bir diğer problem de üniversiteye mail ile gönderilen ne olduğu anlaşılamayan karmaşık listelerin üniversite bilgi sistemine doğru bir şekilde aktarılamaması sebebiyle ortaya çıkan mağduriyetlerdir. Tamamlanan bu proje ile tüm bu sıkıntılara çözüm üretilmiştir. Öğrenciler banka şubeleri, atmler ve internet bankacılığı aracılığıyla hiçbir hesap numarası bilgisine ihtiyaç duymadan sadece öğrenci numaralarını belirtmeleriyle tüm ödemelerini entegrasyon sağlanan bankalarla kolaylıkla gerçekleştirebilirler. Bankalar tarafından alınan ödemeler saniyeler içerisinde üniversite bilgi sistemine aktarılır. Çok nadir de olsa üniversite bilgi sistemine düşmeyen ödemeler varsa, gün sonunda yapılan mutabakat onaylarıyla tespit edilip yeniden üniversite bilgi sitemine gönderilir. Tüm bunların sonucu olarak ne öğrenci, ne banka görevlisi ne de üniversite personelleri bir zorluk yaşamayacaktır.

5. SONUÇ ve ÖNERİLER

Bu çalışmada öncelikle çok geniş kapsamlı bir literatür taraması yapılarak SOA'nın nasıl bir mimari olduğu, nasıl ortaya çıktığı, avantajları ve dezavantajlarının neler olduğu tespit edilmiştir. Dünya üzerinde IBM, Oracle, Microsoft gibi büyük firmalar başta olmak üzere SOA üzerinde uzun yıllardır araştırma ve geliştirmeler yapılmaktadır. Yapılan literatür taraması sonucunda Türkiye'de SOA'nın yeterince keşfedilmediği, kullanılmadığı ve hakkında bilimsel çalışmaların yok denecek kadar az olduğu kanısına varılmıştır.

Bu çalışmayla üniversite bilgi sistemlerinde öğrenci katkı payı, materyal ücreti gibi ödemelerin çok kısa sürede banka şubeleri, atmler ve internet bankacılığı aracılığıyla gerçekleştirilebilmesinin öğrenciler, personel ve banka çalışanları için çok kolaylıklar getirdiği tespit edilmiştir. SOA yaklaşımı ile geliştirilen bu çalışma tamamlanmadan önce üniversite ile bankalar arasında ödeme bilgilerinin transferi offline dosya transfer yöntemiyle gerçekleştirilmiştir. Örnek vermek gerekirse ders kayıt zamanlarında ilgili bankalara ücret yatırması gereken öğrencilerin ödeme miktarlarının elle hesaplanıp bankalara toplu olarak dosya gönderimi yapılmıştır. Borçların dinamik olarak hesaplanması gerekirken personelin hatalı hesap yapması durumunda öğrenciler eksik ya da fazla ücret yatırabilirler. Benzer şekilde bankaların üniversiteye gönderdiği dosya içerisindeki listelerin formatlarından kaynaklanan ya da dosya işlemesini gerçekleştiren personelden kaynaklanan yanlışlıklar sebebiyle, öğrenciler ders alma yapamamakta ve her seferinde bankayla bu kişiler için iletişim kurulması personelin iş yoğunluğunu artırmaktadır. Son yıllarda bazı üniversitelerdeki öğrenci sayıları yüzbinleri bulmuştur. Örgün ve ikinci öğretimin yanı sıra üniversitelerin açıköğretim, uzaktan eğitim ve sürekli eğitim merkezlerinin açılmasıyla öğrenci sayısı ve çeşitlilği artmıştır. Bu şekilde gelişen ve büyüyen bir üniversitenin diğer kurumlarla bilgi paylaşımını çevrimdışı (offline) sistem ile yürütebilmesinin mümkün olamayacağını gözlemledik. Bundan dolayı hem öğrencilerin, hem personelin hem de banka çalışanlarının bu sıkıntılardan kurtulabilmesi için çevrimiçi (online) ödeme sistemi gerekli görülmüştür. Buna çözüm olarak kurumlar arasında kolay entegrasyon sağlayan SOA yaklaşımı ile bu proje

