YAZILIMLARIN SERVİS OLARAK SUNULMASI: SERVİS-YÖNELİMLİ MİMARİ İLE GELİŞTİRİLEN BİR SESLİ İLETİŞİM UYGULAMASI

Gökhan ÖZTOPUZ

YÜKSEK LİSANS TEZİ

Bilgisayar Mühendisliği Bölümü

TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

Ağustos 2009

ANKARA

Fen Bilimleri Enstitü onayı

_____

Prof. Dr. Ünver KAYNAK

Müdür

Bu tezin Yüksek Lisans derecesinin tüm gereksinimlerini sağladığını onaylarım.

_____

Doç. Dr. Erdoğan DOĞDU

Anabilim Dalı Başkanı

Gökhan ÖZTOPUZ tarafından hazırlanan hazırlanan YAZILIMLARIN SERVİS OLARAK SUNULMASI: SERVİS-YÖNELİMLİ MİMARİ İLE GELİŞTİRİLEN BİR SESLİ İLETİŞİM UYGULAMASI adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

_____

Doç. Dr. Erdoğan DOĞDU

Tez Danışmanı

Tez Jüri Üyeleri

Başkan: Yrd. Doç. Dr. Murat ÖZBAYOĞLU _____

Üye: Doç. Dr. Erdoğan DOĞDU _____

Üye: Yrd. Doç. Dr. Kemal BIÇAKCI _____

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada orijinal olmayan her türlü kaynağa eksiksiz atıf yapıldığını bildiririm.

……..

Gökhan ÖZTOPUZ

Üniversitesi : **TOBB Ekonomi ve Teknoloji Üniversitesi**

Enstitüsü : **Fen Bilimleri**

Anabilim Dalı : **Bilgisayar Mühendisliği**

Tez Danışmanı : **Doç. Dr. Erdoğan DOĞDU**

Tez Türü ve Tarihi : **Yüksek Lisans – Ağustos 2009**

Gökhan ÖZTOPUZ

YAZILIMLARIN SERVİS OLARAK SUNULMASI: SERVİS-YÖNELİMLİ MİMARİ İLE GELİŞTİRİLEN BİR SESLİ İLETİŞİM UYGULAMASI

ÖZET

Son 25 yılda yazılım geliştirme yaklaşımları ve süreçlerinde meydana gelen gelişmeler incelendiğinde, her yeniliğin bir öncekinin tamamlayanı ve geliştirilmiş şekli olarak ortaya çıktığı görülmektedir. Servis-Yönelimli Mimari (Service-Oriented Architecture, SOA) son yıllarda yazılım geliştirmede tercih edilen ve yaygınlık kazanan bir yazılım mimarisi yaklaşımıdır ve hızla gelişen web teknolojilerini barındırmaktadır. SOA'nın standart yazılım mimarilerinden farkı, yazılım sisteminin iç ve dış sistemlerle ve yazılım parçalarıyla etkileşiminin servis-temelli olması, yazılımın ihtiyaç duyduğu işlevleri servis (hizmet) olarak diğer parçalardan alıyor olmasıdır. SOA'nın devamı olarak ortaya çıkan, yazılımların uzaktan kullanımını öngören daha yeni bir yaklaşım ise Servis Yazılımları (Software as a Service, SaaS), yada yazılımların servis olarak sunulmasıdır. SaaS'da sadece yazılımın işlevleri değil, fakat yazılımın tamamı lisanslama yöntemiyle

kullanılabilmektedir. SaaS ekonomik ve işlevsel olarak faydalı bir model olarak kabul görmeye başlamıştır.

Bu tez kapsamında yazılımların servis olarak sunulması (SaaS) konusu incelenmiş, kullanılabilecek teknikler, yöntemler ve bu yaklaşımın getireceği kazançlar sunulmuştur. Ayrıca, iletişim ve ses hizmetlerinin SaaS olarak sunumu incelenmiş, örnek bir ses hizmetleri SaaS uygulaması geliştirilmiş ve sunulmuştur.

Anahtar Kelimeler: XML, Web Servisleri, SOA, SaaS, SIP, WS-*

University : **TOBB University of Economics and Technology**

Institute : **Institute of Natural and Applied Sciences**

Science Programme : **Computer Engineering**

Supervisor : **Associate Professor Dr. Erdogan DOGDU**

Degree Awarded and Date : **M.Sc. – Agust 2009**

Gökhan ÖZTOPUZ

SOFTWARE AS A SERVICE:
A VOICE COMMUNICATION APPLICATION DEVELOPED USING SERVICE-ORIENTED ARCHITECTURE

ABSTRACT

When we look at the software development processe over the last 25 years, we see that each step is a complement and enhancement over the previous step. Service-Oriented Architechture (SOA) is a recently developed software architechture approach that utilizes fast developing web technologies. The difference with SOA is that software components interact with each other and other software applications via service calls, software applications utilize other applications via services. A recent consequence of SOA is to make software applications available for use remotely via an approach called Software as a Service (SaaS), in which not just certain funtionalities of software is made available via services, but the whole software is made available for use remotely as service via licensing. SaaS approach is currently being accepted as an economical and functional model.

In this thesis, we investigate SaaS, its techniques, methods, and its benefits. We also present a voice communication application that we developed using SaaS approach.

Keywords: XML, Web Servisleri, SOA, SaaS, SIP, WS-*

TEŞEKKÜR

Tezi hazırlamamda emeği geçen danışmanım Erdoğan DOĞDU' ya, gerek teknik bilgi gerekse vizyonumu genişletmemde yardımcı olan TOBB ETÜ öğretim üyelerine, bu süreçte bana sabırla destek olan aileme ve yakın arkadaşlarıma, göstermiş oldukları desteklerden dolayı teşekkürü borç bilirim.

İÇİNDEKİLER

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

KISALTMALAR

Kısaltmalar	Açıklama
AJAX	:Asenkron Javascript ve XML
BPEL	:İş Akışı Oluşturma Dili
COM	:Bileşen Nesne Modeli
CORBA	:Ortak Nesne İstem Aracısı Mimarisi
DLL	:Dinamik Bağlantı Kütüphanesi
ESB	:Kurumsal Veri yolu
MSMQ	:Microsoft Mesaj Kuyruk Hizmeti
PSTN	:Genel aktarmalı telefon şebekesi
SAAS	:Yazılımların Servis Olarak Sunulması
SDP	:Oturum Bilgilendirme Protokolü
SDK	:Yazılım Geliştirme Aracı
SIP	: Oturum Başlatma Protokolü
SQL	:Yapısal Sorgulama Dili
SOA	:Servis Yönelimli Mimari
SSML	:Ses Sentezleme İşaretleme Dili
SOAP	:Yalın Nesne Erişim Protokolü
REST	:Durum Bilgisinin Temsili Taşınması
RTP	:Gerçek Zamanlı İletim Protokolü
UDDI	:Genel Tanımlama, Kesif ve Bütünleştirme
W3C	: World Wide Web Konsorsiyumu
WCF	:Windows Haberleşme Mimarisi

WSDL	:Web Servisleri Tanımlama Dili
XML	:Genişletilebilir İşaretleme Dili
VOIP	:Internet Protokolü Üzerinden Ses İletimi

1. GİRİŞ

İnternet'in günlük hayatımıza hızlı ve etkin girişi başta teknolojik, kültürel, sosyal ve ekonomik olmak üzere birçok açıdan bakış açımızı değiştirmiştir. Çoğu kişisel gelişim kitaplarında belirtildiği gibi bilgi toplumunda yer almak her zaman sorunların etkin çözülmesinde yeterli olmayabilir. Bilgiyi etkin bir şekilde kullanabilmek, insanı çoğu zaman salt bilgiyle yüklü olmaktan daha avantajlı kılmaktadır. İnternet'in ilk gelişim safhalarında internet ve web teknolojileri, bilgi sunmak üzerine kuruluydu. Bu bilgiler genel olarak insanların okuyup işlemesine yönelik bir mimariyle birlikte bir çeşit görüntüden oluşmaktaydı. Bu durum verilerin görüntüsünden ayrılıp web üzerinden standart bir şekilde paylaşımına yönelik teknolojilerin geliştirilmesini doğurmuştur.. Bunlarda hepimizin bildiği gibi XML ve buna bağlı teknolojilerdir. XML ve buna bağlı teknolojilerin gelişmesiyle yani verinin görüntüsünden ayrılmasıyla veriler internet üzerinden standart şekilde bilgisayar sistemlerinin işleyebileceği bir hale gelmiştir.

Verilerin internet üzerinden servis olarak sunulması ve sistemlerin karşılıklı olarak birlikte çalışabilmesi fikri servis yönelimli mimariyi etkileyen en önemli unsurlardandır. Servis Yönelimli Mimari Web Servislerinin ve buna bağlı olan WS-* Protokollerinin kabul görmesi ile kendi uygulanabilme alanlarını ve olanaklarını arttırmıştır. 2000'li yılların başında da Uygulama Servis Sağlayıcıları kavramı gündeme gelmiştir. Burada uygulama uzak bir sunucu da bulunmakta ve çalıştırılmaktadır. Fakat bu kavramda bir bakıma internet yerel ağ yerine klavye, fare ve monitör kablosunun uzatma görevini üstlenmiştir. Ve verimlilik ve karlılık açısından ise çok da cazip imkanlar sunamamıştır. Bu noktada servis yönelimli mimari ve uygulamaların servis olarak sunulması kavramsal olarak farklı olsalar da uygun mimaride kullanıldıklarında birbirlerini mükemmel şekilde tamamlayan teknolojilerdir [1]. Gerek uygulamaların servis olarak sunulması, gerekse servis yönelimli mimaride web servislerinin zorunlu olmamasına rağmen bu teknolojilerden web servisleriyle daha fazla verim elde edilebilmektedir [1].

Kurumsal anlamda bu gelişmeler yaşanırken internet iletişimde bireysel anlamda da hızlı bir etki göstermiştir. İnternet toplum içerisinde, sosyal ortamlarda ve küçük kurumlarda belki de beklenenden daha fazla uyum sağlamıştır. İlk başlarda web sayfası yalnızca kurumlar için bir itibar sayılırken, gelişmeler çerçevesinde bireyler ve ufak kurumlar için de iş yapılacak bir ortam olarak görülmeye başlanmıştır. Fakat bunları gerçekleştirmek beraberinde kişilere ve şirketlere ek maliyetler getirmektedir. Yazılım dünyası ve iş dünyasındaki hızlı değişimler beraberinde bu yazılımların yönetimi, bakımı, kurulması ve güncellenmesi problemlerini de beraberinde getirmektedir. Şirketler bir taraftan rekabetçi bir piyasada kaliteli yazılımlara ihtiyaç duymakta, bir yandan da bunların getireceği maliyetleri hesaplamaktadır. Bir orta ölçekli firma için hazırlanacak olan bir Satın Alma Bilgi Sisteminin maliyeti 100.000lerce TL ye mal olabilmekte ve bunun bakımı, güncellemesi de ek bir maliyet getirmektedir. Bu durumlar ve web servisleriyle ilgili teknolojilerinin hızlı gelişimi, yazılımı ve yazılım bileşenlerini servis olarak sunulmasına olanak tanımıştır. Bununla bir yerde yazılmış başarılı bir kod veya uygulama, servis olarak sunularak diğer servis istemcilerin kullanıma çok rahat sunulabilir. Bu sayede birbirine benzer kodların tekrar tekrar yazılmasına gerek kalmadan yazılım dünyası, başka problem çözümlerine odaklanabilir ve karlılık ve verimlilik artışı sağlanabilir [3][5][7][8].

1.1 Çalışmanın Amacı

Çalışmamızda bir ses hizmetleri yazılımının servis yönelimli mimaride tasarlanması ve servis olarak sunulması incelenmiştir. Bu tür yazılımların etkin bir şekilde nasıl gerçekleştirilebileceği ve iletişim hizmetlerinin servis olarak sunulmasının getirebileceği faydalar üzerinde durulmuştur. Bu çalışmanın temelinde her kurumun, her kişinin ve hatta her cihazın web servisine ihtiyacı olduğu fikri yatmaktadır. Servis yönelimli mimari ve web servislerinin temel uygulama alanı veritabanı ve verilerin paylaşımı olarak görülmüştür, hatta çoğu kurum tarafından XML ve web servisleri denildiğinde kurumsal bütünleşme anlaşılmaktadır. Bu da web servislerinin ve XML teknolojilerinin uygulama alanlarını dar bir kapsamda görülmesine neden olabilmektedir. Bu kapsamda çalışmamızın bir amacı da web servislerinin uygulama

alanlarının normal bireyler, günlük yazılımlar ve hatta cihazlar için de geçerli olabileceğidir. Bunun için günlük hayatta iletişim için sık olarak kullandığımız ses hizmetlerinin web servisleri yardımıyla bireyler ve XML teknolojilerini kullanabilecek olan her türlü cihaz için kullanıma sunulması hedeflenmiştir. Bu duruma pazarlama açısından yaklaşırsak "Herkesin telefon açmaya ihtiyacı var. Peki, neden yazılımlarımızın ve cihazlarımızın da buna ihtiyacı olmasın?" şeklinde bir ifade ortaya koyabiliriz.

Birey veya bir yazılım geliştirici olarak bu hizmetleri uygulamalarımıza eklemek oldukça maliyetli olabilmektedir. Bir örnekle açıklamak gerekirse yapay zeka üzerine uzmanlaşmış bir ekip, yazılımlarında uygulamanın gerekli gördüğü durumlar için telefon açma yeteneğinin bulunmasını istemektedir. Bu, onlar için tamamen farklı bir uzmanlık alanı olabilmekte ve projelerine bu yeteneği eklemenin maliyeti normal proje maliyeti kadar olabilmektedir. Bu kodun bakımı, kurulması da ayrı bir problem doğurabilmektedir. Ama tez kapsamında hazırlanan web servisleri aracılığı ile bu kodlara telefon açma yeteneğinin kazandırılması yolunda projelere 4-5 satırlık kod eklemek yeterli olacaktır. Aynı şekilde bir Excel uygulamasının da hücrelerindeki bilgileri belirli durumlarda internet üzerinden telefonla bildirmesi gerekebilir. Bu da sunucuda bulanacak olan web servisinin kullanımı ve arka planda ses sentezleme bileşenlerinin kullanımı yanında ses hizmet bileşenlerinin yardımı ve Excel istemcisindeki 4-5 satırlık makro kodu aracılığıyla gerçekleşecektir.

Projemiz iki yönlü olarak geliştirilmiştir. Birinci yön olarak ses hizmetlerinin doğrudan bir uygulama olarak sunulması ve ikinci yön olarak servis ve istemciler olarak düşünülebilir. Fakat bunlar ön planda farklı görülebilmekle birlikte gerek veri gerekse iletişim olarak aynı bileşenleri kullanmaktadır. Servis olarak sunulan bu uygulamada firmalar uygulamayı kendileri için hazırlanmış bir uygulama olarak görebilmekte ve yerelleştirebilmektedir. Bu sayede ortak bir veritabanını kendilerine özgü bir şekilde görebilmektedirler. Servis yönelimli bileşenler gerek yerel ağda gerekse internet üzerinde kullanılarak kod tekrarları en aza iletilmiş ve bileşenlerin birbiriyle uyumunu standartlar çerçevesinde en üst düzeye çıkarmıştır.

Tezin ikinci bölümünde proje kapsamında uygulamaların servis olarak sunulması ile ilgili bilgiler verilmiştir. Bu kapsamda tez içerisinde kullanılan teknikler ve yapılan çalışmalar ile ilgili bilgiler sunulmaktadır. Üçüncü bölümde örnek çözümümüz servis yönelimli olarak gerçekleştirildiğinden bu konu ile ilgili temel kavramlara ve web servislerinin temel yapıtaşları ile ilgili bilgilere yer verilmektedir. Dördüncü bölümde asıl sunulacak ses hizmetinin sunulması ile ilgili teknik bilgiler bulunmaktadır. Internet üzerinde ses hizmetlerinin sunulmasında kullanılan mimari ve standartlardan bahsedilmekte bahsedilmektedir. Tezin beşinci bölümünde sistem gerçekleştirimi detaylı ve basamaklı bir şekilde anlatılmıştır. Tercih edilen yöntemler, nedenleriyle birlikte anlatılmış, gerekli yerlerde kod örneklerine yer verilmiştir.

Ayrıca bu bölümde şu ana kadar yapılan çalışmalardan farklı olarak Uygulamaların Servis olarak sunulmasındaki verilerin dağıtık olması kavramına değinilmiş ve bunun için çift yönlü iletişim örüntüsüne sahip bir servisle bu durumun nasıl ele alınabileceği üzerinde durulmuştur. Böylelilikle uygulamaların servis olarak sunulmasında kritik veriler merkezi bir yerde rakip firmalarla alt alta bulunmadan veriler istemci uygulamadan elde edilecektir. Örnek olarak arama mesajının ve telefon numarasının arama vakti geldiğinde servisin istemciden bu bilgilerin elde etmesi ve arama işleminin gerçekleştirilmesi.

Altıncı bölümde tanıtılan teknolojilerin servisi içerisinde nasıl kullanılabileceği üzerinde durularak çözüm önerileri genişletilmiştir.

Çalışmamızın amacını “kurumlara, kişilere ve cihazlara hizmet verecek bir sesli iletişim alt yapısı sunmak” şeklinde özetleyebiliriz. Burada uygulamaların internet üzerinden servis olarak sunulmasında ortaya koyacağı verimlilik ve karlılık üzerinde çalışmalar yapılmıştır. Örnek çözümde kurumsal bütünleşme uygulamaları yerine, telefon ve ses iletim hizmetleri üzerine çözümlerin kullanıcıların hizmetine sunularak günlük hayatları içerisinde de kullanabilecekleri bir çözüm önerisinde bulunulmuştur. Şu ana kadar iletişimin servis olarak sunulmasında çeşitli çalışmalar [68]’de incelenmiştir. Fakat bu çalışmamızın temelinde de geçmiş web servisi çalışmalarımız bulunmaktadır [63].

2. YAZILIMLARIN SERVİS OLARAK SUNULMASI (SaaS)

Bu bölümde tez kapsamında incelenen ve uygulanan yazılımların servis olarak sunulması kavramına genel bir bakış açısıyla yaklaşılıp bu konudaki kavramlar incelenecektir. Yazılımların servis olarak sunulmasının getirileri ve zorlukları üzerinde durulduktan sonra mimari yaklaşımlar ele alınacaktır. Sonrasında ise veritabanlarının yapıları, güvenlik, verilerin taşınması, denetleme, yapılandırma, istemciler ve dağıtım konularına maddeler halinde değinilecektir.

2.1 SaaS GİRİŞ

Yazılımların servis olarak sunulması (Software as a Service-SaaS) kısaca "Internet üzerinden yayınlanan ve erişilen yazılımlar" [13] olarak tanımlanabilir. Yazılımların internet üzerinden çalıştırılması yazılımlara bakış olan açısını, hem iş yönünden, hem de yazılımların mimari açıdan değiştirebilecektir [13]. Günümüzde e-dönüşüm küçük ve orta ölçekli firmalar için kaçınılmaz bir hale gelmiştir. E-dönüşüm sürecinde en önemli etkenlerden birisi de işletmelerin kullanacakları yazılımlardır. Buna karşın sunucu üzerinden çalıştırılması gereken yazılımlar bu firmalar için yazılımsal, donanımsal, kurulum ve de operasyon maliyeti olarak yakın bir çözüm olarak görülmemesi olağandır. Ayrıca günümüz ekonomik koşullarında yeni kurulum aşamasında olan işletmeler kaliteli, internet destekli yazılımlara ihtiyaçları olmasına karşın o alanda ne kadar süre iş yapacaklarını tahmin etmekte zorlanmaktadır. Bu durumda işletme sahipleri başlangıç aşamasında yazılım ve donanımsal anlamda yüksek maliyetleri bulan çözümlere yönelememekte ve de e-dönüşüm süreci içerisine istedikleri kadar girmekte zorlanmaktadırlar. Bu durum da rekabetçi bir ortamda onlar için bir dezavantaj olarak karşılarına çıkabilmektedir. Fakat SaaS uygulamalarında başlangıç maliyeti olmadan da işletmeler çalışmalarına başlayabilmektedir [2]. İşletmeler için diğer önemli noktalardan birisi de nitelikli eleman bulma ve elamanların şirketlere yüksek maliyetleri sorunudur. Yazılımların servis sağlanan firmanın sunucularında bulunması ve burada yönetilmesi işletmelerin yazılım operasyon maliyetlerini düşürmektedir [5][8][10]. Böylelikle yeni kurulan işletmelerin başlangıç olarak yazılım, donanım ve operasyon maliyetlerini en aza indirmekte [5][4][8][10] ve işletmeler e-dönüşüm süreçlerini hızlı bir şekilde

gerçekleştirerek, büyük veri merkezleri bulunan işletmelerle bilgi işlem açısından rekabet edebilir bir hale gelebilmektedir.

Günümüzün değişen bilgi toplumunda ve iş dünyasında yazılımlar kısa sürede güncelliğini kaybedebilmektedir. Klasik yazılımlarda destek anlaşması bitmişse yazılım güncellemesinin maddi açıdan ve de daha önemlisi doğru şekilde güncellenip kurulması bir problem olmaktadır. SaaS uygulamalarında, temelde arka planda bir yazılım bulunacağından, o yazılım üzerinde yapılacak olan güncelleme ile tüm işletmeler güncel yazılımları kullanabilecektir [5][10]. Yazılımların dağıtılması problemi de kendiliğinden çözümlenmiş olur. Yazılım geliştiren firmalar açısından donanımsal ve operasyon maliyetlerinin yüksek oluşu yazılımdan elde ettikleri karları kısmalarına neden olabilmektedir. Bu durum kısmen aşılmış ve yazılımların korsan olarak dağıtılması da engellenmiş durumdadır [5]. Fakat bununla beraber yazılımların tasarım karmaşıklığı biraz daha artacağından SaaS uygulamalarının başlangıç, geliştirme ve bakım maliyeti tek bir uygulamaya göre biraz daha yüksek olabilecektir [5].

İşletmeler için verileri hayati önem taşıyabilmektedir. Çoğu işletme için verilerin başkalarında olması ürkütücü bir durum olabilmektedir. Bu durumda yazılım hizmeti verecek olan firmaların sorumluluğu biraz daha artacaktır. Verilerin güvenliği ile detaylı bilgiler 2.3. bölümde incelenmiştir. Diğer bir konuda kullanıcıların uygulama içersindeki yetkilendirilmesidir. Belirli işlemleri sadece, yönetici tarafından kendisine yetki verilen kişiler yapabilmelidir. Bu kişiler verileri izin verildiği ölçüde inceleyebilmeli ve düzenleyebilmelidir. Mevcut uygulamaları çok değişik sektörlerden farklı firmalar farklı şekilde uygulayabilmelidir. Bunun için esnek bir yapılandırma yapısıyla işletmeler kiraladıkları yazılımları istekleri ölçüsünde düzenleyebilmeli ve gerekirse iş akış süreçlerine müdahale edebilmelidirler. Servis sağlayıcı işletmelere yazılımların erişimi ve kullanımı konusunda belirli taahhütlerde bulunulmalı ve işletmeler de ödeyecekleri ücretin karşılığını bu taahhütler karşısında beklemelidir. Servislerin yaygınlaşmasıyla beraber yazılım firmaları hızlı bir şekilde yazılımlarını yayınlamak isteyecektir. Arka planda tüm servisler tarafından kullanılabilecek olan bileşenler servis platformları tarafından sağlandığı sürece bu ortak bileşenler tekrar kodlanmadan platform tarafından sağlanabilir [2]. Bu

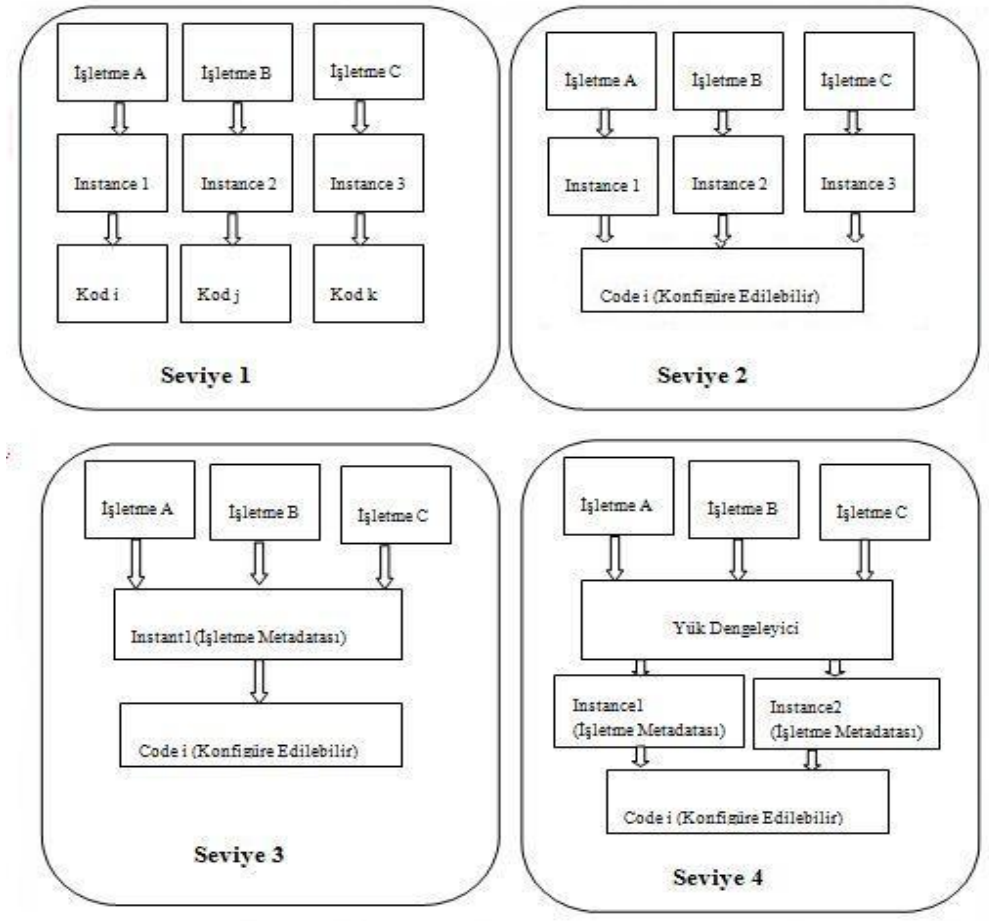
konulardaki incelemeler 2.11. bölümdeki yazılımların yayınlanması konusunda incelenmiştir.

Yazılımların servis olarak sunulması temellerini Uygulama Servis Sağlayıcılara (ASP) dayandırmaktadır [3]. ASP'ler uygulama odaklı iken SaaS servis odaklı bir mimariye sahiptir ve ASP'lerde uygulamalar tek kiracı modeline göre geliştirilmektedir [13][5]. Fakat bahsettiğimiz yazılımların servis olarak sunulmasında kazanım ve başarıların temelinde çoklu kiracı modeli yatmaktadır. ASP'lerdeki sunucu yazılımlarının bakımı da yazılımı kullanan işletmelerde olabilmekte bu da yazılımı kullanan işletmelerin yükünü arttırabilmektedir [3][5]. Yazılımların servis olarak kullanılması ile ASP'lerden farklı olarak gerekli programlama arabirimlerinin yayıncı firmalar tarafından sağlanması ile farklı bölümlerdeki mevcut yazılım ve yeni eklenmesi muhtemel yazılımlarla şirket içi tümleşik yazılımlar geliştirilebilinir [15]. Şu an kullanılmakta olan en popüler SaaS uygulamalarına örnek vermek istersek SalesForce CRM [50], WinSaaS [51], Litware HR [52] olarak sıralayabiliriz. Bu uygulamalar bölüm içerisinde de bahsedeceğimiz gibi varsayılan olarak tek bir Instance üzerinden çalışmakta ve paylaşımlı bir veritabanı kullanmaktadırlar.

2.2 Mimari Açıdan İncelenmesi

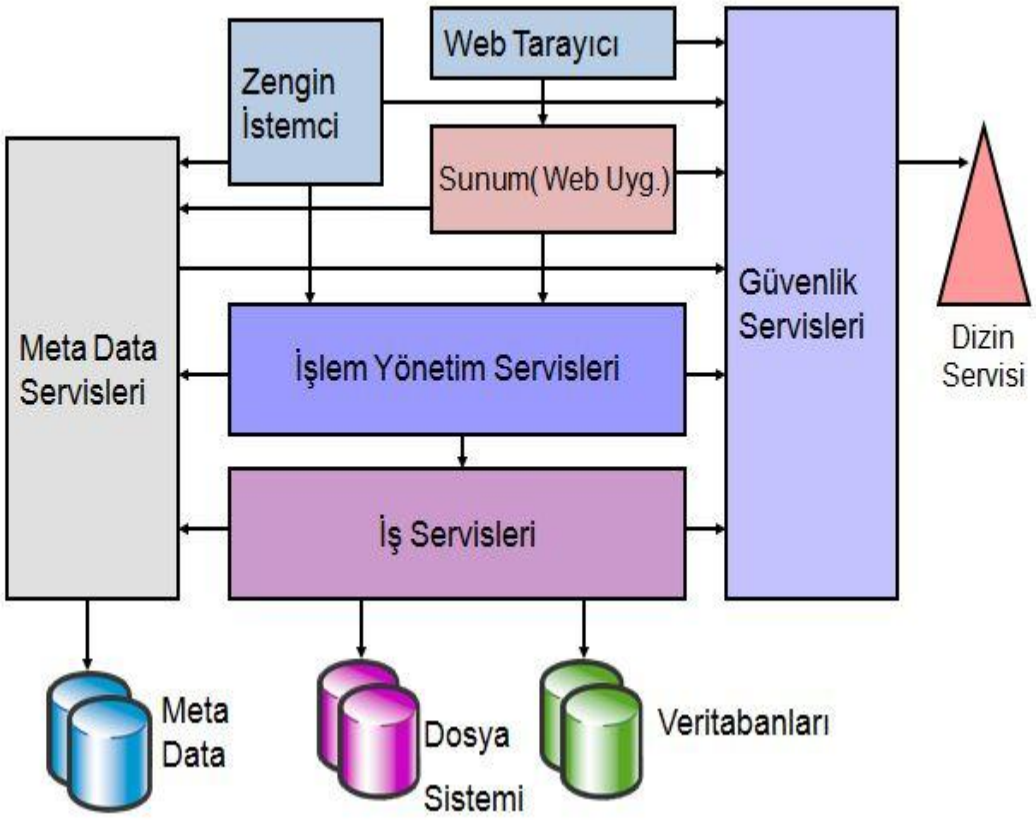
Gelişmiş bir SaaS uygulamasını zayıf olanlarından ayıran en önemli özellikler kaynakların birbirinden yalıtımı, ölçeklenebilirliği, çoklu kiracı desteği ve yapılandırılabilirlik olarak gösterilebilir [13][8]. SaaS modelinde çoklu kiracı modeli, dört seviye olgunluk seviyesinde gösterilir [3]. Seviye 1'de uygulama instanceları birbirinden bağımsız olarak çalışmaktadır. Klasik istemci sunucu uygulamaları fazla mimari değişiklik gerektirmeden ilk seviye olgunluk modeline taşınabilir [13]. Seviye 1 de tüm işletmelerin özelleşmiş farklı kod tabanında olması ASP'lerin çalışma mimarisiyle benzerlik gösterecektir [3]. İkinci seviye olgunluk modelinde Şekil 2.1'den de görülebileceği gibi Seviye 1 den farklı olarak tek bir kod tabanından uygulamalar çalıştırılmaktadır. Bu kod tabanında işletmelere çeşitli şekillerde konfigürasyon seçenekleri sunulmaktadır [3]. Bu seçenekler uygulamaların kullanıcılar tarafından görsel olarak özelleştirilmesinden, iş akışını düzenlemeye

kadar geniş bir yelpazede olabilmektedir. Tüm işletmeler temelde aynı kod tabanını kullanmalarına rağmen tüm kullanıcılar biriringen tamamıyla soyutlanmıştır [13]. Üçüncü seviye olgunluk modelinde tek bir instance Seviye 2 den farklı olarak tek bir instance çalışmaktadır. Tek bir instance oluşu sistemin geniş kapasite ihtiyacını azaltır. Konfigüre edilebilir metadata sayeside her bir kullanıcı için uygulamalar farklı şekillerde çalışabilmekte ve kullanıcılar arka planda tek bir uygulama olduğunu hissetmemektedirler [13]. Seviye 4 de ise konfigüre edilebilir metadata verileriyle beraber sistemin yüksek kapasitelerde çalışması bir yük dengeleyici tarafından kontrol edilmektedir. Yük dengeleyicinin olması, ölçekleme konusunda instance sayısının kolaylıkla arttırılmasıyla birlikte kolaylıklar sağlayabilmektedir [3][13].



Şekil 2.1 Servis Olgunluk Modelleri [13][3]

İşletmeler bu olgunluk seviye modellerinden iş şekillerine göre, programlama mimarilerine göre ve de sistemin yönetilmesindeki tercihlerin uygunluğuna göre bir olgunluk seviyesini seçmektedir.



Şekil 2.2 Örnek SaaS Uygulama Mimarisi [13]

Şekil 2.2'de bir SaaS uygulamasının bileşenler açısından nasıl tasarlanabileceği gösterilmiştir. Uygulamayı kullanacak olan işletmeler yazılımlara masaüstü uygulamalarıyla ve de web tarayıcılarıyla erişebilmektedir. Web tarayıcıları için işlemlerin geliştirilmesi için gelişmiş bir sunucu taraflı bir web uygulaması bulunmalıdır. İşlem Yönetim servisleri uygulama içersindeki iş akışlarını düzenlerken iş servisleri uygulamaların ana iş mantığını ve veri tabanıyla olan bağlantılarını gerçekleştirir [3]. Güvenlik ve yetkilendirme her aşamada ve her katmanda gerekli olan noktalardan birisidir. Uygulamayı kullanan işletmeler güvenliği yeni nesil web servislerinin getirdiği web servisleri güvenlik

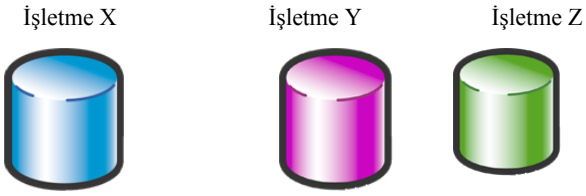
protokolleriyle ve de SSL sertifikalarını kullanarak gerçekleştirebilmektedirler. Her işletme temel de aynı uygulamayı kullanırken her birinin sanki arka planında farklı uygulama kullandığını hissettiren metadata servisleridir. Şekil 2.2'den de görüleceği gibi servis kodu, metadata servislerine her katmanda başvurarak o işletme için gerekli düzenlemeleri alır ve işlemlerini o ayarlara göre gerçekleştirebilmektedir [3][13].

2.3 SaaS Veri Tabanı Modelleri

Veriler işletmeler için hayati önem taşımaktadır. SaaS uygulamalarında işletmelerin verileri başka ortak sunucularda bulunacağı için bu SaaS yazılımını tasarlayan firmalar veritabanı konusunda oldukça hassas olmalıdır. Bu bölümde ise SaaS veritabanlarının tasarımı üzerinde durulmaktadır. Veri tabanının büyüklüğü ve kullanıcı sayısı veritabanın tasarımındaki en önemli etkenlerdir. Verilerin başkalarının sunucularında olması ve de veritabanlarının o kişiler için geniş kullanıcı profiline göre bu uygulamaları tasarlama gerekliliği, veritabanı mimarilerini tasarlamadaki en önemli noktalardan birisidir. Gelişmiş SaaS uygulamalarında şirketler kendileri için özel alanlar oluşturabilmeli ve bunu sisteme dahil edebilmelilerdir [9].

IBM, Microsoft, Oracle ve diğer örnek uygulama hazırlayıcıları SaaS uygulamalarında 3 tür veritabanı yaklaşımını önermektedir [8].

Bu yaklaşımlardan ilki Şekil3'deki gibi tüm işletmeler için aynı uygulama için ayrı bir veritabanı oluşturmaktır. Burada tüm işletmeler farklı veritabanı kullanacaklarından verilerin işletmeler arasında soyutlanması mantık olarak gerçekleştirilmektedir [14][8][7]. Ayrıca bu yaklaşımda veritabanı o işletmeye özel olacağından güvenlik açısından daha yüksek bir seviyede olacaktır. Veritabanının yedeklemesi, geri yüklemesi gibi işlemler kolaylıkla ve firmalar istedikleri zaman kendi başlarına bile yapabilmektedir. Buna ek olarak işletmeler veritabanı tabloları üzerinde alan ekleme, çıkarma düzenleme gibi işlemleri gerçekleştirebileceklerinden işletmeler için esnek bir yapı sunabilirler.

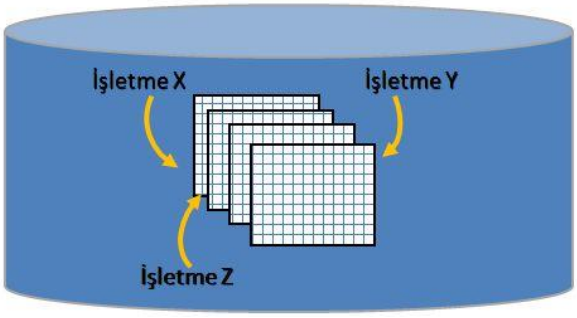


Şekil 2.3 Her işletme için Ayrı veri tabanı [14]

Fakat veri tabanı sayısının fazla oluşu sistemde yazılımsal ve de donanımsal büyüklükleri gerektirecektir [14]. Buna paralel olarak bakım maliyetleri de oldukça artacaktır [19].

İkinci veritabanı yaklaşımı aynı veritabanı üzerinde farklı şema kullanma yaklaşımıdır [14][8][7]. Burada yazılımı kullanan işletmeler aynı veritabanı üzerinde farklı şemaları kullanarak kendi verileri diğer verilerden işlem bakımından soyutlayabilir. Her işletme için ayrı bir şema olacağından veritabanı üzerinde şema oluşturma işlemi genellikle şirketlerin sisteme ilk kayıtları sırasında olmaktadır bunun için gerekli SQL komutları veritabanı sistemlerinde mevcuttur [14]. Öncelikli olarak bu yaklaşımda işletmeler veritabanı yapıları üzerinde istedikleri gibi düzenleme yapma şansına sahiptir. Böylelikle veritabanı üzerinde ihtiyaçlar doğrultusunda eklenecek olan yeni alanlar diğer işletmelerin şemaları farklı olacağı için bir sorun oluşturmayacaktır. Güvenlik açısından aynı veritabanında olmalarına rağmen şema farklı olacağından diğer işletmelerin veritabanı erişimi mümkün olmayacaktır. Buradaki bir diğer avantaj ilk bölümde bahsedilen yüksek yazılım ve donanım kapasitesi gereksiniminin, tek veritabanlı olması nedeniyle daha düşük olacaktır.

Üçüncü yaklaşımsa ortak şema ve veri tabanı paylaşımıdır [14][8][7]. Burada Şekil 2.4'deki gibi tek bir veritabanı ve de şeması bulunmaktadır.



Şekil 2.4 Ortak Şema ve Veri Tabanı Paylaşımı [15] [14][7]

SaaS uygulamalarındaki önemli noktalardan birisi olarak yazılımı kullanacak olan işletmelerin yazılımdaki verilerle sınırlı kalmamasıdır. Bu yüzden uygulamalarda işletmeler kendi veri alanlarını ve o alanların türlerini tanımlayabilmeli ve yazılımı istedikleri gibi kendi ihtiyaçları doğrultusunda genişletebilmelidir. Bu konuda çeşitli yaklaşımların olmasına karşın temelde üç tür yaklaşım ön plandadır. Bunlar önceden ayrılmış alanlar, isim-değer eşlemeleri ve işletmeye özel alanlardır [14][8].

İlk olarak önceden ayrılmış alanlar yapısına balkıdığında, veritabanı tasarımcısı veritabanının genişletebilir olması için önceden ayrılmış alanlar tanımlamıştır [3][8] [14]. Bu alanlar uygulama üzerinde genişletme ihtiyacı olmadığı sürece boş olarak kalacaktır. Şekil 2.5'de bu yapının gösterimi görülmektedir.

İşletmeNo	PersonelAdi	Telefon	BolumNo	Ozel1	Ozel2	Ozel3
100	Gokhan	11111	10	"Bilgisayar"	null	null
110	Hakan	2222	20	null	Samsun	null
100	Kağan	3333	5	"Elektronik"	null	null
120	Çetin	4444	20	null	null	"Evet"

Şekil 2.5 SaaS Tablolarında Özel Alanlar [14]

Şekil 2.5'de veritabanı paylaşımlı veritabanı ve şema yapısına sahiptir ve işletmeler işletme verileri işletme numaralarına göre birbirlerinden ayrılmaktadırlar. Her işletme kendi gereksinimlerine göre veritabanında bulunan bu alanları

doldurabilmekte böylelikle uygulamayı kendi isteklerine göre özelleştirebilme şansına sahip olabilmektedirler [3][8]. Buradaki dikkat edilmesi gereken noktalardan birisi de bu verilerin uygulamalar için hem sunum katmanında hem de iş mantığı katmanında yeni alanlar olacağı için kodlar bu veriler değerlendirilecektir. Kodlar bu verileri işlerken ve süreçlere dahil ederken bu verilerin hangi türde veriler olacağının bilinmesi gerekebilir [14]. Bu durumda işletme için ayrılan özel alanlara bu alanların türlerini ve etiketlerini belirten ayrı bir meta veri tablosu oluşturabilir ve bu sayede tabloda string olarak depolanan veriler kod içerisinde hangi tip verilere dönüştürüleceği görülebilir [14].

Eğer özel alanların sayısı fazlaysa ve de bu alanları işletmelerin çoğu az sayıda özel alan kullanıyorsa boş alanların sayısı fazla olacaktır ve de veritabanında boş yer işgali gerçekleşecektir. Bu durumu aşmak için Şekil 2.6'daki gibi özel her bir özel alan verisi ayrı bir tabloda tutulabilir [14][8]. Veya işletmeler daha fazla özel alana ihtiyaç duyacaklarsa bu konuda daha esnek bir tablo lama yapısına gereksinim duyabilirler. Bu konudaki çözüm isim-değer eşleşmesi çözümüdür [14][3][8].

Şekil 2.6 Örnek SaaS Veritabanı İsim Değer Eşlemesi Genişletme Yapısı [14][3][8]

Şekil 2.6'dan da görülebileceği gibi ana veri tablosunda tek eklentiler için tek bir alan (EkId) ayrılmıştır. İşletmeler kendi kayıtları için belirledikleri özel alanları başka bir tabloda tutarak bu bilgiyle eşleştirme yapabilirler ve bu sayede uygulamanın genişletilebilirliği önceden sınırlanmamış olur. Buradaki dezavantaj sorguların daha karmaşık bir yapıda olması ve bunun da performans kaybına yol açabilmesidir [14] [3].

Eğer ayrı veritabanı ve ayrı şema kullanılacaksa kullanabileceğimiz çözümlerden birisi de veritabanına doğrudan işletmelerin özel sütun eklemeleri olacaktır. Fakat burada arka planda tek bir uygulama çalışacağından aynı kod her işletme için farklı sayıda alanı düzenlemek zorunda kalacaktır. Bu kod açısından uygulanması kolay olmayan bir durumdur [14][9][8].

2.4 SaaS Veri Güvenliği

Yazılımların Servis olarak sunulmasında verilerin güvenliği denildiğinde akla ilk gelecek konulardan birisi de verilerin tutulacağı veri tabanı güvenliği olacaktır. İşletmeler kendileri için hayati öneme sahip bilgileri başka bir sunucuda internet ortamında erişmesinin riskini taşır. Bunun yanında işletmeler kendi alanlarındaki aynı yazılımı kullanan rakip işletmelerle de aynı veritabanını hatta aynı tabloları kullanmak durumunda olacaktır. Diğer bir konu da aynı işletme içerisinde çalışanların izin verilen bilgilere erişmesi durumudur. Bunun için hem uygulama düzeyinde hem de kullanıcılar arasında veriler iyi bir biçimde soyutlanmalı, yetkisiz işlem yapılmamalı ve firmalar gerekli gördükleri takdirde verilerini kendileri özel olarak tüm verileri veya belirli alanları şifreli olarak tutabilmelidirler.

Tüm işletmeler için ortak şema yani ortak tablo kullanılma durumunda SQL sorgusunda işletmeler arasındaki ayrım sql sorgusunun where bölümüne yazılacak olan ek bir işletme koduyla sağlanabilir. Bu durumda SQL Injection saldırıları önceden iyi planlanmalı ve sorgular ona göre düzenlenmelidir. Where cümlesine yazılacak ufak bir sql kodu ile diğer işletme kodu ile yapılan ayrım ortadan kalkabilir ve işletmeler diğer işletmenin bilgilerine de erişebilir [19][8]. Bu ve benzeri

saldırılara karşı yazılım tasarım aşamasındayken tehdit planlaması yapılmalı ve alınacak olan önlemler önceden belirlenmelidir. Kullanıcılardan gelecek her türlü bilgi içerisinde tehdit olabileceği düşünülüp kullanıcıların veritabanına ekleyeceği veya işlem yapmak için servis koduna göndereceği bilgiler uzunluk ve tip kontrolünden geçirilmelidir. Ayrıca program kodu içerisinde alabileceği muhtemel değerler için de ek bir kontrol yapılması uygun olabilir. Bunu için öncelikli olarak servis kodu tarafında metin birleştirmesi ile sql ifadesi oluşturmaktan kaçınılmalı, düzenli ifadeler (regular expression) etkin bir biçimde kullanılmalı, veritabanı sunucusunda ise kayıtlı yordamlar etkin bir biçim de kullanılmalıdır. Veri tabanı içerisinde işletmeler için ayrı ayrı görünüp (view) oluşturup kullanmak kullanıcıların diğer işletmelerin verilerini elde etmesini güçleştirecektir [14]. Çünkü bu sayede işletmeler sorgu ve işlemlerini kendi aynı tabloda olmalarına rağmen kendi görünümleri üzerinden gerçekleştireceklerdir. Bu şekilde işletmelere sadece görünüme erişim izni verilip tabloya doğrudan erişimleri kısıtlanabilir [14][8].