geliştirilmiştir. Microsoft firmasının SOA yaklaşımı için geliştirdiği bir teknoloji olan WCF'in kullanılmasının performans, hız, birlikte çalışılabilirlik gibi konularda avantajlar sağlandığı görülmüştür. Kayıt yenileme ve ders kayıt dönemleri gibi yılın belirli yoğun dönemlerinde servise gelen istek sayıları incelendiğinde günlük ortalama 7-8 bin civarında, maksimum olarak da bir günde yaklaşık olarak elli bin isteğe servis tarafından yanıt verildiği gözlemlenmiştir. Yıllık olarak da yaklaşık olarak bir milyon isteğe servisin yanıt verebildiği tespit edilmiştir.

Materyal ve yöntem kısmında projenin üç önemli ayağı olan WCF servisi, Windows uygulaması ve veri modelinden bahsedilmiştir. WCF servisinin metotlarından, sınıflarından, katmanlarından ve mimari yapısından bahsedip gerekli tanımlamalar şekillerle desteklenmiştir. Bankalar tarafından borç sorgu ve ödeme gibi işlemlerin nasıl yapıldığını gösterebilmek için test amaçlı geliştirilen windows uygulamasının arayüzleri gösterilip, servisle bağlantısının nasıl yapıldığı ve hangi arayüzün arkasında hangi kodların çalıştığı anlatılmıştır. VM'de ödeme sisteminin yürütülebilmesinde ana rol oynayan SP'ler anlatılıp, veri tabanındaki tablolar ve bunlar arasındaki ilişkiler diyagram ile gösterilmiştir.

Son bölümde Microsoft teknolojileri kullanılarak geliştirilen ve çalışmamızın temel bileşenleri olan WCF servisi, Windows uygulaması ve VM anlatıldı. Şekiller üzerinde ayrıntılı bilgiler verilerek projenin iyi anlaşılması amaçlanmıştır. Araştırma bulgularında servisin belirli dönemlerdeki cevap sayısı istatistikleri grafikler yardımıyla anlatılıp yorumlanmıştır.

Gelecek adına bu çalışma geliştirilerek tüm üniversitelerde ödeme sistemleri tek bir merkez üzerinden yürütülebilir. Bütün üniversitelerin SOA yaklaşımı ile geliştirilen bu ödeme sistemini kullanarak ödeme bilgilerini kendi sistemlerine kaydettikleri gibi YÖK'e ya da ilgili başka bir kuruma ait merkezi bir bilgi havuzuna verilerin anlık olarak aktarılmasıyla, ödeme bilgilerine ihtiyaç duyan tüm kurumlara anlık olarak erişim yetkisi verilebilir. Bunun sonucu olarak Yüksek Öğretim Kurumu (YÖK) ya da Maliye Bakanlığı gibi ödemelerle alakalı devlet kurumları, hangi üniversitede ne kadar

ödeme alındığı, hangi öğrencilerin ödeme yaptığı, hangi bankalardan ne kadar tahsilat yapıldığı bilgisine çok sağlıklı bir şekilde saniyeler içerisinde ulaşabilir. Kurumların anlık olarak bu verilere erişebilmesi kurumlar arasındaki resmi yazışmaları, kağıt israfını, zaman kaybını ortadan kaldıracak ve denetim yapabilmeyi daha kolay hale getirebilecektir.

KAYNAKLAR

Anonymous, 2001. http://cse.unl.edu/~reich/XML/soap.html (20.03.2016)

Anonymous, 2014. XML, Simple Easy Learning, http://www.tutorialspoint.com/xml/xml_tutorial.pdf (08.05.2016)

Anonymous, 2015. https://www.akana.com/solutions/security_solutions (20.05.2016)

Anonymous, 2016. http://archive.oreilly.com/pub/a/onjava/excerpt/jws_6/index1.html. (25.02.2016).