Farklı veritabanı veya şema yaklaşımının veritabanı tasarımında kullanılması durumunda ise tablolar veya görünümler üzerinde SQL cümlesiyle gelen grant komutuyla belirli tablolara belirli kullanıcıların ayrı ayrı Seçme, Ekleme, Silme ve Düzenleme yetkileri kullanıcı bazında verilerek güvenlik sağlanabilir [14][8].

Veritabanı güvenliğinde düşünülmesi gereken durumlardan bir diğeri ise veritabanı üzerindeki bilgilerin şifrelenmesi durumudur. Güvenlik önlemleri ne kadar güçlü olursa olsun verilerin başka ve yetkisiz kişilerin eline geçmesi hem işletmeler hem de hizmeti veren yazılım firmaları için oldukça riskli bir durumdur. Bu yüzden güvenlik önlemlerine ek olarak ayrıca veriler tablolarda şifrelenmiş bir şekilde de durabilir. Bu sayede veriler başkalarının eline geçme durumunda bile şifreli olacağı için bir anlam ifade etmeyecektir. Veri tabanı şifrelemesinde simetrik ve asimetrik şifreleme metotları kullanılabilmektedir. Asimetrik şifreleme fazladan mevcut şifreleme üzerine ek bir şifreleme yaparak sistemin çalışma performansını düşürebilir [14]. Bu durumda simetrik şifreleme metodu performans açısından daha öne çıkmaktadır. Her işletme için ayrı bir simetrik anahtar oluşturarak ve de şifreli veriyi açacak olan ek bir anahtarlama yöntemiyle veya veri tabanı sunucusunda bulunabilecek ve

kullanıcıların sisteme ilk kayıt esnalarında oluşturulan bir sertifika ile bu durum aşılabilir [14].

Şekil 2.7 Örnek Veritabanı Alan Şifrelemesi

2.5 Veritabanı Performans ve Ölçeklenebilirliği

Çoğu işletmeler yazılımlarının ölçeklenebilir olması oldukça önemlidir. Yazılımların servis olarak sunulmasında ise aynı yazılımı çok sayıda işletme kullanacağından bu konu bir kat daha önem taşımaktadır. Dikey ölçeklendirme olarak daha fazla işlemci disk, hafıza gibi sistem kaynaklarının makinelere eklenmesi düşünülebilir [8]. Yatay ölçeklendirme ise daha fazla makinenin sisteme eklenmesidir [8]. Yazılımların servis olarak sunulmasında ölçeklenebilirlik açısından veri tabanı replikasyon işlemleri yapılabileceği gibi veri tabanı bölümlendirme işlemleri performans ve ölçeklenebilirlikte ön plana çıkmaktadır. Bölümlendirmeyle birlikte performans artışı olabilmesine karşın mimari açıdan birinci seviyeye bir yaklaşma olacağından yedekleme, geri yükleme ve bakım maliyetleri de artacaktır [19][8].

Yatay ve Dikey bölümlendirme veritabanında uygulanabilir. Dikey bölümlendirmede veri tabanında kullanılan tablolar ayrı ayrı bölümlendirilebilir.

Yatay bölümlendirme yazılımın servis olarak sunulmasında dikey bölümlendirmeye göre daha sık kullanılabilinmektedir [14]. Dikey bölümlendirmede etken olarak kayıt sayısının fazlalığı ve veritabanını o anda kullanan aktif kullanıcı sayısı görülebilir [8]. Bunun için yazılımın üzerinde belirli bir bölümlendirme stratejisi belirlemek uygun olacaktır. Kayıt sayısına göre bölümlendirmede kayıt sayısı sürekli değişebileceğinden bölümlendirme işlemi periyodik olarak tekrarlanabilir.

İşletme sayının yüksek olma durumunda ve saniyede yapıla işlem sayının çok yüksek olmadığı durumlarda ise işletmeye özgü şema ve ortak şema-tablo kullanımının daha

yüksek performans verdiği görülmüştür [8]. Kümelenmiş indekslemenin genellikle ufak veri boyutlarında kullanılması önerilmektedir [8].

2.6 Servis Sağlayıcının Değiştirilmesi

İşletmeler kullandıkları yazılım içerisinde farklı sürümlere geçebileceği gibi [9] yazılımlar ve ihtiyaçlar çok hızlı değiştiğinden, kullandıkları yazılımları mümkün olan en az çaba ile değiştirip farklı yazılımlara ve güncellemelere geçiş yapmak isteyebilirler. Bunun için servis sağlayıcılar işletmelere yazılım hizmetini iki yönlü seçenek şeklinde sunabilmelidir. Birinci olarak mevcut uygulamalar da verileri olan işletmelerin kendi verilerini tekrar giriş yapmak zorunda kalmadan, kendi uygulamalarıyla tümleştirebilmelidir. Aynı şekilde ileride kendi üzeriden yazılım hizmeti alan işletmelerin herhangi bir durumda farklı bir servis sağlayıcıdaki uygulamaya geçmesi veya kendi sunucularında yeni uygulama geliştirme gibi isteklerinde işletmelere kendi verilerini belirli bir standartta sunmalıdırlar.

Verilerin XML olarak saklanması veya mevcut verilerin XML şeklinde sunulması çoğu veritabanı yönetim sistemi tarafından desteklenmektedir. Verilerin XML olarak transfer edilmesi işletmeler için dahili bir internet desteği de sağlayabilir. Ayrıca Microsoft NET gibi uygulama geliştirme platformlarında gerek web gerek Windows gibi uygulamalarda bilgiler veritabanından çekildikten sonra uygulama içerisinde kullanılan hafızadaki veri kümelerinde XML olarak tutulmaktadır, bu durumda yazılımlardaki veri dönüşümleri çok daha kolay olabilmektedir. XML şemalarıyla birlikte verilerin türleri de açıklanacağı için işletmeler eşleştirmeleri kolaylıkla yapabilir.

Ayrıca bu farklı yazılımlara geçişte elde edilen XML verilerini kullanarak işletmeler mevcut uygulamalarıyla birlikte servis sağlayıcıdan kullandıkları uygulamaları birlikte çalıştırabilme şansına sahip olabilirler.

2.7 Kullanıcı Girişleri ve Yetkilendirilmesi

Günümüzde web projeleri belirli bir güvenlik seviyesine ulaşmış durumdadır. Kullanıcı hesap bilgileri SSL ve WS-Security gibi güvenlik protokolleri kullanılarak istemci ve sunucu arasında sağlanacağı için hesap bilgileri ve veri güvenliği belirli bir ölçüde sağlanmış olacaktır. Burada yazılımı geliştiren firmalar kullanıcıların sisteme girişleri için genellikle 2 farklı tipte yöntem veya bu 2 yöntemin bir arada kullanıldığı hibrit yönetimi benimsemişlerdir. Bunlar hesap bilgilerini merkezi olarak tutma ve de kullanıcıların kendi yerel sistemlerinde girdikleri bilgilerin servis sağlayıcıda da geçerli olma durumlarıdır [13][3][4].

Merkezi veri tabanında kullanıcı giriş bilgilerinin depolanması web yazılımları hazırlayan firmaların ve işlerinin bir bölümünü web ortamına taşıyan işletmelerin çok yabancı olmayacağı bir yapıdır. Kullanıcı hesap bilgileri sistemi kullanacak tüm işletmeler için tek bir veri tabanında depo edilecektir [13]. Burada farklı işletmelerin bilgileri kullanıcı giriş bilgileri aynı tablo üzerinde saklanabileceğinden güvenliği arttırıp hesap tablosunda şifre bölümüne doğrudan işletme veya kullanıcı şifresi yerine bunun şifreli değeri de saklanabilir.

Kullanıcı girişlerinde kullanılan yöntemlerden bir diğeri ise işletmelerdeki sistemi kullanacak olan kullanıcıların mevcut yerel sisteme girişleriyle servis girişlerinin eşlenmesidir [13]. Bu durum Microsoft'a göre işletme tarafında hesap yönetimi ve yazılım arasında iletişimi kuracak ek bir sunucu ile çözülebilir [13]. Bu durum hesap yönetimini ve kurulumu karmaşıklaştıracak ve maliyeti arttırabilecektir. Fakat kullanıcılar servisi sisteme tek bir girişle sanki yerel bir uygulamaymış gibi servisi kullanabileceklerdir [13] ve sisteme birden fazla kez zorunlu olarak giriş yapmaktan kurtulacaklardır [3].

Yazılımlarda kullanıcı kavramında karşımıza üç tür kullanıcı kavramı ortaya çıkmaktadır. Bunlar yazılımın yöneticileri, işletme yöneticileri ve işletmedeki kullanıcılardır [3]. Ve kullanıcılar yetkileri dâhilinde işlemleri birbiriyle karıştırmadan gerçekleştirirler [5]. Bu durumda yazılım önceden çeşitli roller tanımlamalı ve işletme yöneticileri kullanıcılarını pozisyonlarına göre gerekli rollere atayarak erişim denetimini ve yetkilendirmeyi sağlayabilmelidir [13].

2.8 Yazılımların Servis Olarak Sunulmasında Konfigürasyon

Yazılım konfigürasyon yönetimi yazılım mühendisliği konularında da ele alınan bir konudur [10] ve değişim yönetimi içindedir [18]. Yazılımların servis olarak internet üzerinden sunulması, çeşitli sektörlerden işletmelerin uygulamamızı verimli bir şekilde kullanmasını hedeflemektedir. Bu durumda her müşterinin ihtiyacı tek olma prensibi göz önüne alınarak [10] yazılımdaki kaliteyi arttıracak faktörlerden birisi de işletmelerin servis olarak sunulan yazılımı sanki kendisi için hazırlanmış bir yazılım olarak hissedip kullanmasıdır. Kısaca SaaS uygulamalarında her işletme gereksinimleri hedeflenen endüstriler, müşteri tutumları, yasal düzenlemeler, kültürel farklar ve operasyon stratejileri konularında farklılıklar gösterecektir [10]. Bu durumda işletmeler yazılımları istedikleri gibi düzenleyebilmelidir. Bu durum 2.3. bölümde bahsedilen veri tabanı düzenlemelerinin yanı sıra ekran görünümleri üzerinde oynamalar, işletme logolarının, gerekli resimlerin sisteme yüklenmesi gibi bölümler olabilir. Böylelikle işletmeler kendi alıştıkları görünüm düzenini tüm sayfa bölümleri üzerinde yapabileceği değişikliklerle düzenleyerek elde edebilir. Görsel düzenlemeden daha önemli bir durumsa kullanıcıların uygulamanın çalışmasını ve iş mantığı yapısını ihtiyaçları doğrultusunda rahatlıkla düzenleyebilmeleridir ve bunun için metadata servisleri kullanılabilmektedir [3]. Servis olarak sunulan uygulamalar ne kadar fazla özellik içerirse düzenleme ihtiyacı da işletmeler için daha fazla olacaktır [10]. Bu durumda iş akış modelleri çözüm olarak ön plana çıkmaktadır. Yazılımlarımız servis yönelimli mimaride tasarlandığından WS-BPEL protokolü bu durumda yazılımların iş akışlarıyla olan ihtiyaçları konusundaki düzenlemeleri yapabilir. Onay mekanizması veya mevcut onay mekanizmasındaki değerler üzerinde oynama yapılarak değişikler gerçekleştirilebilir [16].

Yazılımın servis olarak sunulmasında servislerin düzenlenmesinde 2 ana yaklaşım bulunmaktadır. Bunlar konfigürasyon ve özelleştirmedir [10]. Konfigürasyon seçeneklerinde işlemler belirli parametreler üzerinden metadata servisleriyle sağlanabilirken özelleştirmede ise belirli bir kod değişikliği yapılabilir [10]. Fakat kod değişimi yazılımın olgunluk seviyesini biraz düşürecektir. Yazılımındaki kod değişikliği aynı zamanda hem yazılım geliştiricilere hem de bu yazılımı kullanacak kişilere ek bir maliyet olarak geri dönecektir [10]. Bu durumda işletmeler değişim

yönetimleri politikalarını oluştururken, maliyetin yanında her işletme ihtiyaçlarının özel olmasını ve alternatif firmaların çözümlerine geçiş yapma durumlarını göz önüne almalıdırlar. Çünkü genişletilebilirlik seçenekleri belirli bir noktada sınırlı kalırsa ve de alternatif çözümler de zamanla ortaya çıkarsa mevcut müşterileri kaybetme durumu da yazılım firmaları için bir sorun olacaktır [10].

2.9 Mevcut Yazılımların İzlenmesi ve Denetlenmesi

Bir yazılımı kullanmak isteyen işletmeler yazılımın kesintisiz ve mümkün olan en az sorunla çalışmasını istemektedirler. Bu, yazılımların seçiminde bu başlıca ölçütlerden birisidir. Yazılımlar servis olarak internet üzerinden, çok sayıda kullanıcıya sunulacağı için bu durum biraz daha önem kazanmaktadır. Yazılımın o an çalışmaması veya çalışma performansının düşmesi işletmeler için ciddi zararlara yol açabilmektedir. Buradaki önemli hususlardan birisi 7x24 sorunsuz çalışabilen işletim sistemi ve yazılım geliştirme platformlarının seçimidir. Büyük web projelerinden elde edilen deneyimler bizlere gerek Net gerekse Java uygulama geliştirme platformlarıyla geliştirilen web projelerinin çalışabilirlik açısından bu performans ölçütlerini yakaladığını göstermiştir. Yazılımların servis olarak sunulmasında da uygulama geliştirme ortamı olarak bu platformların kullanılması durumunda benzer sonuçlar elde edilebilecek ve elde edilen bilgi birikimi ve deneyimler burada da uygulanabilecektir.

Performansın takibi için kurumsal ölçekli sistemlerde bulunan gerekli araçlar kullanılarak takip yapılabilir Burada bir diğer husus da sistemin de çalışmasında anormal bir durum olur ve belirli bir performans seviyesinin altına ani düşüşler gerçekleşirse bu durumun sistem yöneticileri tarafından takip edilebilir. Bu hata ve uyarı kayıtlarına gerekli mesajların işlenmesi şeklinde yapılabileceği gibi uygulamaların telefon açarak hata mesajını sesli olarak bildirmesi ve de sistem yöneticisin de gerekli gördüğü takdirde telefon tuş tonlarıyla sisteme geri bildirim sağlaması daha etkin bir çözüm olabilecektir. Bu durum tez içerisinde önerilen ve gerçekleştirilen sistemin gerekliliğini ve uygulanabilirliğini bir kez daha gösterebilmektedir.

2.10 İstemciler

Yazılımın servis olarak sunulmasındaki önemli konulardan bir tanesi de istemcilerin etkin kullanımıdır. Yazılımlar internet üzerinden kullanıcılara sunulacağı ve sunucu taraflı çalışan uygulamalar olacağı için akla ilk gelen erişim şekli web tarayıcılarıdır. İşletmedeki kullanıcılar pratik olarak hiçbir işleme ve yatırıma gerek kalmadan mevcut web tarayıcılarıyla sistemi kullanabilirler.

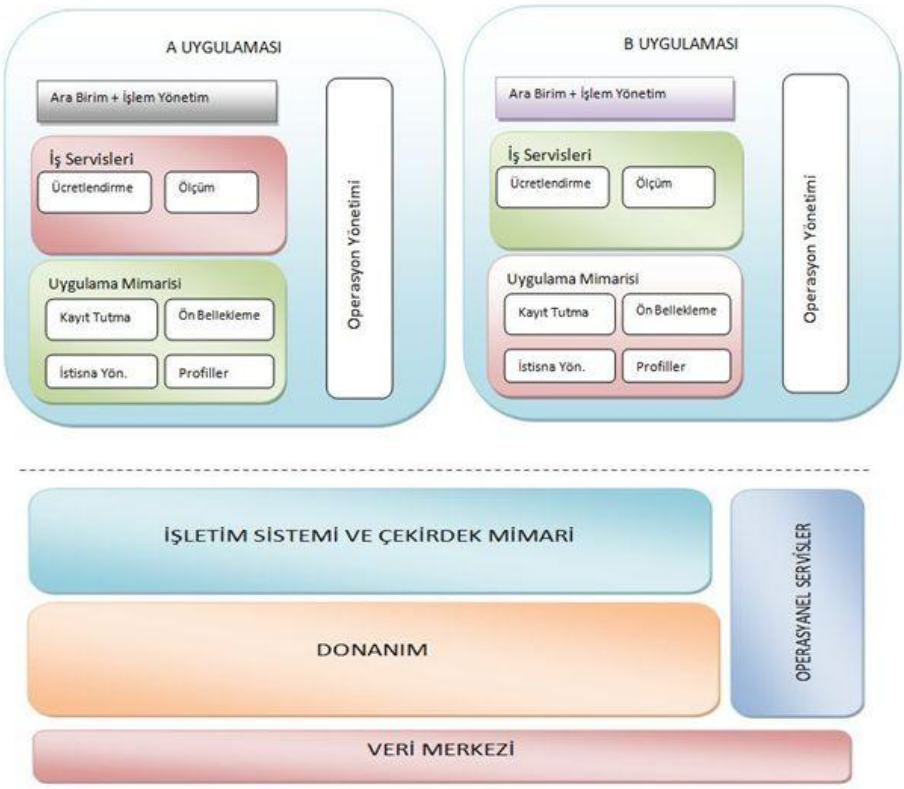
Web tarayıcıları uygulamaları son yıllarda Web 2.0 uygulamalarının, AJAX benzeri teknolojilerin gelişmesiyle daha zengin Windows uygulamalarına benzer şekilde çalışabilecek bir yapıya dönüşmüştür. AJAX'ın istemci tarafında çalışması ve sunduğu ara yüzün zenginliği kullanıcıların uygulamayı Windows uygulamalarının rahatlığıyla kullanabilmelerini sağlayacaktır.

Web tarayıcı istemcilerinde AJAX gibi teknolojilerin yanı sıra Silverlight, Flash gibi teknolojiler de kullanılabilir. Bu sayede görünüm daha kullanışlı hale gelebilir. İstemcilerde hafif ve vekil kod oluşturulmadan servis bağlantıları için REST yapısı kullanılabilmektedir. REST yapısı 3.5 numaralı bölümde incelenecektir.

2.12 Yayınlama

Servis parklarının gündeme geldiği [11] ve de Microsoft firmasının da Windows işletim sistemini internet üzerinden ayrı bir sürümle [54] sunmaya başladığı bir süreçte servislerin yayınlanması gerçekten önemli bir konu olarak gündeme gelmektedir. Servis olarak sunulan uygulamayı gerçekleştiren ve yayınlayan firmalar aynı olmak zorunda değildir [17]. Uygulamaları yayınlama hizmeti veren yayıncı firmalar temek paylaşımlı servisler sunabilirse platformlarında daha çok servisi barındırabilir ve servis uygulaması geliştiren yazılım firmalarının kodlama işlemlerini en aza indirebilirler. Çeşitli firmalar SaaS yapısını bir basamak daha öteye taşıyıp platformları servis olarak sunmaya başlamışlardır [12][17]. Uygulama geliştiriciler bu platform üzerinde uygulamalarını servis sağlayıcının ortaya koyacağı arabirimlerle geliştirip yayınlama imkanlarına sahip olabilmektedir ve Salesforce.com, Google App Engine bunlara örnek olarak gösterilebilir [12].

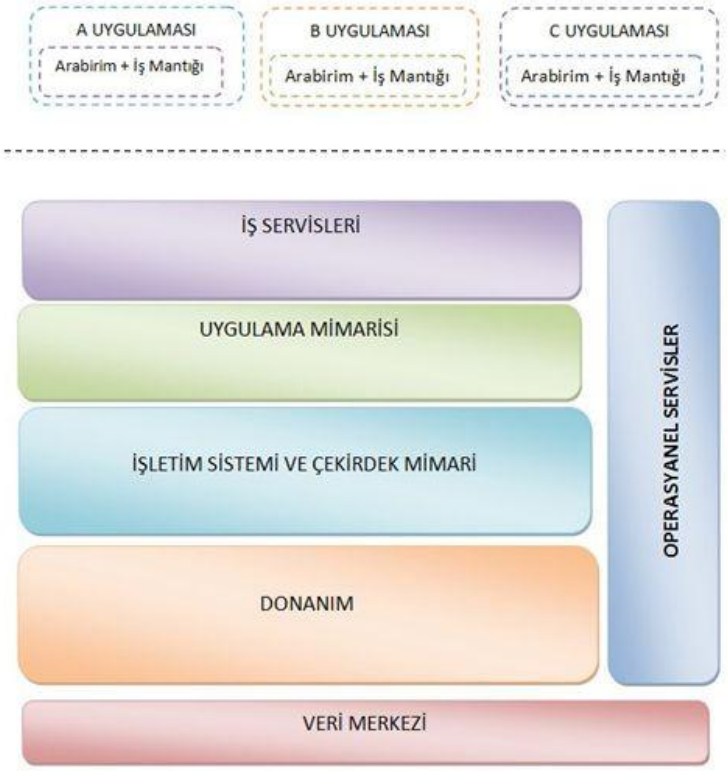
Burada Şekil 2.8 de görüleceği gibi belirli bir platform dışında bazı hizmetler servis olarak uygulamayı geliştiren firmadan farklı olarak servis sağlayıcı tarafından geliştirilmiş olabilir. Bu yazılan servisler üzerine, servis olarak yazılımlar geliştirilerek hem zamandan hem de platformun yeteneklerinden daha fazla yaralanılabilir.



Şekil 2.8 SaaS Hizmet Dağıtım Platformu [17]

Bu şekilden de görülebileceği gibi uygulama geliştiren firmalar iyi bildiği işle yani sadece uygulamalarını geliştirerek yayına hazır hale getirebilir. Bunun yanında firmalar uygulamalarının yayınlanması ile ilgili teknik işlemleri bu konuda uzmanlaşmış başka firmalara vererek onlar da donanım, bakım ve operasyon maliyetlerinden kurtulabilmektedirler. Buradaki önemli noktalardan birisi şekil üzerinden de görülebileceği gibi SaaS uygulamaları ne kadar farklı olurlarsa olsunlar benzer bir mimariye sahiptir. Gerek aynı uygulama geliştirici firmalar gerekse farklı firmalar benzer kod bileşenlerini değişik uygulamalar için tekrar tekrar yazmak

zorunda kalabilmektedirler. Şekil 2.8'de görülebileceği gibi iş servisleri ve uygulama mimarisi katmanları iki farklı uygulama için geçerli olabilecek ve benzer işlemleri yerine getirebilecektir. Bu durumda Şekil 2.9'da görüldüğü gibi sistem tasarımıyla SaaS uygulamalarında ortak olabilecek servisler bir platform altına toplanarak SaaS uygulama geliştiricileri için ortak bir platform olarak sağlanabilmektedir.



Şekil 2.9 Gelişmiş SaaS Hizmet Dağıtım Platformu [17]

Hizmetlerin yayınlandıktan sonra nasıl bir abonelik modeliyle sunulacağı da SaaS yazılımını tasarlayan firmaların göz önünde bulundurması gereken noktalardan birisidir. Bu modeller aylık abonelik, kullanılan kadar ödeme, sistem üzerinde gerçekleştirilen işlem kadar ödeme, seçilen özellik kadar ödeme olabilir [66]. Bu modelleri etkileyecek ortak noktalar reklam ve işletmelerin alacağı özel destek paketleridir [67].