Anonymous, 2016a. https://msdn.microsoft.com/en-us/library/ms733128.aspx (28.04.2016)

Anonymous, 2016b. http://www.tutorialspoint.com/uddi/uddi_overview.htm (23.05.2016)

Ahuja, S.P., Patel, A., 2011. Enterprise Service Bus: A Performance Evaluation

Arsanjani, A., 2004, Service-oriented modeling and architecture, How to identify, specify, and realize services for your SOA

Arsanjani, A., Zhang, L.J., Ellis, M., Allam, A., Channabasavaiah, K., 2014. Design an SOA solution using a reference architecture, Improve your development process using the SOA solution stack, https://www.ibm.com/developerworks/library/ar-archtemp/ar-archtemp-pdf.pdf (20.05.2016)

Beklen, A., 2009. Servis Odaklı Mimari, http://edizsaykol.weebly.com/uploads/9/8/6/5/9865252/servis_odakli_mimari.pdf (04.04.2016)

Bokhari, S.M.A., Azam, F., Abbas, M., 2015. Limitations of Service Oriented Architecture and its Combination with Cloud Computing

Cerami, E., 2002. Top Ten FAQs for Web Services, http://webservices.xml.com/lpt/a/1130 (01.05.2016)

Cerami, E., 2002. Web Services Essentials, Distributed Applications with XML-RPC, SOAP, UDDI & WSDL, ISBN: 0-596-00224-6

Chen, D., Doumeingts, G., Vernadat, F., 2008. Architectures for enterprise integration and interoperability: Past, present and future

Coulouris, G., Dollimore, J., Kindberg, T., Blair, G., 2012. DISTRIBUTED SYSTEMS Concepts and Design, Fifth Edition

Cruz, E.F.Z., 2014. Service-Oriented Architectures, Enterprise Service Bus, Middleware from Oracle, and TIBCO

Dikmans, L., Luttikhuizen, R., 2012. SOA Made Simple, Discover the true meaning behind the buzzword that is 'Service Oriented Architecture'

Dirksen, J., 2013. SOA Governance in Action, ISBN: 9781617290275, REST AND WS-* ARCHITECTURES

Dönmez, O., Önal A., Toker L., 2010. Mobil Peer – To –Peer (P2P) Ağlarda Servis Tabanlı Yazılım Geliştirme

Endrei, J., Ang, J., Arsanjani, A., Chua, S., Comte, P., Krogdahl, P., Luo, M., Newling, T., 2004. Patterns: ServiceOriented Architecture and Web Services,IBM Redbook,First Edition

Extreme, 2000. Remote Procedure Calls in SOAP, SoapTeam: Extreme Computing Lab, https://www.extreme.indiana.edu/xgws/papers/sc00_paper/node5.html (16.05.2016)

Flurry, G., 2007. Exploring the Enterprise Service Bus, Part 1: Discover how an ESB can help you meet the requirements for your SOA solution, http://www.ibm.com/developerworks/architecture/library/ar-esbpat1/ (20.05.2016)

Josuttis, N.M., 2007. SOA in Practice, The Art Of Distributed System Design, First Edition

Gehlen, G. and Pham L., 2005. "Mobile Web Services for Peer-to- Peer Applications", Consumer Communications and Networking Conference

Hasan, J., 2004. Expert Service-Oriented Architecture in C#, Using the Web Services Enhancements 2.0, ISBN (pbk): 1-59059-390-1

Herand, D., 2015. Servis Odaklı Mimari, Bir Uygulama Metodolojisi, ISBN:978-605-65066-2-8

Hustad, E. And Lange, C., 2014. Service-oriented architecture projects in practice: A study of a shared, document service implementation

IBM, 2016. IBM Knowledge Center, Web Services, https://www.ibm.com/support/knowledgecenter/SSAW57_8.5.5/com.ibm.websphere.nd.doc/ae/cwbs_wbs2.html

Indrakanti, S., 2012. Service Oriented Architecture Security Risks and their Mitigation, Command, Control, Communications and Intelligence Division Defence Science and Technology Organisation

Informit, 2001. Simple Object Access Protocol(SOAP), http://www.informit.com/articles/article.aspx?p=131328&seqNum=10 (05.05.2016)

Jenkov, J., 2014. SOAP Envelope, http://tutorials.jenkov.com/soap/envelope.html (01.05.2016)

Juric, M.B., Loganathan, R., Sarang, P., Jennings, F., 2007, SOA Approach to Integration XML, Web services, ESB, and BPEL in real-world SOA projects , ISBN 978-1-904811-17-6

Karimi, O., 2011. Security Model For Service-Oriented Architecture

Kart, F., Moser L.E., Melliar, P.M., 2008. Building a Distrubuted E-Healthcare System Using SOA