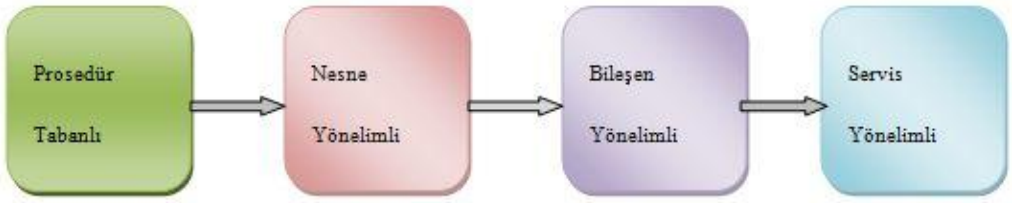
3. SERVİS YÖNELİMLİ MİMARİ (SOA)

Servis Yönelimli Mimari mevcut yazılım geliştirim modellerinin sonucunda ortaya çıkmış bir yaklaşımdır. Tamamıyla sıfırdan geliştirilen bir yapıdan çok önceki modellerinin eksiklikleri ile iyi yönleri göz önüne alınarak ve önceki modellerde eksik olan internet ayağının tamamlanmasıyla ortaya çıkmıştır. Web servisleri standartlarını belirleyen en önemli kurumlardan birisi olan OASIS'e göre, "Servis Yönelimli Mimari farklı alanların sahipliğinde ve kontrolünde bulunan dağıtık yeteneklerin kullanımı ve organizasyonu üzerine bir paradigmadır" [22][20]. Bu tanımlamanın en başta yapılmasının bir nedeni de Servis Yönelimli Mimari üzerine farklı kurumların ve kişilerin Servis yönelimli mimariyi farklı şekilde algılayarak farklı tanımlar yapabilmesi ve bu kavramın en çok karıştırılan konulardan biri olabilmesidir [21]. Bu tanım tam olarak net bir açıklama yapmadan genel bir çerçeve çizdiği için biz de Servis Yönelimli Mimariyi bu tanım çerçevesinde inceliyoruz.

Üçüncü bölümde ilk olarak servis yönelimli mimarinin gelişim sürecinden, hangi noktaları hedeflediğinden ve genel mimarisinden bahsedilecektir. İkinci maddede servis yönelimli mimari karakteristikleri ele alınacaktır. Servis yönelimli mimariye uygun servisler geliştirmek için servislerimizin hangi özellikleri barındırması gerektiği ve bu özelliklerin genel yapısı işlenecektir. Üçüncü maddede web servisinde kullanılan protokoller ve yapılar ele alınmıştır. Bu protokolleri etkin bir şekilde kullanmak servis yönelimli mimariden elde edecek faydaları arttıracaktır. Dördüncü maddede servislerin akış yolu olarak düşünülebilen Kurumsal Servis Veri Yolu (ESB) incelenmiştir. Ve son bölümde ise web servisi protokollerini, HTTP üzerinden daha farklı ele alan REST konusu incelenmiştir.

3.1 Servis Yönelimli Mimari ve Gelişim Süreci

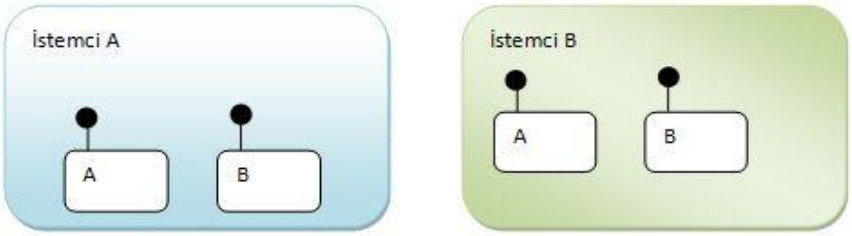
Servis yönelimli mimari belirli bir gelişimin sonucunda ortaya çıkmış olan bir yaklaşımdır. Internet'in hızlı gelişimi, iş ve çözüm platform oluşu da bu süreçte oldukça etkili olmuştur. Servis Yönelim öncesinde yazılım geliştirim modellerini ve süreçlerini incelersek Şekil 3.1'e benzer bir şekil görebiliriz.



Şekil 3.1 Yazılım Geliştirim Modellerinin Gelişim Süreci

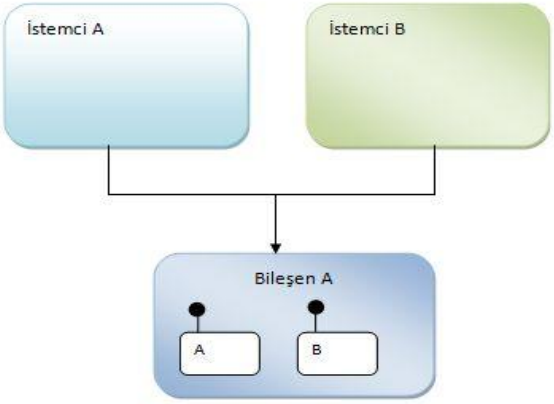
Yakın geçmişten günümüze kadar olan programlama modellerini incelediğimizde Yordamsal(Procedural), Nesne Yönelimli(Object Oriented), ve bileşen tabanlı programlama modellerini görmekteyiz. Her bir modelde öğrenilen kazanımlar ve tecrübeler, sonraki modellerin tasarımında etkin bir rol oynamıştır [20]. Günümüzde ise bileşen tabanlı modeller yerini servis yönelimli modellere bırakmaktadır. Servis yönelim, nesnesel programlama modeline bir alternatif bir yaklaşım değildir. Tam tersine servis yönelimli mimari içerisinde nesnesel programlama da kullanılmaktadır. Yazılım dünyasının ilk safhalarından beri (C ve Pascal programlama) programcıların aklında kodların tekrar kullanımı mevcuttu. Fakat bu durum bir fonksiyondan, prosedürden veya kodun kopyalanıp yapıştırılmasından öteye geçememiştir. Nesnelerimiz, görsel programlama dillerinde çeşitli bileşen ve kütüphanelere dönüşmesiyle bu sayede yazılan kodların işlemler arası kullanılması sağlanmıştır. Nesne yönelimli programlamada daha büyük kod blokları, sınıf dediğimiz yapılar içinde toplanmıştır. Sınıflar verileri ve programın iş mantığını içerebilmektedir. Dezavantajı ise sınıfların farklı teknolojiler arasında kullanılabilir durumda olmamasıdır. Ör: C++ sınıflarının Java içerisinden kullanılması. Şekil 3.2'den de görülebileceği gibi kodlar sınıflar içerisinde toparlanmaktadır. Kodun tekrar

yazılmasının önüne geçilmiş olunmuştur fakat bu durum kodun kopyalanmasını gerektirmektedir [20].



Şekil 3.2 Kodların Sınıflar İçerisinde Toplanması[20]

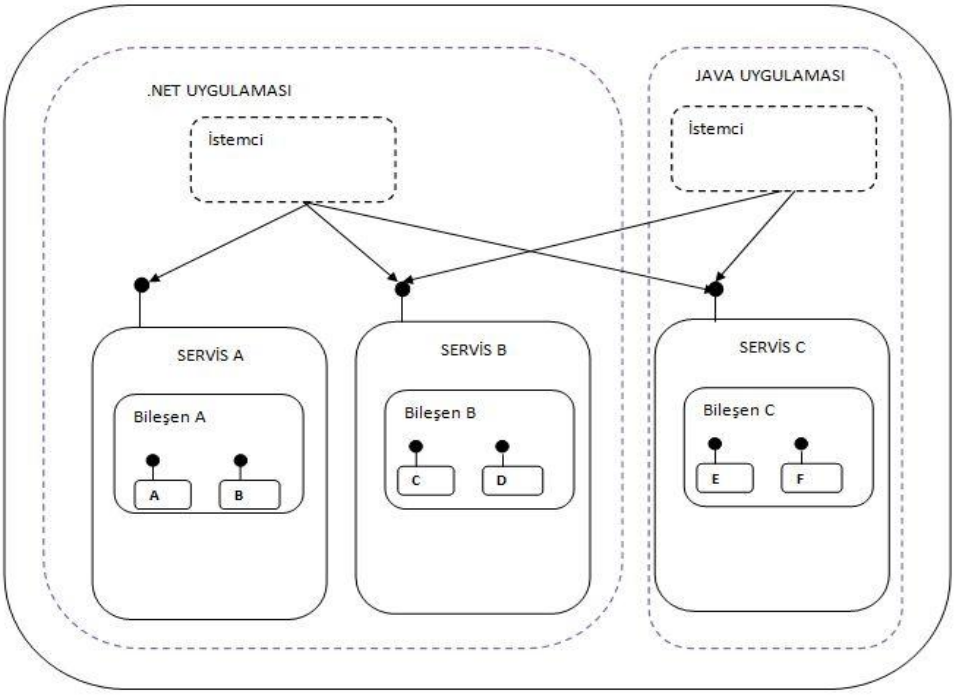
Bileşenler sadece bilgisayar üzerindeki işlemlerle birlikte çalışmayabilir. Bu bileşenlerin farklı bilgisayar üzerindeki işlemlerde çalışmasıyla birlikte dağıtık programlama kullanılmıştır. Corba, Com, Dcom, Com+ gibi teknolojilerin gelişmesiyle farklı kodlar arasında Şekil 3.3'deki gibi ortak nesnelere erişim sağlanmıştır.



Şekil 3.3 Bileşen Tabanlı Yapı (Ortak Nesneler) [20]

Bu nesneler yerel olarak bilgisayarlarda da bulunmayabilir. Fakat bu yapıların web ve internet desteğinin oldukça sınırlı olması farklı teknolojiler ve platformlar arasındaki sınıf erişimini sınırlandırmıştır. Ör: Unix ortamındaki Java uygulaması ile Windows ortamındaki c# uygulaması. Internet ve buna bağlı protokollerin gelişmesi

dağıtık programlamaya farklı boyutlar katmıştır. Böylelikle dağıtık programlamayı uygulama ve platform bağımsız şekle sokabilmiştir. Bu da genellikle temel web servisleriyle oluşmuştur. Şekil 3.4 üzerinde bir gösterim görülmektedir.



Şekil 3.4 Farklı Platformdaki Bileşenlere Web Servisleriyle Erişim [20]

Bu web servislerinin kullanımı genellikle eski bileşen tabanlı dağıtık programlamanın bir web ara yüzü olarak görülmüş ve bu da web servislerinden beklenen verimi çok arttıramamıştır. Tüm bu gelişmeler sonucunda servis yönelimli model bir çözüm noktası olarak belirmiştir.

Servis yönelimli mimari prensipleri 3.2. bölümde açıklanacaktır. Fakat bunları incelemeden önce Microsoft firmasının bu prensipleri 4 ana başlık altında topladığı yapıyı inceleyelim:

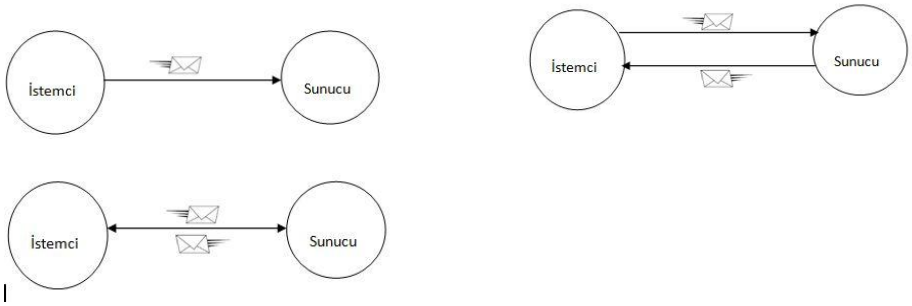
Birinci madde servis sınırlarının belirgin olmasıdır. Bu sayede servisler kontratlar yoluyla dış dünyaya yapabileceklerini anlatırlar. Ne başka servislerin kendi kod yapısına izin verirler ne de başka servislerin kodlarına doğrudan müdahale ederler [23]. Her istemci veya başka servis, servisin çalıştırdığı kod ve platform hakkında; kontrat bilgisinin dışında ekstra bilgiye sahip olmak zorunda değildir [23][24].

İkinci özellik servislerin otonom bir yapıya sahip olmasıdır. Her ne kadar arka planda başka sistemleri, iş bileşenlerini ve veri tabanlarını kullansalar bile başka yere taşındıklarında veya yer değiştirdiklerinde çalışmalarına bu yeni durumdan etkilenmeden devam edebilmelidirler [23]. Arka planda çalışacak olan servis kodlarındaki değişim servis için yeni bir sürüm olmamalıdır. Arka planda ve servisin üzerinde çalıştığı ortamda oluşacak hatalar servisi ve servise bağlı olan sistemleri etkilememelidir [23][24]. Servis yönelimi mimaride otonomi konusu 3.2.5 numaralı bölümde detaylı olarak incelenecektir.

Üçüncü olarak servis ve istemciler kod yerine kontratları paylaşırlar [23]. Kontratların sürümleri ve uyumlulukları bu yüzden oldukça önemlidir. Bununla ilgili detaylı inceleme 3.2.1 numaralı bölümdedir.

Son olarak dördüncü madde de servislerin uyumluluğu bazı anlaşmalar üzerine kurulması bulunmaktadır [23]. Burada kontratların, servisin yeteneklerini belirtmesinin yanı sıra servis ve istemci arasındaki iletişimin hangi yöntemlerle ve nasıl yapılacağını belirleyen anlaşmalara da gereksinim vardır. Bu sayede servis ve istemci istenilen kanallardan, istenilen standartlarda ve istenilen şekilde iletişim kurabilirler [20][23].

Servis yönelimli mimari içerisinde karşımıza temelde üç çeşit mesajlaşma örüntüsü çıkmaktadır. Bunlar çeşitli kaynaklarda farklı isimler alabilmelerine karşın temelde yapıları aynıdır [24]. Bunlar temel tek yönlü iletişim, karmaşık-çift yönlü iletişim ve istek ve yanıt yapılarıdır. Ayrıca servislerinde çeşitli koşullarda istemcilere uyarı ve bilgilendirme mesajları göndermesi de bu yapı içerisinde düşünülebilir. Şekil olarak bakarsak Şekil 3.5'deki gibi görülebilir.



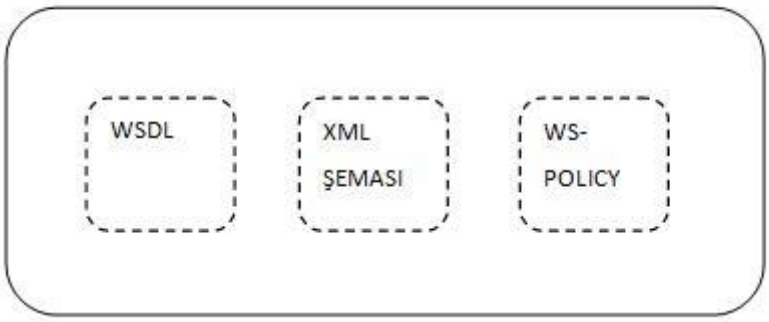
Şekil 3.5 Servislerde Kullanılan Mesajlaşma Örüntüleri [24]

3.2 Servis Yönelimli Mimari Tasarım Prensipleri:

Servis Yönelimli Mimari prensipleri, Servis yönelimin tanımı gibi farklılıklar içermektedir. Fakat bu farklılıklar olmasına rağmen temelde içerik olarak birbirine yakındır. Çoğu durumda aynı konular farklı isimlerle farklı konu başlıkları altında bulunabilmektedir. Biz bu noktada servis yönelimin maddelerini 8 madde altında incelemekteyiz. Aynı içerik farklı prensip tanımlarında ve farklı prensip maddelerinde de görülebilmektedir.

3.2.1 Servis Kontratları:

Servisler özelliklerini ve yeteneklerini dış dünyayla kontratlar vasıtasıyla paylaşırlar [26][20]. Bu yüzden bu anlaşmalar sayesinde iletişim kurulacağından servisler ve bunu kullanacak olan istemciler açısından çok önemli bir roldedir ve standartlara dayalı olmalıdır. Servis kontratları yapısı içerisinde 3.3.1. bölümde bahsedilecek olan web servisleri tanımlama dili WSDL, servis içersindeki verileri ve yapıları tanımlayan XML şema doküman ve anlaşmaları bulunduran WS-Policy dokümanı bir web servisinin kontratını oluşturabilmektedir.



Şekil 3.6 Servis Kontratı Yapısı

Günümüzde Şekil 3.6'da görülen kontrat yapısı WS-MetadataExchange protokolü içerisinde oluşturulabilmektedir. Bu sayede tek bir referans sayesinde kontratlar tanımlanabilmektedir. Kontratlar yazılım dünyası için web servisleriyle gündeme gelmiş yeni bir kavram değildir. Çeşitli istemci-sunucu yapılarında önceden yapılan tanımlamalarla birlikte günümüzdeki WSDL dokümanlarının karşılığı olabilecek

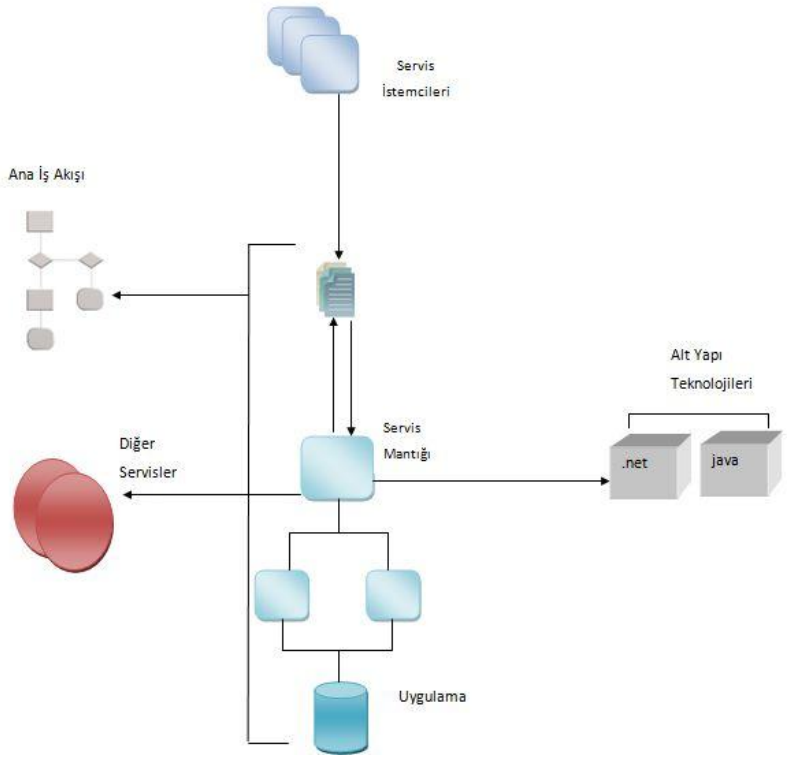
IDL dili mevcuttu. Projelerde karşılaşılan durumlardan birisi de kontratların sürümlerini yönetmedir.

3.2.2 Servislerin Gevşek Bağlanması:

Teknolojik gelişmelerin nasıl bir yol izleyeceğini tahmin etmek ve planlamak kolay bir durum değildir. Genellikle bu değişimlere bağlı olan değişimler bilgi işlem sistemlerine bir dış etki olacaktır [21]. Servis Yönelimli mimari gelişim sürecinde bahsedildiği gibi kodların belirli bir blok yapılarında tutulması ve bunların birbiriyle olan ilişkileri yazılım tasarımında etkili bir yaklaşımdır. Burada beklenmedik teknolojik gelişmeler ekseninde bu değişimlere verilecek hızlı cevaplar servis yönelimli mimarinin ana hedefleri arasındadır. Burada servisler ve bileşenler arasında çevik hareket sağlanması servislerin birbirleriyle gevşek bağlaşımı sayesinde gerçekleştirebilir [21][26].

Burada servisler birbirlerini önceden belirlenmiş kontratlar ve parametreler sayesinde tanımakta ve iletişim kurabilmektedir. Bu sayede servisler birbirlerini iyi tanımakta fakat bağımsızlık korunarak birbirlerinden etkilenmeleri en aza indirgenmektedir [20][21][26]. İç mimarideki bileşenlerin sıkı bağlaşımı servis kontratları sayesinde servise gevşek bağlaşım özelliği kazandırabilir. Bununla birlikte Şekil 3.7'den görülebileceği gibi servis kodları başka ana iş akışlarına da gevşek bağlaşım sayesinde kolaylıkla bağlanabilmektedir. Burada servis mantığının servis içerisindeki bağlaşımları görülmektedir.

Servis istemci programları servis kontratlarıyla bağlaşım içerisine girerek bir anlamda servis içeriğinde bağlaşım ve servisin bağımlılıklarını da miras almış olurlar. Bu bağımlılıklar servisin kullandığı uygulama geliştirme arabirimleri, süreçler dış servisler ve veritabanları da olabilir [26]. Bağımlılıklar kontratlar üzerinden sağlanacağından burada kod değişikliği yapmaya gerek kalmadan uygulanabilinir.



Şekil 3.7 Servislerin Gevşek Bağlaşımı Gösterimi [26]

3.2.3 Servis Soyutlanması:

Servis Yönelimli mimariyi geçmişten gelen teknolojik gelişimlerin bir sonucu olarak ele aldığımızda ve soyutlama kavramı gerek nesne yönelim gerekse de bileşen tabanlı mimaride sıklıkla kullanıldığından bu yaklaşım servis yönelimli mimarinin de temel yapıtaşlarından birisini oluşturmaktadır. Böylelikle servisler detayları saklanmış bir kapalı kutu görevi görebilirler [21]. Servislerin kapalı kutu olarak detayları dış dünyadan soyutlamaları servis kodlarında gerçekleşecek olan güncellemeleri, teknoloji bağımlılıklarını [26] ve çeşitli değişimleri de istemcilerden saklayarak çalışmalarına devam etmelerini sağlayacaktır. Gevşek bağlaşımın bir gerekliliği olarak ve servis kullanıcılarının servis koduyla ilgili güvenlik açısından daha az detayın ve gereksiz bilgilerin dışarıya verilmeyip servislerin daha sade bir hale gelmesi ve bağımlılıklar soyutlama açısından önemlidir [26]. Ayrıca bir servisin soyutlama ve erişim düzeyleri sabit olmayıp her istem için farklılık gösterebilir [26].

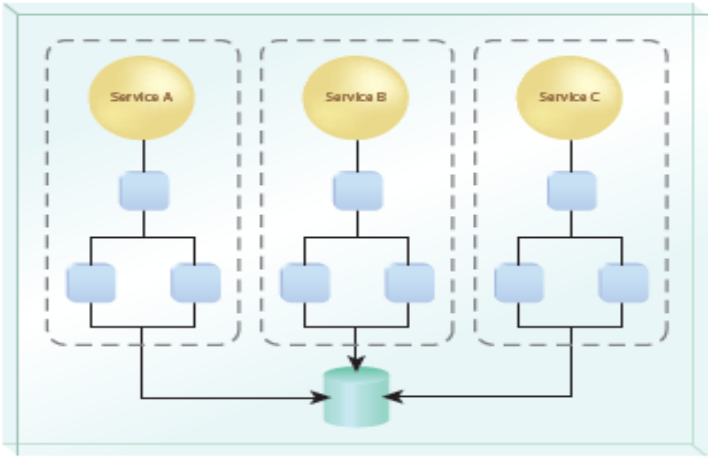
3.2.4 Servislerin Tekrar Kullanılabilirliği:

Servis yönelimi bölümünün başında bahsedildiği gibi tekrar kullanılabilirlik yazılım geliştiriciler için önemli noktalardan biridir. Yazılım geliştirim süreçlerinin ve yaklaşımının gelişmesi sonucunda servis yönelimli mimaride tekrar kullanılabilirlik önemini ve yeteneğini biraz daha arttırmıştır. Tekrar kullanılabilirliğin internet ortamında servis yönelimli mimarilerde kullanılabilmesi kod tekrarını en aza indirmiş, maliyetleri düşürmede yardımcı olmuştur [26]. Servis kodu üzerinde yapılacak olan değişimler, bu servisi kullanan istemcilerin kodlarını bozmadan merkezi olarak yapılabilmesine olanak tanır. Burada servislerin yeni sürümlerinin istemcilerde bulunan kontratla oluşturulmuş kodlarla uyumu göz önünde bulundurulması gerekir. Bu yapıda servis kodlarının eş zamanlı olarak kullanımı söz konusudur ve bu tasarım yaklaşımı web servislerinin bütününde uygulanabilmektedir [26]. Bu yaklaşımda servislerin eş zamanlı kullanımı arttıkça servislerin başka yazılımlarındaki kullanımı servislerin ölçeklenebilir olmasını da gerektirebilecektir [21][26]. Servislerin tekrar kullanılabilirliğini etkileyen faktörler: Servisin özel bir işlem birimine bağımlı olmaması ve birden fazla otomasyonda kullanılabilir olmasıdır [21][26].

3.2.5 Servislerin Otonomisi:

Bir servisin otonom olması kısaca servisin çalışma zamanını yüksek seviyede kontrol edebilmesi olarak tanımlanabilir [26]. Böylelikle kendi sorumluluk alanı artacağından yükleme ve çalışma durumunda başka çalışma zamanı hizmetlerine ve programlarına olan bağımlılık [21] en aza indirilebilir. Bağımlılığın azalması ile servisler daha rahat taşınabilen ve hareket edebilen parçalar haline dönüşebilmektedir. Servis içerisinde oluşabilecek hatalar da, bağımlıkların en az düzeyde olması nedeniyle sistemin çalışmasını an az düzeyde etkileyecektir [20][25]. Servislerin otonom yapıya sahip olmasıyla beraber servislerin güvenilirliği ve işlemlerin önceden tahmin edilebilirliği de artacaktır [21][26]. Çünkü işlemlerin çalışmasından büyük ölçüde servisin kendisi sorumlu olabilecektir. Şekil 3.8'deki

servisler belirli bir otonom yapı içerisinde bulunmakta olup gene de veritabanını ve çalışma platformunu diğer servislerle ortaklaşa paylaşmaktadırlar.



Şekil 3.8 Servislerin Otonom Olarak Farklı Etki Alanlarında Çalışabilmesi [26]

Otonom yapıda her bir servis ayrı bir çalışma zamanında, farklı veritabanlarını da kullanarak tasarlanabilmekte ve bu sayede bağımlılıklarını en üst düzey de tutabilir. Fakat birinci bölümde bahsedilen uygulamaların servis olarak sunulmasındaki seviyelere benzer güçlükler burada da karşımıza çıkabilecektir.

3.2.6 Servislerin Durumları:

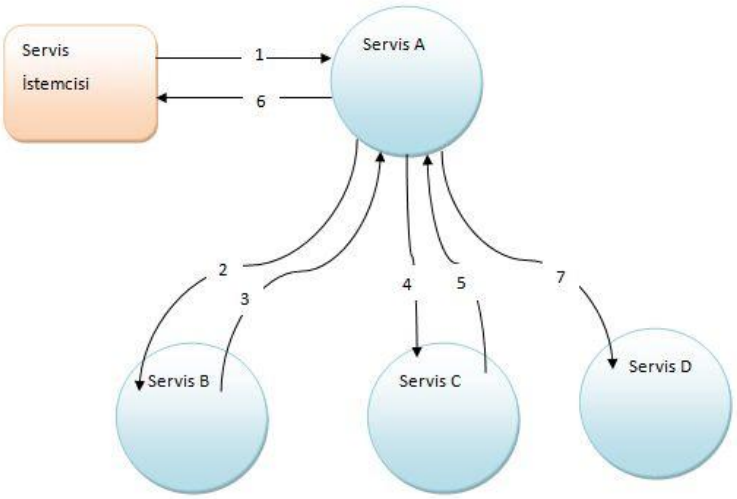
Web uygulamalarında durum bilgisi her zaman göz önünde bulundurulması gereken durumlardan birisidir. Web Servisleri servis yönelimli uygulamaların en önemli uygulama şekillerinden biri olduğu için de bu durumdan oldukça etkilenebilmektedir. Burada servisler çalışma zamanında kullanacakları durum bilgisi miktarlarını ve bunların kullanım sürelerini en alt düzeyde tutmalıdırlar [21]. Bu sayede sistemdeki ölçeklenebilirlik ve çalışma zamanı kullanılabilirliği artacaktır [26]. Servis yönelimli mimaride durum bilgisinin en alt düzeyde olmasının temelinde doküman stili mesajlaşma bulunmaktadır. Bu da mesajlara başlık bilgilerinde barındırılan daha fazla yetenek sayesinde kazandırılabilinir. Uygulamamızda durum bilgisini saklamaya zorunlu olduğumuz yerlerde durum bilgisi saklanabilmelidir.

3.2.7 Servislerin Keşfedilebilirliği:

Oluşturulan servisleri başka uygulamaların ve servislerin kullanımına sunmak servis yönelimin temelini oluşturan maddelerden birisidir. Servislerin keşfedilebilir olması ihtiyaç fazlası servisin veya kodunda gereksiz işlemler barındıran servislerin kullanımını en alt düzeyde tutmamızı sağlayacaktır [21]. Servisler kendileri hakkında kolaylıkla keşfedilebilir ve yorumlanmalarına yardımcı olacak bilgileri içermelidir [26]. Bu bilgiler genellikle kontratlar yoluyla yapılabilmektedir. Servislerin keşfedilir olması web servislerinin ilk günlerinden beri gündemde olmasına karşın bunu sağlamak için tasarlanan UDDI yapısı etkin bir kullanım alanı bulamamıştır. Büyük işletmeler için servislerin tasarımda, belirli bir servis havuzunda bilgilerin toplandıktan sonra bu servise ihtiyaçları doğrultusunda yeniden yazılmadan bulunup kullanılması şeklinde düşünülebilir. Bu araştırma servis kontratlarının projeye eklenmesi öncesinde düşünülüp kontratların eklenmesiyle tamamlanmaktadır [26]. Çalışma zamanın da kontratlar 3.3.4 bölümünde anlatılacak olan WS-MetadataExchange protokolü ile gerçekleşebilmektedir. Bir servisin keşfedilebilir olmasındaki bazı ölçütler düzgün bir anlatıma sahipliği, işlevselliğinin kontratlarda düzgün bir şekilde anlatılma durumu, servis kayıtlarının yeterli bilgiyi barındırması, standartlara olan uygunluğu ve denetleyiciler tarafından incelenmesi olarak düşünülebilmektedir [26]. Buradaki problemlerden biri de servislerin keşfedilip kullanıldıkları zamanda önceden belirttikleri servis seviyesi anlaşmalarına olan uygunluklarıdır. Servis seviyesi anlaşmalarına uygunluğu kullanıma başlanıldığında tekrar gözden geçirilmesi gereken noktalardan birisidir [26].

3.2.8 Servis Kompozisyonları

Bir kurumsal uygulamanın tek bir servis üzerine kurulu olması servis yönelimli mimarinin kendi prensiplerine de uygun olmayacaktır. Bu yüzden bir uygulama birden fazla servis üzerine kurulmuş olabilir. Bu noktada servislerin birbirleriyle net ve düzgün bir biçimde çalışması beklenmektedir. Bu noktada büyük bir problemin ufak servis çözümlerine bölünerek tekrardan büyük sonuca ulaşması servis kompozisyonlarının hedeflerinden biridir [26].



Şekil 3.9 Kompozisyonları Oluşturarak İşlem Yapılması [26]

Şekil 3.9'de servislerin örnek bir kompozisyonu görülmektedir. Orkestrasyon motorlarıyla beraber birden fazla servisin çalışması istenilen düzende çalışması sağlanabilmektedir.

3.3 Servis Yönelimli Mimaride Kullanılan Temel Protokoller:

Web Servislerinin ilk yaygınlaşmaya başladığı süreçte karşımıza SOAP, WSDL ve UDDI gibi kavramlar geldi. Bu kavramlar web servislerinin temelini oluşturdu. Web servislerinden elde edilebilecek yetenekler ve potansiyel çok daha fazla olduğu için temelde bu protokolleri de kullanan WS-* protokolleri dediğimiz protokoller geliştirildi. Bu protokoller genellikle W3C ve OASIS gibi standartlar bünyesinde geliştirildi ve kabul gördü. Bu bölümde temel protokollerle beraber bazı temel WS-* protokollerini incelemekteyiz.

3.3.1 Temel Web Servisi Protokolleri (XML-SOAP-WSDL)

Giriş bölümünde bahsedildiği gibi XML internet ortamında görüntü ile verinin birbirinden ayrılmasını sağlayarak bilgisayarların internet üzerinden veriyle iletişimini gerçekleştirmede önemli basamak olmuştur. XML (Extensible Markup Language), HTML gibi SGML tabanından gelen bir dil ve metin tabanlı esnek bir

dil olduğu için mevcut HTML ve HTTP protokolleri üzerinden çalışan sistemlere uyumu çok kolay olmuştur. Bu durum XML'in uyum sürecini hızlandırmıştır. XML esnek bir dil olduğu için farklı kullanıcılar kendileri için gerekli yapıları düzenleyebilmektedirler. Burada bir uygulama içi ve uygulamalar arası veri değişimini belirli bir yapıda standartlaştıracak olan kavram ise XML şemalarıdır. Şemalar XML dosyalarından ayrı olarak oluşturulup kullanılırlar. Şemalar sayesinde bir uygulama veya kurum için hazırlanan XML dosyasının diğer tüm kullanıcıların bu yapıya uyumlu XML dokümanları oluşturmalarına olanak tanır. Bu sayede esnekliğin beraberinde getirebileceği uyumsuzluklar ortadan kalkabilecektir.

```
<ogrenciler>
        <ogrenci no=”100”>
                <isim>A</isim>
                <bolum>B</bolum>
        </ogrenci>
        <ogrenci no=”110”>
                <isim>C</isim>
                <bolum>d</bolum>
        </ogrenci>
</ogrenciler>
```

Şekil 3.10 Örnek Xml Belgesi

XML veri türlerinin diğer programlama platformlarında standart olarak tanımlanabilmesi ve tanımlanan bu verilerin tiplerinin şemalar içerisinde belirtilmesi ve şemalardaki bu veri tiplerinin birçok programlama platformu için de mevcut veri tipleriyle eşlemesi sonucunda XML'in yazılım geliştirme platformları içerisindeki uyum sürecini oldukça hızlandırmıştır.

```
<xs: element name="ogrenciler">
  <xs: complexType>
    <xs: sequence>
      <xs: element name="isim" type="xs: string"/>
      <xs: element name="bolum" type="xs: string"/>
    </xs: sequence>
  </xs: complexType>
</xs: element>
```

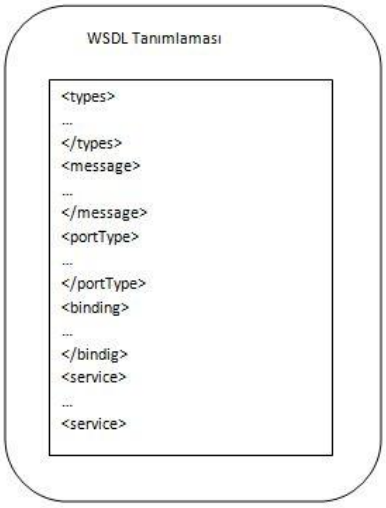
Şekil 3.11 Örnek XML Şeması

Web servisleri ve servisleri temeli ve geçmişini oluşturan bir diğer kavram da uzak yordam (RPC) çağrılarıdır. Web servislerinde bu işlemler XML tabanlı bir protokol olan SOAP (Simple Object Access Procotocol) ile yapılmaktadır. SOAP protokolü sayesinde bir webservisi üzerinde bulunan metotlar çağrılabilmekte ve işlem sonucunda gelen yanıt gene SOAP yapısı içerisinde olmaktadır. Kısaca servis istemci arasındaki bilgi alışverişi SOAP üzerinden yapılmaktadır.

SOAP'ın mesajını oluşturan temel yapıya SOAP zarfı denilmektedir ve başlık ve gövde olmak üzere iki bölümden oluşmaktadır. SOAP tüm WS-* protokolleri için temel mesajlaşma yapısıdır. Başlık bilgisi içerisinde bu protokoller bulunabilmektedir. SOAP hem metotların çağrılmasında hem de gelen sonuçların alınmasında gövde kısmını kullanmaktadır.

Şekil 3.12 SOAP Mesaj Yapısı Gösterimi

WSDL (Web Services Description Language) dili web servislerinin özelliklerini, metotlarını, metotların aldığı ve sonuç olarak döndürdüğü parametreleri ve türlerini belirtmekte kullanılmaktadır. Web servislerinde istemci tarafında oluşturulan vekil sınıf WSDL dokümanlarına bakılarak platform tarafından oluşturulabilmekte ve buradan vekil sınıf SOAP çağrılarını kendisi yönetebilmektedir.



Şekil 3.13 WSDL Doküman Tanımlaması Gösterimi

WSDL ayrıca temel veri tiplerinden başka karmaşık veri tiplerini ve yapılarını da XML şemaları içerisinde barındırarak bunların SOAP üzeriden iletimi ile ilgili bilgileri taşır.

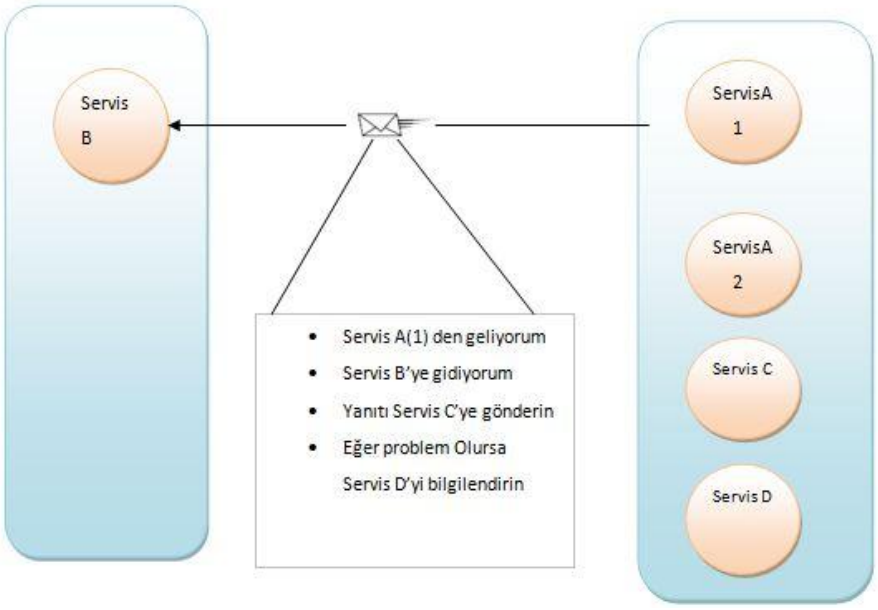
3.3.2 Ws-Addressing

WS-Adressing protokolü [56] adından da anlaşılacağı gibi web servislerinin adreslenmesi ile ilgilidir. Fakat kullanım alanı oldukça geniş olup çoğu WS-* protokolü ile birlikte kullanılmaktadır. Bu sayede beraberinde kullanılan protokolden alınacak verimi daha üst noktalara taşıyabilmektedir. WS-Addressing protokolü SOAP yapısının başlık kısmında bulunur. Bu sayede fazladan mesaj paket trafiği yaratmamakta ve o mesaj hakkında daha detaylı bilgiler verebilmektedir. Kısaca SOAP zarfına kazandıracağı yetenekler kısaca bakılırsa. SOAP mesajının nereden geldiği bilgisini, hangi adrese ve bu adreste gitmesi gerektiğini, mesajın yanıtlanmasının nereye yapılacağını ve de hata durumunda uyarıların nereye gönderileceği gibi bilgileri WS-Addressing SOAP zarf yapısının başlık bilgisinde saklayarak mesajın özelliklerini ve yeteneklerini arttırabilmektedir [21].

Ws-Adressing protokol yapısında iki temel kavram vardır. Bunlar Bitiş Noktası Referansları ve Mesaj Bilgi Başlıklarıdır. Bitiş noktası referansları bir bakıma WSDL protokol yapısındaki bazı eksik noktaları kapatabilmektedir. Bunlardan en önemlisi eğer servisin birden fazla instance'ı olma durumunda hangisinin üzerinden geldiği ve hangi servise gideceği gibi bilgileri bitiş noktası referansları tutabilmektedir ve gerekebilecek parametre değerleri de taşınabilmektedir. Bitiş Noktası Referansları adres, referans özellikleri, referans parametreleri, port özellikleri, WS-Policy bölümlerinden oluşmaktadır. Temel mesajlaşma örüntülerini geliştiren kısım ise Mesaj Bilgi Başlıklarıdır[21][20].

WS-Addressing protokolü tek başına kullanıldığı gibi WS-ReliableMessaging, WS-Policy, WS-MetadataExchange, WS-Eventing ve WS-Security protokolleriyle beraber kullanıldıklarında hem kullanıldıklarını protokollerin yeteneklerini hem de kendi yeteneklerini arttırabilirler.

Şekil 3.14'de WS-Addressing yapısındaki bir mesajın gösterimi vardır.



Şekil 3.14 WS-Addressing Mesaj İşleyiş Yapısı [21]

3.3.3 Ws-Policy:

Giriş bölümünde web servislerinin kullanım amaçlarından biri de sistemlerinin birbirleriyle insan faktörü olmadan doğrudan bir otomasyon içinde çalışması olarak bahsedilmişti. Otomasyon sistemi belirli kurallar ve kısıtlamalar içinde çalışmayı gerektirebilmektedir. Bu kısıtlamalar verinin türü, iş seviyesi anlaşmaları ve güvenlik düzeyleri olarak görülebilir [21]. Servisler ayrıca kendi karakteristiklerini belirli bir kuralda sunabilmeli ve çalıştıkları sistemin karakteristikleri hakkında detaylı bilgi sahibi olarak karşı sistemin tercihleri, davranışları ve yeterlilikleri hakkında belirli bir bilgiyi standart yapı üzerinden erişmeli, sunabilmelidir. Bu yapı WS-Policy protokolü [57] içerisinde tanımlanmıştır ve WSDL doküman tanımlamalarını daha da geliştirmiştir [21][25].

Bir WS-Policy dokümanında tanımlayıcı olarak birden fazla tanımlama dokümanı WS-PolicyAttachments ve WS-PolicyAssertions ile bulunabilmekte; bu sayede servislerin kuralları ile ilgili gerekli esneklik sağlanabilmektedir.

WS-Policy protokolünün elamanlarını ve yapısını incelersek kök etiket içerisinde Policy etiketi bulunduğunu, ExactlyOne, All, Preference gibi etiketler mevcut kurallara uyum ile bilgiler verdiğini görebilmekteyiz [21].

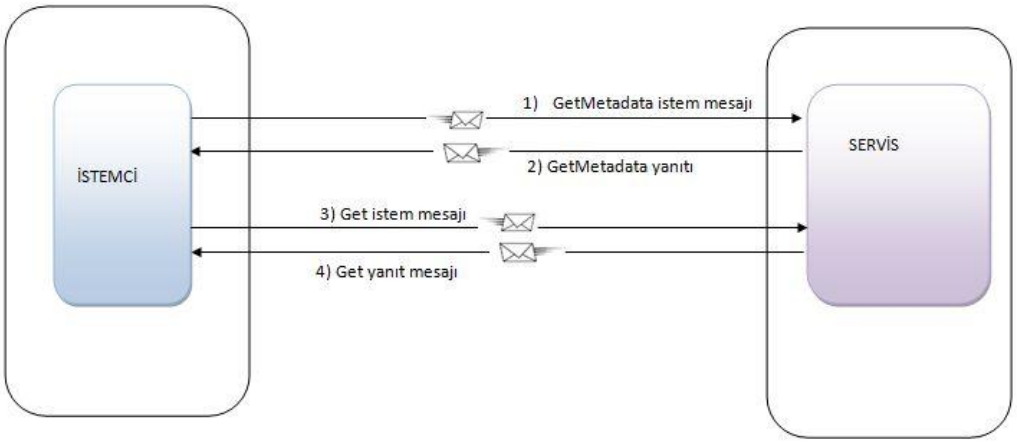
Ws-Policy protokolü servis yönelimli mimari ve bağlı protokollerle beraber sıklıkla kullanılabilmektedir. Kendi yapısı içersinde Ws-Addressing kullanabilmekte ve Ws-MetadataExchange protokol yapısında servis tanımlamasının bir bölümünü oluşturmakta ve Ws-ReliableMessaging protokolü için iletim ile ilgili uyulması gereken kuralları belirtmektedir. Tüm bu protokollerden başka Ws-Security içerisinde sıklıkla kullanılmakta ve güvenlik için uyulması gerekenleri istemcilere belirtir.

3.3.4 MetadaExchange

3.2.1 numaralı bölümde servis yönelimli mimarinin temel özelliklerinden servis kontratlarını incelemiştik. Bu bölümde incelendiği gibi servis kontratları ileri düzey servis yönelimli uygulamalarda WSDL dokümanları kontratlar üzerindeki detaylı bilgileri taşımakta yetersiz kalabilmekteydi. Servis kontratlarının içereceği tüm bilgileri taşıyabilmesi için tasarlanan protokol WS-MetadataExchange protokolüdür [55]. İçerisinde XSD şema yapıları WSDL ve WS-Policy dokümanlarını barındırabilir ve yeni eklenebilecek olan doküman tanımlama tiplerini taşıyabilecek esnekliktedir. Servisler hakkında bilgi veriminde MetadataExchange protokolü tanımlamaların dinamik olarak çalışma zamanında da tanımlanmasını mümkün kılmaktadır. Bu sayede çalışma zamanında programın çalışması kesilmeden programatik olarak kontrat güncellemeleri yapılabilmektedir. Fakat MetadataExchange dokümanları mevcut servislerin keşfedilmesi için gerekli olan yapıları barındırmaz [21].

Şekil 3.15'de MetadataExchange haberleşmesinin örnek gösterimi görülmektedir. Burada birinci basamakta servise GetMetadata istem mesajı gönderilerek servisten tüm verebileceği tanımlama bilgileri istenmekte ve ikinci basamakta görüldüğü gibi bu yanıt gönderilmektedir. Fakat bu yanıt içerisinde bazı dokümanların bilgileri yerine adresi bulunduğundan, üçüncü basamakta eğer bu doküman istemcinin

gerçekleştireceği işlemler için gerekliyse Get mesajı ile özel olarak istenmekte ve dördüncü basamaktaki yanıtla istemciye yanıt döndürülmektedir.