Keen, M., Coutinho, R., Lippmann, S., Sollami, S., Venkatraman, S., Baber, S., Cui, H., Fleming, C., 2012. Developing Web Services Applications, IBM

Kress, J., Maier, B., Normann, H., Schmeidel, D., Schmutz, G., Trops, B., Utschig, C., Winterberg, T., 2013. Enterprise Servis Bus, http://www.oracle.com/technetwork/articles/soa/ind-soa-esb-1967705.html (20.04.2016)

Lui, M., 2009. What is WCF, https://www.packtpub.com/books/content/wcf-%E2%80%93-windows-communication-foundation (03.06.2016)

Madhusoodanan, H. and Krishnan, N., 2008. Using the SOA ESB pattern to integrate WebSphere Commerce with existing order management systems, http://www.ibm.com/developerworks/websphere/library/techarticles/0806_madhusoodanan/0806_madhusoodanan.html (28.05.2016)

Menge, F., 2007. Enterprise Service Bus

Oracle, 2008. Web Services Security: What's Required To Secure A, Service-Oriented Architecture, An Oracle White Paper, 2008,

http://www.oracle.com/us/products/middleware/identity-management/059410.pdf (15.05.2016)

Osmancık, M.Z., 2013. WCF Service Nedir, http://www.mzekiosmancik.com/2013/03/03/wcf-service-nedir/ (01.06.2016)

Papazoglou, M.P. and Heuvel, W.J.V.D., 2006. "Service-Oriented Design and Development Methodology", Int. J. of Web Engineering and Technology (IJWET)

Pathak, N., 2011. Pro WCF 4, Practical Microsoft SOA Implementation. SECOND EDITION

Patil, Y.V., 2010. What's the Difference between WCF and Web Services?, http://www.codeproject.com/Articles/139787/What-s-the-Difference-between-WCF-and-Web-Services (02.06.2016)

Raines, G., 2009. Cloud Computing and SOA, SERVICE-ORIENTED ARCHITECTURE (SOA) SERIES

Remy, D.L. and Rosenberg, J., 2004. Securing Web Services with WS-Security, ISBN: 0672326515

Serrano, M.A.G., Pérez-Marín, D., Alarcón, M.I., 2014. Architecture Proposal Model for SOA in Universities Educational Software

Stevens, M., 2002. The Benefits of a Service-Oriented Architecture, http://www.developer.com/services/article.php/1041191/The-Benefits-of-a-Service-Oriented-Architecture.htm (12.05.2016)

Techtarget, 2005. UDDI (Universal Description, Discovery, and Integration), http://searchsoa.techtarget.com/definition/UDDI (27.04.2016)

Techtarget, 2014. SOAP (Simple Object Access Protocol), http://searchsoa.techtarget.com/definition/SOAP (24.05.2016)

Valipour, M.H., Amirzafari, B., Maleki, K.N., Daneshpour, N., 2009. A Brief Survey of Software Architecture Conceps and Service Oriented Architecture

Vogel, P., 2011. WCF and Service-Oriented Architectures, https://visualstudiomagazine.com/Articles/2011/06/01/pcnet_WCF-and-SOA.aspx (03.06.2016)

Zamani, A.S., Gupta, P.K., 2015. SOA-Based Distributed System in Online Transaction Processing, Journal of Information Sciences and Computing Technologies(JISCT) ISSN: 2394-9066

ÖZGEÇMİŞ

1983 yılında Bursa'nın Mudanya ilçesinde doğdu. İlk ve orta öğrenimini Ülkü Köy İlköğretim Okulu'nda, lise öğrenimini ise Mudanya Ahmet Rüştü Lisesi'nde tamamladı. 2007 yılında Kocaeli Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümünde lisans eğitimini tamamladı. 2008 yılında Ankara GATA'da askerlik hizmetini yaptı. 2009 yılında KPSS ile Atatürk Üniversitesi'ne Bilgisayar Mühendisi olarak atandı. Bu tarihten itibaren Atatürk Üniversitesi Bilgisayar Bilimleri Araştırma ve Uygulama Merkezinde bilgisayar mühendisi olarak görev yapmaktadır.