Şekil 3.15 WS-MetaDataExchange Protokol Yapısı [21]

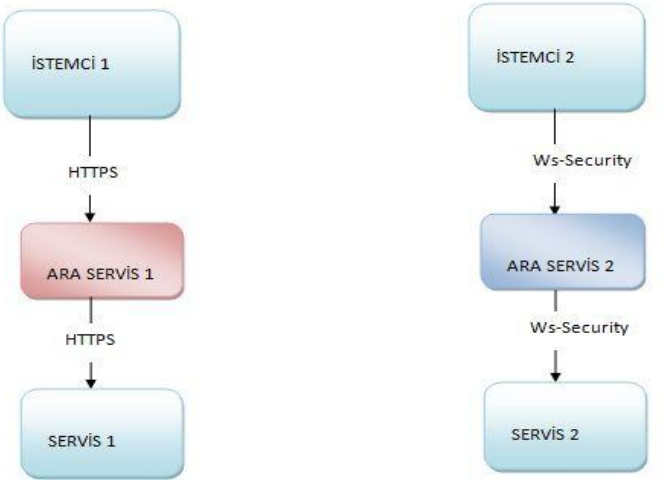
3.3.5 Ws-Security

Servis Yönelimli mimari içerisinde de servislerin güvenliği en önemli noktalardan biridir. Kullanıcı girişleri, giriş yapan kullanıcıların hangi yetkilere sahip olacakları ve servis üzerinde gerçekleştirecekleri işlemlerin kapsamlarını belirlemek servis yönelimli mimaride güvenliğin temel noktalarından birini oluşturmaktadır. Kullanıcı girişlerinden farklı olarak web servislerin güvenliğindeki diğer etmenler; verilerin gizliliği yani internet ortamında taşınırken başkaları tarafından içeriğinin görülmemesi ve bilgilerin herhangi bir değişikliğe uğramadan istenilen hedefe ulaştırılmasıdır. Tüm bu işlemler ve daha fazlası için geliştirilen güvenlik protokolü WS-Security'dir [58].

Bu protokol kendi başına bir yapı olmaktan çok güvenlikle ilintili diğer protokol ve yapıları taşıyan bir çatı görevini de görmektedir. XML-Signature, XML-Encryption, WS-SecurityPolicy, WS-Trust, SAML bunlardan bazılarıdır. XML-Signature verilerin bütünlüğü ve verilerin içeriklerinin değişip değişmediği ile ilgilenirken, XML-Encryption verilerin şifreli bir şekilde iletimi işlemlerini gerçekleştirmektedir

ve SAML protokolü 2.7. Kullanıcı Girişleri ve Yetkilendirmesi bölümünde de bahsedilen kullanıcıların tek bir hesap ve tek bir kullanıcı girişi ile farklı servislere de erişimini sağlayabilen protokol yapısını oluşturmaktadır.

WS-Addresing protokolü Servis yönelimli mimarinin en önemli protokollerinden birisi olduğuna değinilmişti. Gelişmiş bir servis yönelimli mimaride servisler ve istemcilerin tek bir katmandan haberleşmesi yerine birden fazla servis ve katman üzerinden haberleşmesi karşılaşılan durumlardandır. SSL protokolü iletim katmanında güvenliği sağlamaktadır ve noktadan noktaya verilerin güvenli bir biçiminde iletiminden sorumludur. Fakat Şekil 3.16'dan da görüleceği gibi ara katmanlara ulaştığında servis ve istemci katmanında bazı saklı kalması gereken bilgiler ara katman tarafından görülebilmektedir. Ara katmanlar iletim seviyesinde noktadan noktaya güvelik konusunda problem oluşturabilmektedir. WS-Security protokolünün temelinde SOAP mesajının gövde kısmının tümü veya mesaj içerisinde belirli bölümler şifrelenebilmektedir. Başlık kısımlarında ara katmanın yönlendirmeyi gerçekleştirmesi için gerekli bilgileri barındırarak istemci servis arasındaki haberleşmeyi sağlamasını mümkün kılmaktadır. Şekil 3.16'de İstemci2 ile Servis2 arasında bu durumu açıklamaktadır.



Şekil 3.16 Ara Katmanlarda Güvenlik Erişim Yapısı

Ws-Security protokol yapısını incelediğimizde ise başlık kısmında Security ana etiketinin altında kullanıcı adı ve şifrelerinin saklanmasını için UserNameToken bölümü bulunmaktadır. BinarySecurityToken bölümü içerisinde ise sertifika bilgileri bulunmaktadır. Gövde kısmında şifrelenmiş olan veriler EncryptedData etiketi altında bulunmaktadır ve bu sayede tüm gövde bölümünün şifrelenmesi yerine gövde kısmında istenilen verilerin ve bölümlerin şifrelenmesi gerçekleştirilebilir. EncryptedData şifreleme işlemleri için CipherData ve CipherValue etiketleri bulunmaktadır [21]**.** Ws-Securtiy protokolü diğer Ws-* protokolleriyle de birlikte kullanıldığında da gelişmiş servis yönelimli mimarilerin ortaya çıkmasını sağlayabilmektedir.

3.3.6 MTOM

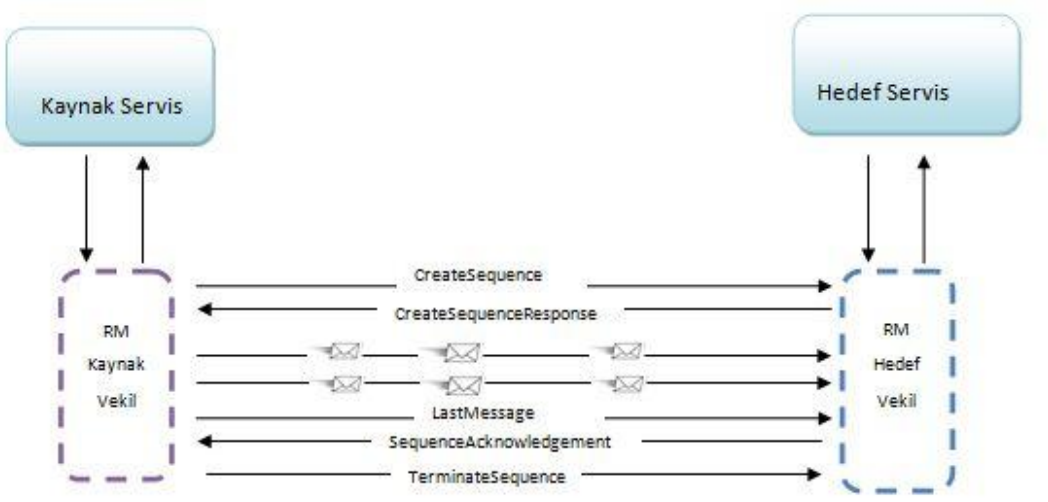
Web servisleri içerisinde genellikle metin tabanlı içeriğin SOAP gövdesi içerisinde taşınmasıyla haberleşmeler yapılmaktadır. Fakat temel veri tiplerinin yanında dosyalarında web servisleri mesajlarında taşınması da servis yönelimli mimarilerde karşılaşılabilecek durumlardan biridir. İkili verilerin bu temel yapı içerisinde taşınması ikili verilerin base64 olarak kodlanması nedeniyle mesajların yapısına ek bir yük getirecektir**.** Ayrıca mesaj boyutu da büyüyeceği için XML düzenleyicilerinin bu büyük XML mesajlarını düzenlemeleri de güçleşecektir. MTOM (Message Transmission Optimization Mechanism) bu sorunların üstesinden gelmek için hazırlanmış bir protokoldür [20][21]**.**

Bu protokol temel olarak büyük mesajların XOP (Xml-binary Optimized Packaging) ile birlikte SOAP gövdesi içerisinde, mesajların ayrı bir MIME (Multipurpose Internet Mail Extensions) olarak gönderilmesinin özelliklerini belirtir. Böylelikle XML dosyasının boyutu azalacak ve ikili veriler base64 olarak kodlanmayıp bu kodlamanın getireceği ek yükler olmayacaktır.

3.3.7 WS-ReliableMessaging

Web servisleri iletişimlerinde gönderdikleri mesajların takibini genellikle HTTP protokolü üzerinden iletişim kurdukları için yapamazlar. Bu takipten kasıt mesajın hedefe başarılı bir şekilde ulaşıp ulaşamadığı, mesajların iletimde problemlerle karşılaşılıp bazı mesajların yeniden iletilip iletilmeyeceği ve mesajların geliş düzenleridir [21]. Yukarıdaki maddeler ışığında bu işlemleri gerçekleştirmek için tasarlanan yapı Ws-ReliableMessaging protokolüdür. Bu mesaj yapısı da diğer çoğu WS-* protokolleri gibi bu düzenleme için SOAP mesajının başlık bilgilerini kullanmaktadır [20][28].

Bu protokolü kullanan servisler vekil bir sınıf ve tampon alan oluşturup gerekli düzenlemeleri yapmakta ve istenen sonuçları doğru şekilde ve sırada servislere iletilmektedir. Şekil 3.17 üzerinde gerekli mesaj sıralamasını gösteren yapı görülmektedir.



Şekil 3.17 Ws-ReliableMessaging Örnek Haberleşme Düzeni [21]

3.3.8 Ws-AtomicTransicton (WS-AT)

Hareketler (Transactions) kurumsal uygulamalarda uzun zamandır süregelen yapılardır. Hareketler bir dizi işlemlerin yap hep ya hiç mantığına göre çalışması prensibine dayanır. Kurumsal yazılım geliştirme platformları ve veritabanları bu

konuda gelişmiş bileşenlerin geliştirilmesi için uygun bir yapı sunmaktadır. Servis yönelimli mimariyi bileşen tabanlı mimarinin bir sonraki ve gelişmiş bir basamağı olarak gördüğümüz için temel hareket özelliklerinin servis yönelimli mimaride web servisleriyle kullanımını bu protokol sağlamaktadır. Temel hareket özellikleri dört tane olup terminolojide kısaca ACID (Atomic, Consistent, Isolated, Durable) olarak ifade edilmektedir. Atomic özelliği ya hep ya hiç mantığını tanımlarken, Consistent verilerin işlemler bitmeden kullanıcılara yanlış ve yarım sonuçlar gösterilmesini engelleme özelliğidir. Isolated sistem üzerinde birden fazla hareket gerçekleşmesi durumunda bunların birbirlerinin çalışmasını engellemeden birbirinden izole bir şekilde çalışmasıdır. Durable ise herhangi bir nedenden dolayı hareket içindeki bir işlemin tamamlanamaması veya yarım kalması durumunda hareketin istenilen işlemleri sonradan gerçekleştirebilmesidir [21][20].

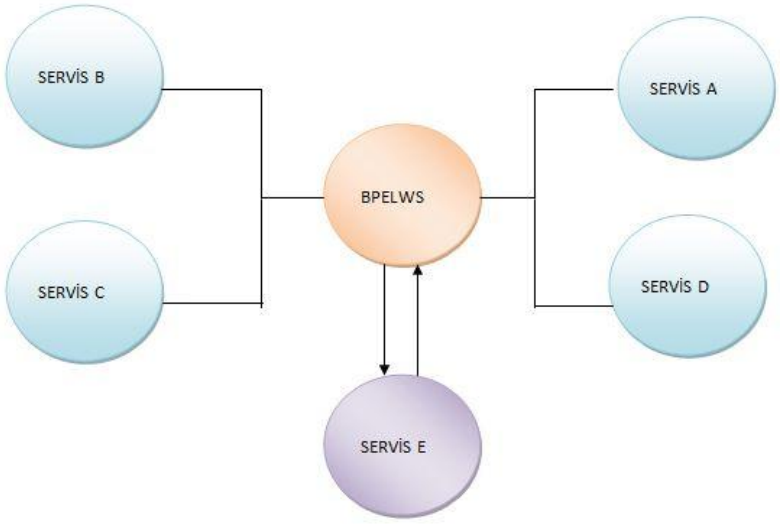
WS-AT protokolü temelinde WS-Coordination protokolünü kullanarak bir üst koordinatörü kullanarak işlemlerini gerçekleştirmektedir [21]. Bu durumdan da anlaşılacağı gibi hareketlerde bir hareket yöneticisi tüm hareketleri yönetmekle sorumludur. Birden fazla WS-AT hareketi gerekli olduğu durumlarda WS-BussinessActivity protokolü kullanılarak bunların toplu bir iş hareketi ve aktiviteleri mümkün olabilmektedir [21].

Belirli bir koordinatörün altında toplanan işlemler, koordinatör tarafından gönderilen bir hazırlık mesajıyla durumlarını servise bildirir. Servisler belirtilen işlemi yapıp yapamadıklarını belirten bir mesajı koordinatöre yanıt olarak iletir. Sonrasında koordinatör servislerden gelen bu yanıtları toplayarak her hangi bir olumsuz yanıtta tüm servislere işlemi iptal etmeleri için gerekli mesajı veya hepsi olumluysa işlemi gerçekleştirmeleri için onay mesajını gönderir [21].

3.3.9 BPELWS

Servis yönelimli mimaride servislerin otonom yapıda kendileri işlem yapabildiğine değinilmişti. Fakat bir grup servisin belirli bir yapıda farklı olduğu ve servislerin birbirlerini çok iyi tanımadığı bir durumda çalışma söz konusu da olabilir. Bu durum

genellikle mevcut servisleri kullanarak yeni bir uygulama yazmadan süreçlerin yönetilmesi ile ilgilidir. Bu durum bileşen yönelimli mimaride kurumsal yazılım entegrasyonu kavramıyla karşımıza gelmekteydi ve bileşenler belirli bir süreç içerisinde gerçekleştirilerek ortada bu bileşenler arası koordinasyonu sağlayan bir yazılım yönetim ya da diğer bir değişle orkestra eden bir uygulama mevcuttur [21]. Bu kavram da servis yönelimli mimaride servislerin belirli standartlar üzerinde farklı bir yapıda ve farklı platformlar üzerinde olacağı için arada bu koordinasyonu sağlayacak olan mimarinin de belirli standartlar üzerine kurulacak olması servis yönelimli mimariye oldukça yüksek kazanımlar sağlayacaktır. Ve bu bölümde değineceğimiz BPELWS bunu hedeflemektedir.



Şekil 3.18 Servis BPEL ile Yönetilen Servis Çalışma Gösterimi [26]

BPEL'de koordinasyon önemli bir yer tutmaktadır. Servislerin bu koordinasyona katılmalarıyla birlikte BPEL işlem servisi üzerinden birbirleriyle iletişim kurmaktadır. Bu yapıda BPEL içerisinde koordinasyon ile ilgili temel iş mantığı kurallarını da içerebilecektir.

BPEL dilinde işlemler belirli bir iş sırasında birbirinin sonuçlarını bekleyerek ilerleyebilir buna sequence denilmektedir ya da birinin sonucuna bağlı olmaksızın belirli bir akış içerisinde ilerleyebilmektedir buna da flow denmektedir. BPEL dilinin

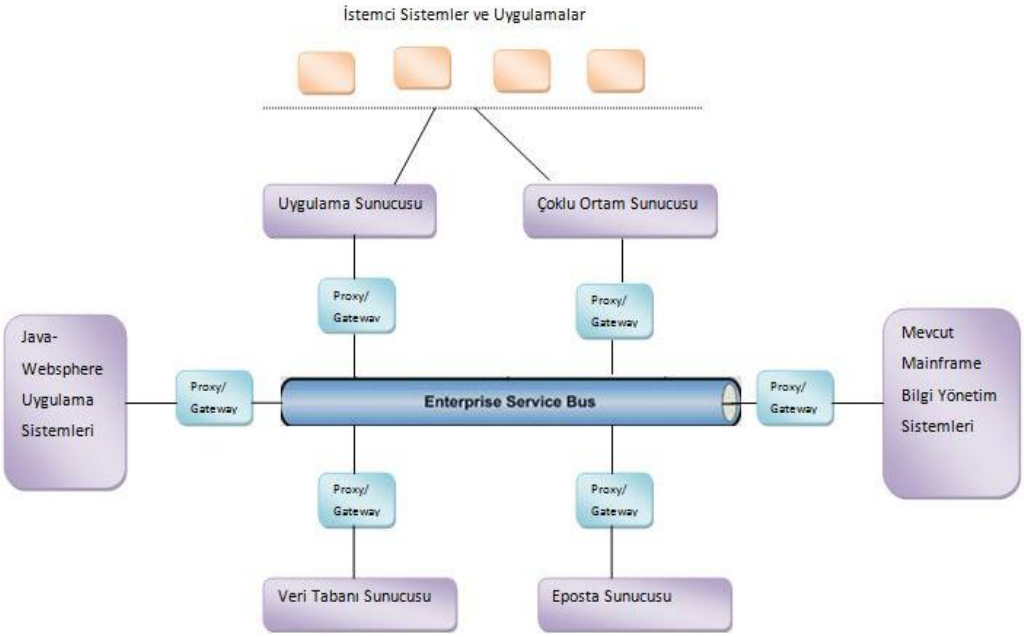
temelini bu 2 yapı oluşturmaktadır. BPEL dilinde tüm yapılar process etiketi içerisinde oluşturulmaktadır. Process içerisinde bulunan partnerlink elamanları ile bpel işlemlerine katılacak olan servisler adresleri ve işlemleri variables ile servislerin işlem sonuçlarından gelen değerler saklanmaktadır [21]. Invoke belirli bir servis üzerindeki işlemleri çağırma, belirli bir işlemleri istekleri kabul etme ve reply da gerekli yanıt verme durumları içindir. Programlama dillerinden bildiğimi Swicth..Case yapısı burada swict, case ve otherwise olarak karşımıza çıkmaktadır [21]. Değerleri işlem içerisinde belirli yerlere atayabilmek için assign, copy, from ve to mevcut olup hata durumlarında yapılacak olanları belirleyebilmek için faulthandlers ve catch yapısı vardır [21].

```
<sequence>
<receive name="…"
partnerLink="…"
portType="..."
operation="…"
variable="…"
createInstance="…"/>
<invoke name="..."
partnerLink="…"
portType="…"
operation="…"
inputVariable="…"
outputVariable="…"/>
<reply partnerLink="…"
portType="…"
operation="…"
variable="…"/>
</sequence>
```

Şekil 3.19 Örnek BPEL Kod Yapısı [21]

3.4 Enterprise Service Bus (ESB)

Çeşitli kaynaklarda kurumsal hizmet veri yolu olarak da adlandırılan bu kavram servis yönelimli mimarideki servislerin ve kurumsal iş bileşenlerinin geniş ölçekli bir yapıda sayısının artması ve bunların birbirleriyle çalışmasını hedefler. Ayrıca yeni servislerin eklenmesi veya eskilerinin bu yapıdan çıkarılmasıyla problemsiz çalışmasını hedeflemektedir. Bu yapıda her servis kendini yapabileceklerini ve yeteneklerini bu kanala sunar. Bu kanala dahil olan diğer sistemler bu servislere bu kanal üzerinden erişim sağlarlar. Şekil 3.20'den de görülebileceği gibi gerek çalışma alt yapısı gerek çalışma mantığı farklı olan çok sayıda sistem birbirleriyle ortak bir kanal aracılığıyla anlaşabilmekte ve entegrasyon problemlerini en alt düzeyde tutabilmektedir. Şekilde tüm servis ve sistemler ESB'ye belirli bir vekil üzerinden erişmektedir. Bunlar adapter olarak da bilinmekte ve sistemlerin mevcut diğer sistemlerle kod üzerinde değişiklik yapmadan ve karşı sistem veya servis için bir özelleştirme yapmadan bağlantı görevini de üstlenebilir. Bu sayede birbirlerini tanımayan sistemler gerekli dönüştürücü ve veri yolu sayesinde, ek bir kod yüküne ihtiyaç duymadan entegrasyonu sağlamakta ve uygulamalar karşılıklı konuşabilir bir yapıda çalışabilmektedirler.



Şekil 3.20 Kurumsal Veriyolu Uygulaması Gösterimi [29]

ESB temel olarak işletmelerde oldukça verimli sonuçlar üretebilmesine karşın, ESB kullanan veya kullanacak olan çeşitli işletmeler yönetimsel ve eğitimsel maliyetlerle karşı karşıya kalacaklardır. Bulut bilişiminin yazılımların servis olarak sunulmasının bir türü olarak bahsetmiştik. Bu durumda kurumsal veri hizmet yolu yapısının da internet üzerinden servis olarak sunan kurumlardan temin edilmesi, donanımsal, yönetimsel, personel eğitimi ve başlangıç maliyeti gibi kavramlarda işletmelere ve hatta yazılımı servis olarak sunan işletmelere de çeşitli kazanımlar sağlayabilmektedir. Tezin öneriler kısmında değindiğimiz Microsoft tarafından gerçekleştirilen ve hizmete sunulan .Net Service Bus uygulaması da bu kavrama bir örnektir [30].

3.5 REST (Representational State Transfer)

Proje içerisinde kullanacağımız servis yönelimli mimarideki istemci modellerinden birisi de REST modelidir. Bu kavram uzun zamandan mevcut olmasına rağmen Web 2.0 ile mevcut HTTP bilgilerinin yeniden keşfi ile kendisine geniş kullanım alanı bulmaya başlamıştır [65]. Amazon gibi firmaların web servislerinde REST'i kullanmaktadır.

HTTP protokolünün temel noktalarından birisi de URI kavramıdır. URI bir bakıma bize web ortamındaki kaynaklara erişmemizi sağlayan konumlandırıcıdır ve URL kavramının bir üst basamağını oluşturmaktadır. Burada sunucuyla haberleşme URI kavramı üzerinden gerçekleşeceği için URI'lere daha fazla anlam vererek ve gerekli HTTP metotlarını kullanarak sunucuya istediğimiz verileri ve bu verilerle nelerin yapılabileceğini anlatmamız mümkün olabilecektir. Bunun için de sunucu üzerinde isteğe uygun bir URI adresi oluşturulmalı ve sunucudaki işlemlerin bu URI hedeflenerek gerçekleşmesi sağlanmalıdır. Bu da bir bakıma web servisi üzerindeki metotlar URI adresleri üzerinden erişime açıktır.

URI'ler üzerinden işlemler yapılacağı, mesajlaşmalar ve istekler bu adres üzerinden kullanılacağı için REST modelinde sunucu üzerinde servislerle iletişimde kullanılmak üzere bir karmaşık bir vekil sınıfa ihtiyaç duyulmamaktadır. Bu sayede istemciler çok daha hafif bir yapıya sahip olabilirler.Verilerin iletişiminde ise HTTP üzerindeki MIME yapısını kullanarak göndermektedir ve sunucunun gönderdiği

değerleri XML formatında alabilmekte, basit bir javascript koduyla bile istenilen işlemler geçekleştirilebilir. Bu yapı doğrudan adresler üzerinden geçekleşen işlemler olduğundan aynı URI adresini birden fazla istemciler için web sunucusu tarafında belirli bir önbellekleme yapmak mümkündür. Burada tüm işlemlerin adreslere taşınması bunların kullanıcılar için belirli bir anlamsal düzende de olmasını gerektirebilmektedir.

- http://webservisleri.com/sesservisi/islemlistesi/kullanici1
- http://webservisleri.com/sesservisi/telefonac

Şekil 3.21 Örnek Gerçekleştirilebilecek URI'ler

Şekil 3.21' de ki REST servisi örnek URI'leri görülmektedir. Durum bilgisi servisler için önemli olup durum bilgisi URI adresinde taşınarak servise iletilmektedir. Burada URI adresinde kullanci1 değeri taşınarak işlemlerin kullanici1 için gerçekleşmesi sağlanacak ve ayrım bu şekilde sağlanabilecektir.

İstekleri istemci ve servis arasında gerçekleştirecek olan yapı HTTP metotlarıdır. Dört temel HTTP metodu olmasına karşın bazı eklentiler sayesinde bu metot sayısı sekize çıkabilmektedir. Bazı istemler varsayılan olarak sadece Post ve Get metotlarını desteklemekte ve eklentiler gerektirmektedir. Bu temel dört metot veritabanlarında sıkça kullanılan işlemlerle ilişkilendirilip servislerin HTTP üzerinden gelen metoda göre veritabanında veya kendi iç işleminde benzer işlemleri gerçekleştirmesi istenmektedir. Bu veritabanı bağlantılı olabilecek işlemler İngilizce terminolojide kısaca CRUD (Create, Read, Update, Delete) olarak adlandırılmaktadır ve Çizelge 3.2'de mevcut http metotlarıyla olan eşleşmeleri verilmiştir [31][32].

Örneğin http://webservisleri.com/sesservisleri/kayitlar/kullanici1 şeklinde bir URI'si olabilecek olan REST servis inde GET metodu kullanıldığında kullanıcı1 ile ilgili kayıtlar getirilebilinirken DELETE ile kullanıldığında ise kullanıcı1'e ait olan kayılar için silme işlemi gerçekleşebilecektir.

Çizelge 3.1 HTTP metotlarının, Veritabanları İşlemleriyle Eşleştirilmesi

HTTP	CRUD
POST	CREATE
GET	READ
PUT	UPDATE
DELETE	DELETE

SOAP kavramı XML ve web servisleri kavramının ilk günlerinden beri süre gelmiş bir haberleşme şeklidir ve servis yönelimli mimarinin ana bileşenini oluşturmaktadır. WS-* protokolleri her gün daha çok olgunlaşmakta ve kendine servis yönelimli mimaride daha çok yer bulmaktadır. REST kavramı, SOAP protokolünün yerine geçecek bir yapıdan daha çok mevcut HTTP ile servis ve işlemci yükünün daha aza indirilebilecek uygulamalarda karşımıza daha çok çıkabilir. Günümüzde web servislerinin önemli yapıtaşlarından biri olan WSDL kavramının REST'e özel tam karşılığı kullanıcılar arasında farklı yaklaşımlarla neden olabilmektedir [31]. WSDL 2.0 içerisinde belirli bir REST desteği mevcut ve ayrıca farklı bir yapı olarak WADL (Web Application Description Language) kavramı da servis tanımlamaları için mevcut olmakla beraber fakat bu eğilimler günümüzde SOAP ve WSDL tümleşmesinden uzaktadır [31].

4. SES HİZMETLERİNDE KULLANILAN TEKNOLOJİLER

Bu bölümde sistem gerçekleştirimi esnasında ses hizmetlerinde kullandığımız teknolojilere değinilmektedir. Öncelikli olarak VoIP olarak adlandırdığımız ve Internet üzerinden ses iletimini temelini oluşturan SIP protokolü üzerine açıklamalar yapılmıştır. SIP protokolü ses iletiminin başlaması için öncesinde gerçekleşecek tüm anlaşmaları ve sonlandırmaları gerçekleştirmektedir. SIP protokolündeki, anlaşmalar sonrasında ses iletimi başlayacak ve bölüm içerisinde açıklayacağımız RTP protokolü üzerinden gerçekleşecektir. Sonrasından tezin öneriler bölümünde değineceğimiz ses sentezleme işlemini yönlendiren SSML ve konuşma diyalogları oluşturmamıza olanak tanıyan VoiceXML konuları hakkında bilgi verilecektir.

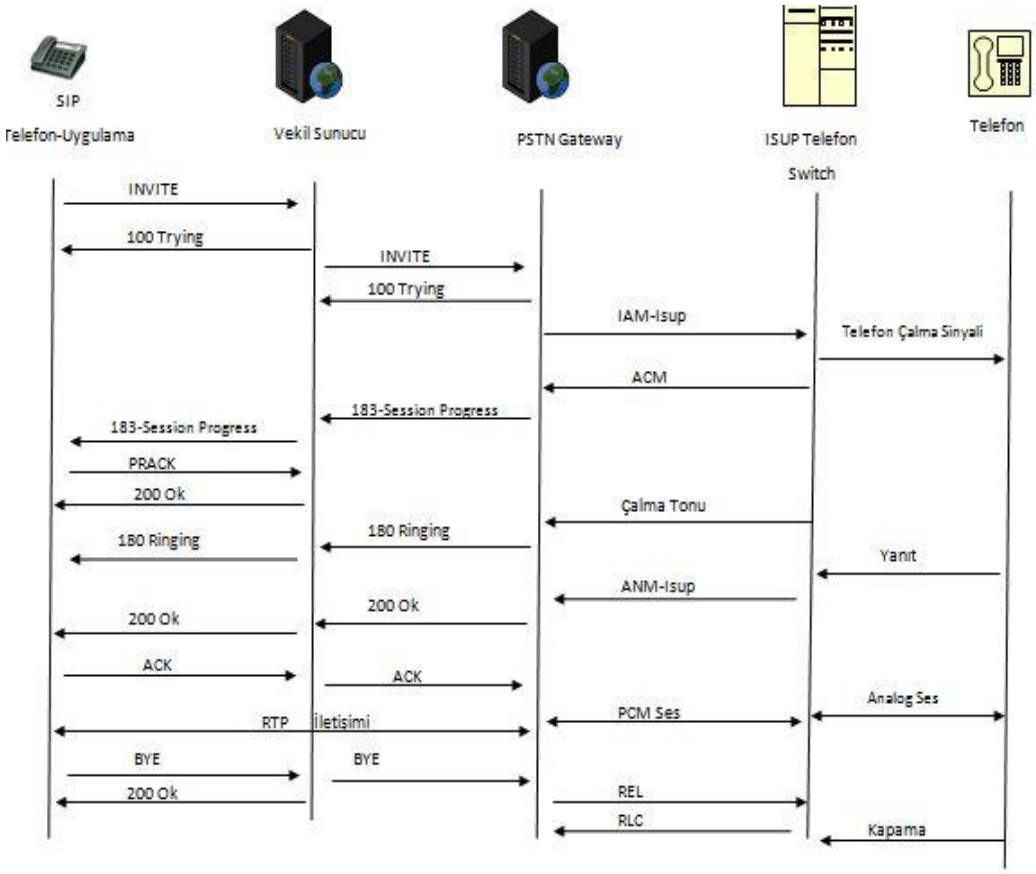
4.1 Session Intiation Protocol (SIP)

Birçok haberleşme ortamında gerek ses, video gerekse de anlık mesaj haberleşme olsun karşılıklı iletişim kurulmaya başlamadan önce, taraflar arasında belirli isteklerin yanıtların ve konuşmanın nasıl gerçekleşeceği hakkında bilgilerin iletilmesi yapılmaktadır. SIP protokolü arada bu oturumu ve düzenlemeleri sağlayan bir protokoldür. SIP protokolü kullanım alanını günümüzde özellikle internet üzerinden ses hizmeti verilmesinde yani VoIP (Voice over IP) kullanılmasına rağmen çoğu IMS (Ip Multimedia Subsystem) gibi internet üzerinden çoklu ortam verilerinin taşınmasındaki gibi yapılarda da temel protokollerden birisi olmuştur. Kısaca SIP için ses ve diğer çoklu ortam bilgilerinin taşınmasıyla ilgili bilgileri taşıyan ve bu görüşmeleri yöneten bir protokol diyebiliriz.

Teknik anlamda incelenirse SIP, FTP ve HTTP gibi uygulama katmanında bulunan bir protokoldür. SIP, TCP/IP ve UDP protokollerini kullanabilmekte fakat yaygın olarak UDP protokolü üzerinden işlem yapmaktadır [34]. SIP, oturum ve sesin nasıl ileteceği ile bilgileri taşımak ve oturumu yönetmekle görevlidir. Çoklu ortam verilerinin taşınmasını RTP (Realtime Transport Protocol) gerçekleştirmektedir.

Ayrıca SDP (Session Description Protocol) protokolü, oturumun hangi özelliklerde gerçekleşeceğini bildiren ve SIP in bir alt kümesini oluşturan bir protokoldür. SDP örneğin ses kodlamasının nasıl gerçekleşeceği gibi oturum açıklamalarını SIP INVITE mesajının içerisinde bulunarak aktarım sağlayabilir [33].

Ev telefonlarını arama gibi durumlarda ise sistem tamamen farklı çalışmak zorunda kalacaktır. Bunun sebebi normal eve telefonlarında oturum açma ve SIP mesajlarını çözümleme gibi bir yapı söz konusu değildir. Şekil 4.1'de sistem gerçekleştirimi sırasından da oluşturulacak olan bir bağlantı örneği incelenmiştir.



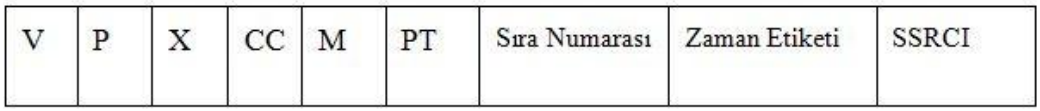
Şekil 4.1 SIP Protokolü ile Telefon Görüşme Gerçekleştirim Gösterimi [34]

Bu örnek görüldüğü gibi Telefon santraliyle SIP uygulamalarının iletişim kurabilmesi için PSTN (Public Switched Telephone Network) çıkış noktası dediğimiz sunuculara ihtiyaç vardır. Bu sunucular sayesinde telefon santralleriyle

haberleşmemiz mümkündür. Fakat haberleşme için ekstra komutlar gerekecektir. Bu mesajlara bakacak olursak SIP PSTN çıkış noktası ISUP IAM (Initial Address Message)ını telefon santraline iletilir [34]. Bu mesajın içeriğinde arayan ve aranan gibi bilgilerden bulunmaktadır. IAM mesajı iletildikten sonra PSTN sunucularından çıkış noktasına ACM (Address Complete Message) mesajı gönderilir. Bu mesajın içeriğinde telefon numarasının alındığı ve arama işleminin başladığı bilgisi bulunmaktadır [34].

4.2 Real Time Transport Protocol (RTP)

Bu protokol adında anlaşılacağı gibi verileri hızlı bir şekilde anlık iletimiyle ilgilidir ve uygulama katmanındadır. Bu protokolde performans önemlidir ve de kendine çoklu ortam uygulamalarında geniş bir kullanım alanı bulmuştur. RTP genel olarak UDP'yi kullanmakla beraber TCP için kullanımında RTCP protokolü de mevcuttur.



Şekil 4.2 Örnek RTP Protokol Yapısı [34]

Bu başlık bilgisinde V sürüm bilgisini, P (Padding) belirli şifreleme algoritmalar için paketlerin sabit uzunluğunu sağlamak için, X ek bir başlık bilgisinin bulunup bulunmayacağını, CC birden fazla RTP mesajını tek bir RTP mesajına dönüştürmede gerekli olan başlık işlemleri için kullanılır [34]. M yeni bir ses veya görüntü oluşumunu ifade eder. PT ise ses veya görüntü verilerinin hangi kodlama ile gönderildiğini belirtir [34]. Bu bilgiler SIP mesajında Invite içerisinde bulunan SDP bölümünde mevcuttur. Sıra numarası ve bağlı zaman etiketinden başka SSRCI mesajın eşsiz olmasını sağlayan bilgiyi içermektedir [34].

4.3 Metin Ses Dönüşümü ve SSML (Speech Synthesis Markup Language)

Tezdeki çözüm önerisinin temelini oluşturan noktalardan birisi de uygulamaların telefon açabilmesi sesli olarak mesajlarını iletilmesiydi. Uygulamalar ürettikleri mesajları veya kullanıcılarının girdiği verileri telefonlara aktarım esnasında metinleri sese dönüştüren bir işlem kullanmaktadır. Bu işlem genellikle bulunduğu platformla ve işletim sisteminin ses kütüphaneleriyle yakın ilgilidir.

Proje kapsamında metin bilgisinin ses dosyasına dönüşmesi sırasında Windows platformun da kullanan ses sistemi yapısını (SAPI)[40] kullanmaktadır. İnternetin gelişmesiyle beraber ses sentezleme işlemiyle ilgili yönergeler belirli bir standart yapı içerisinde de gerçekleşebilmektedir Bu da SSML (Speech Synthesis Markup Language)[35] diliyle olabilmektedir. Bu dil sayesinde sentezlenecek olan sesin telaffuzu, belirli bölümlerdeki vurgular, ses düzeyi ve bunun gibi dilsel işlemler kontrol altına alınabilir.

```
<?xml version="1.0"?>
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
   http://www.w3.org/TR/speech-synthesis/synthesis.xsd" xml:lang="en-US">

Okunacak Bölüm 1….<break strength="weak"/>

Okunacak  Bölüm2…..

<break time="5s"/> Bölüm3…..

<voice name="SistemdekiSes">bölüm4….</voice>

. <emphasis level="strong"> vurgulu bölüm  </emphasis> Bölüm5
</speak>
```

Şekil 4.3 Örnek SSML Dokümanı

Temel SSML doküman yapısına ve etiketlerine bakarsak; SSML speak etiketiyle başlar ve biter. Break oluşturulan ses arasındaki boşlukların ne kadar süreceğini ve

boşlukların verilme şiddeti düzenler [35]. VoiceName ile sistemdeki mevcut seslerden hangisini kullanarak sentezleme yapılacağına karar verilir. Emphasis vurgulanacak yerleri ve vurguların seviyelerini belirtir. Auido ses oluşturulurken araya hazır bir ses dosyası eklemek içindir. Phoneme ve lexicon fonetik ve telaffuz bilgileri için gerekli etiketlerdir [35].

4.4 VoiceXML

Bu protokol bilgisayar sistemleriyle kullanıcılar arasında gerçekleşebilecek olan muhtemel diyalogları tanımlamak için kullanılmaktadır ve bir W3C standardıdır. HTML ekran görüntülerinin nasıl görüneceğini tanımlarken VoiceXML protokolü de görüşmelerin nasıl gerçekleşeceğini belirtmektedir. Bu protokolün hedefleri arasında tek bir doküman içerisinde birden fazla etkileşimi barındırdığı için istemci sunucu etkileşimini en aza indirgemek, diyalogları platform bağımlılığından ve de sunucu iş mantığı kodlarından soyutlamak ve diyalogları basitleştirmek bulunmaktadır [36]. Görüşme diyaloglarının tasarımı dışında ses sentezlemesi, okunacak olan ses dosyalarının belirlenmesi, diyalogların kayıt edilmesi, basılan tuş tonlarının tespiti ve çeşitli telefon işlem özellikleri de görevleri arasında sayılabilir [36].

Bir muhtemel gerçekleşebilecek sistem-kullanıcı görüşmesi ve bunun akışını sağlayacak olan bir VXML okuman örneği ise;

Sistem: Sonuç öğrenme servisine hoş geldiniz. Lütfen sonuç numaranızı giriniz.

Kullanıcı: 1234

Sistem: Girdiğiniz numara 1234'tür doğru mu? Doğruysa 1 'e basın

Kullanıcı: 1

Sistem: Lütfen Şifrenizi giriniz

Kullanıcı: 1234

Sistem:Sonuçlarınız …….. dir. İyi günler

Şekil 4.4 Örnek VoiceXML Dokümanı

Bu VXML kodları içerisinde bulunan temel elemanları incelersek: <form> belirli bir diyalog bölümü oluşturmak için kullanılmaktadır. <audio> ses sentezlenmesi ve belirli bir ses dosyasının çalınması için kullanılabilmektedir.<field> temelde kullanıcıdan gelen değerlerin tespiti için kullanılır.<nomatch> eşleşme olmaması durumunda, <filled> ise olayların tamamlanıp başarılı bir giriş elde edildiğinde yapılacakları ifade eder. <catch> programlama dillerindeki gibi hata durumlarında yapılacakları, <goto> da VXML dokümanı içerisinde başka bir form yapısına yönlenmeyi ifade eder.

Burada yönlenme sonucunda sistem tarafından yanıtları iletecek başka bir VXML doküman yapısı oluşturulup kullanıcıya sonuçların hangi şekilde iletileceği bildirilebilinir.

```
<?xml version="1.0" ?>
<vxml version="2.0">
   <var name="sonucNumarasi" />
   <form id="NumaraGiris">
      <block>Sonuç öğrenme servisine hoş geldiniz. </block>
      <field name="sonucNo" type="digits?minlength=4;maxlength=6">
         <prompt>Lütfen sonuç numaranızı giriniz?</prompt>
         <noinput count="1">
            <audio>Yanıt gelmedi</audio>
            <reprompt />
         </noinput>
         <noinput count="2">
            <audio>Lütfen tekrar deneyiniz iyi günler</audio>
            <exit />
         </noinput>
         <nomatch>
            <audio>Geçerli bir numara girmediniz</audio>
            <reprompt />
         </nomatch>
```

Şekil 4.5 Örnek Diyaloğun VoiceXML Kodlaması

```
<form id="SonucNoDogrula">
      <block>Girdiğiniz numara <value expr="sonucNumarasi" /></block>
      <field name="kontrol" type="boolean">
        <prompt>ür doğru mu .Doğruysa 1 ‘e basın</prompt>
        <filled>
          <if cond="kontrol">
            <goto next="#SonucGoster" />
            <else />
            <goto next="#NumaraGiris" />
          </if>
        </filled>
      </field>
  </form>
 <form id="SonucGoster">
      <field name="sifre" type="digits?length=4">
        <prompt>Lütfen Şifrenizi giriniz?</prompt>
        <noinput count="1">Şifre girilmedi.<reprompt /></noinput>
        <nomatch count="1">Dört basamaklı bir sayı girmelisniz<reprompt /></nomatch>
        <catch event="eslesme yok" count="3">
          <audio>Lütfen sonra tekrar deneyin</audio>
          <exit />
        </catch>
         <filled>
          <submit next="SonucOgren.aspx" method="POST" namelist="SonucNo sifre" />
         </filled>
       </field>
    </form>
</vxml>
```

Şekil 4.6 Örnek Diyalogun VoiceXML Kodlaması

Bu protokollerle bağlantılı ve birlikte çalışabilen diğer protokoller ise; Call Control eXtensible Markup Language (CCXML) telefon görüşmeleri, görüşme transferleri gibi daha çok telefon servislerine bağlantı ve bu tür ihtiyaçların düzenlemesi için kullanılan bir protokoldür. Media Server Control Markup Language (MSCML) protokolü ise çoklu ortam sunucusun kontrolü için geliştirilmiş bir protokoldür VoiceXML bu iki yapıda da etkileşimli kullanıcı işlemleri için kullanılabilmektedir [38].

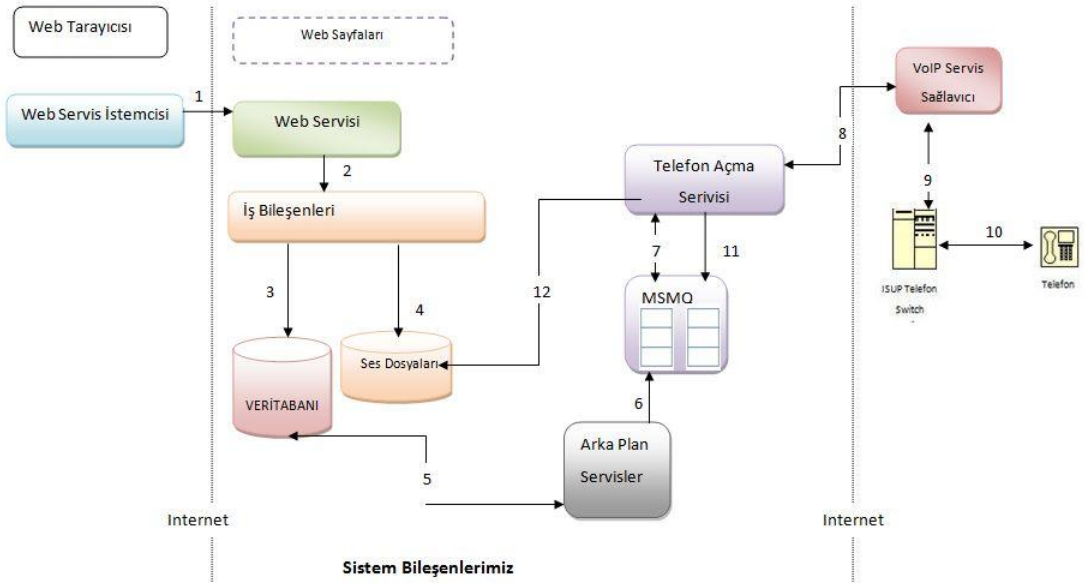
5. SİSTEM GERÇEKLEŞTİRİMİ

Beşinci bölüm sistem gerçekleştirimi üzerine olup, sistemimizin temelden nasıl bir gelişim sürecine girdiğini göstermek için çeşitli sürümler halinde geliştirilmiştir. Bu servislerin gerçekleştirimi aynı zamanda teknolojideki geçmişten gelen mevcut yaklaşımlarla paralellik göstermiştir. Böylelikle mevcut duruma nasıl bir süreçten geldiğinin görülmesi sağlanmıştır. Hangi eksikliklerin ne çeşit yöntemler, teknolojiler ve yaklaşımlar ile üstesinden gelindiğini göstermeyi amaçlamaktadır. Bu yüzden bu bölümde sistem gerçekleştirilmesine temel düzeyden başlanmış ve aynı servisin gelişmiş sürümleri devam eden bölümlerde gerçekleştirilerek, sisteme yaptığı katkılar gösterilmiştir. İlk olarak ses hizmeti vereceğimiz için bu hizmeti sağlayacak olan temel yaklaşımlar, sonraki bölümde ise bilgisayarın gerçekleştireceği metinden ses sentezleme işlemleri anlatılacaktır. En son bölümde ise servislerimizin gerçekleştirim bölümlerine geçilecektir.

Servislerin basamaklı olarak geliştirilmesinin bir nedeni de; öncelikli olarak şu ana kadar sektörde yapılan benzer çalışmalarla, kendi geliştirdiğimiz giriş düzeyinde servislerle yetenek ve yaklaşım olarak benzer şekilde eşitlenmeye çalışılmıştır. Sonrasında bizim bu servislere getirdiğimiz teknolojik yaklaşımların ve yeteneklerin servis yönelimli bir yapıda tasarlanmasının, benzer çalışma yapan akademik ve kurumsal yapılara ışık tutacağı düşünülmektedir.

İleri düzey servisler geliştirilmesinden sonraki bölümde ise bu yapının nasıl SaaS olarak nasıl uygulanabilineceği konusunda bir fikir verebilecek temel düzeyde bir çalışma örnek olarak gösterilmiştir.

Öncelikli olarak temel düzeyde sistemin çalışmasını basamak basamak incelersek



Şekil 5.1 Sistem Genel Yapısı

Örneğimizde 1 numaralı işlemle bir web servisi istemcisi web servisinin arama gerçekleştirme ile ilgili metotlarından birisini çağırıp ve arama bilgilerini parametre ile servise göndererek arama gerçekleştirme işleminin ilk basamağını başlatmıştır. İkinci basamaktaki web servislerinin servis bileşenleriyle birlikte 3. basamaktaki veritabanına kayıt işlemini ve 4. basamaktaki servise gönderilen görüşme mesajının ses dosyasına dönüşümünü görebiliriz.

Bu noktada web servisin görevi tamamlanmış olup arama vaktinde çağrılmak üzere gerekli bilgiler veritabanına işlenmiş ve gerekli ses dosyası Aramalar klasöründe oluşturulmuştur. 5. basamakta arka plan servisleri belirli aralıklarla veri tabanını kontrol ederek o anda gerçekleştirilecek olan görüşmeleri veri tabanından çekmektedir. 6. basamakta telefon açma servislerinden birisinin müsait olduğu zamanda bu arama bilgisini alması için arama bilgilerini kuyruğa gönderir. 7. basamakta Telefon Açma servisi aramaya müsait olduğu bir zamanda kuyruğu kontrol ederek sırada bekleyen görüşme olup olmadığına bakar ve görüşme varsa bu bilgileri kuyruktan alarak arama işlemini gerçekleştirir. 8. basamakta arama işlemi için VoIP servis sağlayıcı ile olan bağlantı görülmektedir. 9. ve 10. aşamalarda görüşmenin VoIP servis sağlayıcısından çıkıp normal bir ev telefonunda sonlanma işlemi gösterilmektedir. Görüşme çift yönlü gerçekleştirilirken ses kaydı ve tuş

sonuçları Telefon Açma servisi tarafından kaydedilir. Bu kayıt bilgileri; arka plan servislerinin bunları tekrar alması ve tuşlar için 11. basamakta olduğu gibi kuyruğa gönderilirken, 12. işlemde ses kayıtları disk üzerinde bir ses dosyası olarak saklanmaktadır.

5.1 SES HİZMETLERİ GERÇEKLEŞTİRİMİ

Ses hizmetlerinin gerçekleştirimi diğer adıyla VoIP işlemlerinde karşımıza birden fazla yöntem çıkabilmektedir. Bunlardan birincisi arama trafiğinin yoğun olacağı bir yapıda araya bir SIP sunucusu kurarak tüm farklı kullanıcılardan gelen trafiği doğrudan servis sağlayıcıya topluca iletmektir. Fakat bu durumda çoğu işletmenin VoIP lisansını satın alamayacağı için VoIP servis sağlayıcılarla belirli düzeyde anlaşmaları gerekmektedir. Bu yüzden bu tür hizmet vermeyi düşünen küçük işletmeler için bu durum uygun olmayabilir. Fakat kurumsal yapıda bir SIP sunucusu üzerinden işlemlerin bir VoIP Gateway dediğimiz yapılarla geliştirilmesi tavsiye edilmektedir. İlk bölümünde Microsoft Unified Communication Server üzerinden böyle bir sistemin nasıl geliştirilebileceği anlatılmıştır.

Fakat daha ufak ölçekli yapıların yüksek trafik anlaşmaları yapmadan basit bir VoIP istemcisi olarak da sistemi kullanabilmeleri için sunucuda telefon açma isteklerinin bir sanal SIP telefonu ve tek bir hesap üzerinden gerçekleştirilmesi yöntemi uygulanabilir. Bu durumda da gene Windows işletim sisteminin bünyesinde bulanan Unified Communication Server API'larına oldukça benzeyen ve bu API'nin temelini oluşturan RTC kütüphaneleriyle beraber farklı SIP kütüphaneleri de sisteme dahil edilebilmektedir. İkinci bölümde ise bu konulara değinilmiştir.

5.1.1 SIP Sunucularına Bağlantı ve Haberleşme:

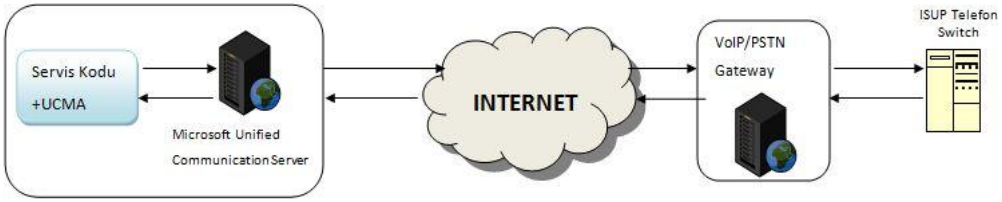
Bu bölümde VoIP hizmetlerinde kullanılan temel protokol olan SIP protokolünün sistem içerisinde nasıl kodlandığını ve çözümlendiği işlenecektir. 4.1. numaralı bölümde temel SIP komutları ve mesaj yapısına kısaca değinilmişti.

SIP protokolünün internet üzerinden çoklu ortam uygulamalarında yaygınlaşması sonucunda yazılım firmaları bu konudaki yatırımlarını arttırmış ve mevcut konferans sunucu gibi iletişim uygulamalarını da SIP protokolü üzerine taşımıştır. Bu yaklaşımı benimseyen yazılım firmalarından birisi de Microsoft firmasıdır. Gerek konferans, ses sunucuları gerekse de telefon sunucularını tek bir yapı altında birleştiren firma, bu birleşme sonucunda Microsoft Unified Communications Server ürününü çıkarmıştır [41].

Temelde bu ürün SIP sunucusu olarak da çalıştırılabilmesine karşın bizim projemiz kapsamında SIP sunucusu olarak kullanılmamaktadır. Bunun asıl nedenlerinden birisi projemiz içerisinde SIP sunucusu olarak VoIP hizmeti veren bir firmayı seçmek zorunda kalmamızdır. Çünkü proje kapsamındaki çözümlerimizde uygulamaların telefon açabilmesi ve bu telefon hizmetlerinin genellikle cep telefonları ve iş-ev telefonları üzerinden gerçekleşecek olmasıdır. Bu durumda belirli bir PSTN santraline bağlanma gerekliliği bu durum ek bir maliyet ve internet üzerinden ses iletimi lisansı gibi yasal sorunları beraberinde getirebilecektir. Bu durumun aşılması için güvenilir bir VoIP hizmeti veren uluslararası bir kurum olan CallCentric [60] seçilmiştir. Ayrıca bu sayede servislerimiz in ölçeklenebilirlik açısından uluslararası bir boyut kazanması durumunda sistem değişimi yapılmadan servis hizmetleri sürdürülebilir.

Bu durumda Proxy Server olarak kullanacağımız Microsoft Unified Communications Server ürünü de aynı zamanda bizlere CallCentric firmasının sağlayacağı Proxy görevini sağlayacaktır.

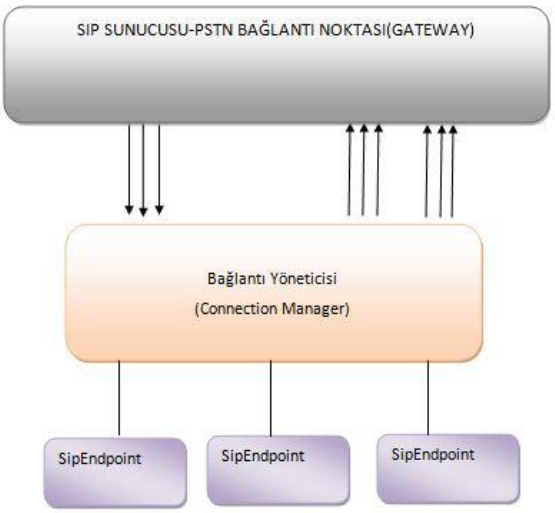
Fakat Micorosft UC Server kullanıldığı takdirde PSTN çıkışları için doğrudan PSTN sunucularına bağlanmaya olanak tanıyan farklı bir VoIP servis sağlayıcıyla çalışmak zorunda kalınabilir.



Şekil 5.2 Microsoft Unified Communications Server üzerinden sistemin çalışması

SIP İşlemlerini gerçekleştiren bileşenleri, kodları ve kodların çalışma şekillerini maddeler halinde incelersek;

1- Microsoft.Rtc.Collabration bileşeni sisteme başarılı bir şekilde dahil edildikten sonra belirli bir SIP Sunucusuna bağlanmak için kod içerisinde gerçekleştirilecek olan ilk işlem; projeye bağlantıların yönetiminden sorumlu olan bir Bağlantı Yöneticisi (Connection Manager) eklemektir. Bağlantı yöneticisi en alt seviyede sunucu ile gerekli olan haberleşme işlemlerini gerçekleştirecektir. Birden fazla SIP bağlantı son noktası (SipEndpoint) tek Bağlantı Noktası üzerinden işlem yapabilmektedir. Bu işlemler belirli bir havuz üzerinden gerçekleştirilir.



Şekil 5.3 Bağlantı Yöneticisiyle SIP Sunucusuna Bağlanma

Bu yapıyı sağlayan kodların bir bölümü Şekil 5.4 üzerinde görülmektedir. Ve buradaki kodlarda bileşenin doküman kodları temel alınmıştır.

Bu kod bölümlerindeki açıklamalardan da anlaşılacağı gibi Sip bağlantısı için kullanacağımız nesnelerin Namespace'i, Sip bağlantısı ve de bağlantı nesneleri sırayla tanımlanmıştır. Sonraki bölümde new ile bu tanımlamalara bağlı nesne oluşturmaları yapılmıştır. Bu oluşturma sırasında SipBaglanti nesnesinin haberleşeceği adres, protokol türü, bağlanacağı domain, bağlanacağı port ve hangi bağlantı nesnesi üzerinden bağlanacağı gibi bilgiler mevcuttur

```
//Namespace kodumuza dahil edilmektedir.
using Microsoft.Rtc.Signaling;
. . . . . .
//Sip bağlantısını tanımlayacak nesnemiz ve
// bağlantıyı yönetecek olan bağlantı yöneticisi tanımlaması
  internal SipEndpoint SipBaglantisi;
  RealTimeClientConnectionManager BaglantiNoktasi;
. . . . . .
BaglantiNoktasi=  new RealTimeServerTcpConnectionManager ();
SipBaglantisi = new SipEndpoint (sipAdresi,SipA
SipTransportType.Tcp,
"UCSSUNUSUSU",5060,false,BaglantiNoktasi,"X1");
```

Şekil 5.4 SIP Sunucuya Bağlantıyı Tanımlayan Kod Örneği[41]

2- Temel olarak Bağlantı yöneticisini ve bağlantı noktasını oluşturduktan sonra kullanıcıların sunucuya bağlantılarını sağlayacak olan kullanıcı girişleri için kullanıcı isimleri ve şifreleri, sonrasında ise kaydolma işlemi (register) başlaması için gerekli metotlar girilmektedir. Bu metotlar Şekil 5.5'deki gibi olabilmektedir.

```
//Kullanıcı ve Şifre  Tanımlamaları
 System.Net.NetworkCredential giris;
   giris= new System.Net.NetworkCredential ();
   giris.UserName=kullaniciAdi;
    giris.Password=sifre;
    //Tanımlamanın Bağlantı noktasına eklenmesi
         sipBaglantisi.CredentialCache.Add
(SipEndpoint.DefaultRtcRealm,giris);


//Sunucuya Kayıt işleminin  başlaması
SipResponseData response =  sipBaglantisi.EndRegister
(sipBaglantisi.BeginRegister (null, null));
```

Şekil 5.5 Sunucuya Bağlantıyı Gerçekleştiren Kod Örneği [41]

3- Sunucuya kayıt işlemi gerçekleştikten sonra oturum açma işlemlerine geçilir. Burada SIP protokolü üzerinden bölüm 4.1'de bahsettiğimiz Invite mesajı ile haberleşme başlayacaktır. Şekil 5.6'de SIP protokolünün temelini oluşturan noktalardan birisi olan oturum nesnesi oluşturulmaktadır.

```
  private SignalingSession oturum
. . . . . .
// Oturum nesnesi oluşturumu
oturum = new SignalingSession (sipBaglantisi, new
RealTimeAddress (oturumUri));


   try
     {          oturum.EndParticipate
(oturum.BeginParticipate (null, null));
}
    catch  (FailureResponseException e)
     {
     }
```

Şekil 5.6 Tanımlamalardan Sonra Oturum Açma Kodu [41]

4- Bu aşamada ise ortam gereksinimlerini ve haberleşmenin detaylarını bildiren Sessison Description Protocol (SDP)'nin nasıl oluşturulduğu incelenmektedir.

Burada SDP işlemleri kodumuz içerisinden sınıfımıza eklenecek olan IOfferAnswer

```
private ContentDescription SdpBilgi  (SignalingSession oturum)
  {
 Sdp<SdpGlobalDescription, SdpMediaDescription> OturumIcerik
 =new Sdp<SdpGlobalDescription, SdpMediaDescription> ();
  System.Net.IPAddress ipAddress;
  ipAddress = oturum.Connection.LocalEndpoint.Address;

OturumIcerik.GlobalDescription.Origin.Connection.Set
(ipAddress.ToString ());
OturumIcerik.GlobalDescription.Connection.TrySet (ipAddress
.ToString ());
OturumIcerik.GlobalDescription.Origin.Version = 0;
OturumIcerik.GlobalDescription.Origin.SessionId = sessId;
OturumIcerik.GlobalDescription.Origin.UserName = kaAdi;
SdpMediaDescription icerik =
new SdpMediaDescription ("audio");
 icerik.Port = 5061;
 icerik.TransportProtocol = "RTP/AVP 0 8";
 icerik.Formats = "null";
 SdpAttribute ozellik =
new SdpAttribute ("accept-types", "text/plain");
icerik.Attributes.Add (ozellik);
OturumIcerik.MediaDescriptions.Add (icerik);
System.Net.Mime.ContentType ct = new
System.Net.Mime.ContentType ("application/sdp");

return new ContentDescription (ct,OturumIcerik.GetBytes ());
        }
```

Şekil 5.7 SIP Bağlantı Özelliklerini Tanımlayan SDP Dokümanı [41]

ara yüzünün eklenmesiyle gerçekleştirilecektir. Bu ara yüzle birlikte GetAnswer, GetOffer, HandleOfferInInviteResponse, HandleOfferInReInvite, SetAnswer metotları kod içerisinde uygulanabilir. GetOffer sistem kod tarafından tetiklenen iletişimde Invite komutunun çalışması ve ortam bilgilerinin ve türlerinin karşı sisteme aktarması için kullanılmaktadır. SetAnswer ve GetAnswer karşı taraftan gelen ortam bilgileri ve Invite mesajı için gereklidir. [41].

Bu fonksiyonda SDP doküman yapısının tanımlaması yapılmıştır. Kodları sırayla incelenirse; Arayüz içerisinden çağırılacak bu sınıfta ara yüzden bizim tanımladığımız fonksiyona oturum nesnesi aktarımı olacak ve bu oturum nesnesi üzerinden IP adres çözümlemesi yapılmaktadır. Oturum içeriğini belirten nesnenin tanımlaması ve oluşturulmasından sonra, sıra SDP doküman yapısını oluşturan bilgilere gelmektedir. Connection SDP dokümanındaki c ile isimlendirilen bağlantı adres bilgilerini belirtir. Attributes SDP dokümanındaki a bilgisi için belirtilmiştir. Port haberleşmenin sağlanacağı port numarası m özelliği içerisinde bulunur. MediaName ne tür verinin taşınacağını belirtir ve m özelliği içersinde bulunmaktadır. Transport Protocol de son olarak verilerin nasıl taşınacağını belirtir ve SDP içerisinde m elamanının son bölümünü oluşturmaktadır. Ve son olarak SDP dokümanı mime olarak kodlanıp Invite mesajlaşması içerisinde yer alabilecek bir yapıya kavuşur.

5- Proje kapsamında metinden sese dönüşüm sağlanmakta ve ses iletimi belirtilen hedeflere disk üzerinde oluşturulan ses dosyasından yapılmaktadır. Ses dosyalarının karşı kullanıcıya iletilmesi için belirli bir Player nesnesi oluşturulmaktadır. Bu Player nesnesi içerik olarak ayrı bir MediaSource nesnesi kullanarak iletişimi sağlar. Görüşme AudioVideoCall ve AudioVideoFlow nesneleri üzerinden gerçekleşmekte ve Player bu nesnelerle temas halinde görüşmeleri aktarmaktadır.

```
Player player = new Player ();
player.AttachFlow (gorusmeAkisi);
. . . . . .
WmaFileSource source = new WmaFileSource (dosya);
          source.EndPrepareSource
(source.BeginPrepareSource
 (MediaSourceOpenMode.Buffered, null, null));
. . . . . .
player.SetMode (PlayerMode.Automatic);
player.SetSource (source);
player.Start ();
```

Şekil 5.8 Mevcut Ses Dosyasını Mevcut Görüşmeye Aktaran Kod Örneği [41]

Player nesnesinin kontrolü ayrıca AudioVideoFlow nesnesinin olayları içerisinde kontrol edilir.

6- Kullanıcıdan gelen tuş değerlerinin alınması ve servis tarafından bu değerlerin kullanıcılara geri bildirilmesi, servisteki kullanıcı-servis etkileşiminin temelini oluşturan noktalardan birisidir. Gelen tuş değerlerine göre servisi çağıran kod gerekli yanıtları bu sayede işleyebilmektedir. DTMF ton algılamasının kod yapısı itibariyle ses gönderime benzemekte olup Player nesnesi yerine ton yakalamasında ToneController nesnesi görev yapmakta ve görüşme akışına bu nesne dahil edilmektedir. ToneController nesnesinin ToneReceived olayında ise kullanıcıdan gelen tuş değerleri alınıp işlenebilinir.

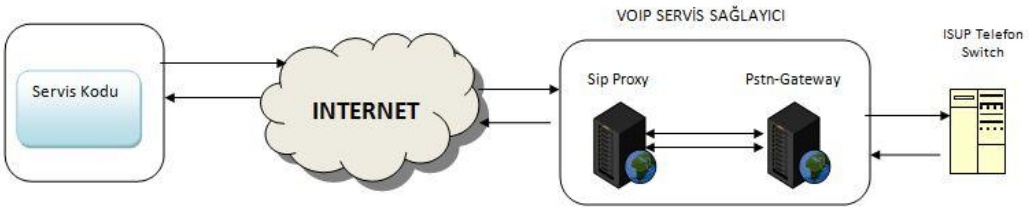
```
ToneController tonKontrol = new ToneController ();
 tonKontrol.AttachFlow (gorusmeAkisi);
```

Şekil 5.9 Mevcut Görüşme İçerisinde Kullanıcı Tuşları Algılama Özelliği Kullanımı [41]

5.1.2 Alternatif SIP Bağlantı Yaklaşımı

Bir önceki bölümde ele alınan yaklaşımda servis arka planda çalışan haberleşmeden sorumlu bir sunucu üzerinden işlemler gerçekleşmektedir. Bu sunucu her türlü cihaz ve platformun ortak bir zeminde SIP protokolü üzerinde haberleşmesine olanak tanımaktadır. Bu zengin çeşitliliğe karşın bizim servis uygulamamız bazı özel işlemler için kullanılabilir. Bu durumda, bizim servisimize bağlı bir SIP sunucusunu yapılandırmamıza gerek kalmayabilir. Burada servis tarafında bulunan kodumuzu bir bakıma bir SIP telefonu gibi yapılandırıp VoIP sunucuları ile bağlantıya geçmek durumundayız. Bu noktada Windows kütüphaneleriyle birlikte gelen Microsoft.Rtc kütüphanesi [43] SIP haberleşmesi için temel bir yapıya olanak sunmakta fakat verilerin gönderimindeki RTP protokolü için yetersiz kalabilmektedir. Bu durumda gene Microsoft Research tarafından geliştirilen RTP Kütüphanesi sisteme dahil edildiğinde bu 2 yapının hibrit kullanımı bizi çözüme götürebilmektedir [42]. Ayrıca RTC Kütüphanesi 5.1.1. bölümünde bahsedilen UCMA kütüphanesinin temellerini oluşturmaktadır.

Alternatif SIP Bağlantı Yaklaşımı

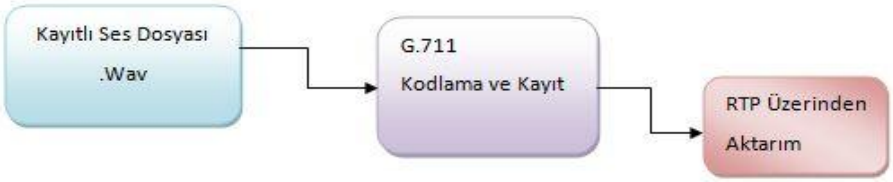


Şekil5.10 SIP Bileşe Koduyla Normal Bir VoIP Kullanıcısı Olarak Arama İşlemi

Burada servis tarafında yapılan işlemleri ve telefon açma ile ilgili kodları basamak basamak incelersek;

1) Birinci basamakta 5.1.1 numaralı bölümde bahsettiğimiz SIP bağlantısı, INVITE mesajı ile görüşme anlaşmaları ve SDP mesajları oluşturulmaktadır. İstemci içerisinde oluşturulan kütüphanelerden CreateSessionWithDescription ile yapılandırılan SDP dokümanı, yeni oluşturulan istemci sınıfındaki oturum oluşturma nesnesi içinde oluşturulur. Bu şekilde SIP Proxy sunucusuna görüşmenin nasıl gerçekleşeceği bilgisi aktarılır.

2) Gerekli anlaşmalar sağlandıktan sonraki kısımda yapılacak olan işlem verilerin transferi olacaktır. Bu verilerin RTP üzerinden gönderiminde ses dosyaları disk üzeriden okunup G.711A olarak kodlanıp RTP üzerinden gönderilecektir. G.711 kodlaması hemen hemen bütün VoIP servis sağlayıcıları tarafından desteklenen temel bir kodlamadır. Ses dosyaları bu kapsamda gelen bilgilerin alınması ve geri çözümleme işlemine tabi tutulması söz konusu olacaktır [44]. .Net kütüphaneleri içerisinde bulunan FileStream nesnesi ses dosyalarındaki verileri okuyup gerekli kodlamaları yaptıktan sonra ses kanallarına aktarım işlemini gerçekleştirebilmektedir.



Şekil 5.11 Ses Dönüşüm İşlemi Sırası

Burada wav dosyalarının büyüklüğü ve projedeki görüşmelerin öncelikli olarak eş zamanlı olmayışı sayesinde dosya dönüşüm; işlemcinin durumuna göre belirli bir zamanda gerçekleştirilerek belirli bir noktada saklanabilir. Ve görüşme esnasından RTP üzerinden aktarım sağlanırken dönüşüm yapılmadan işlemci üzerindeki yük azaltılabilir.

3) Fakat bu durum incelendikten sonra Microsoft Firmasının Tüm bu işlemlere gerek duymadan da metinden istenilen ses dönüşümü, işleminde istenilen formatta çıktı verebileceği görülmüştür. 2. maddede belirtilen durumlar bu durumda sadece anında

ses iletimi yapacak muhtemel çalışmalara ışık tutması açısından tez kapsamından çıkartılmamıştır. Bu durum 5.2. bölümde ses sentezlemesinde de incelenecektir.

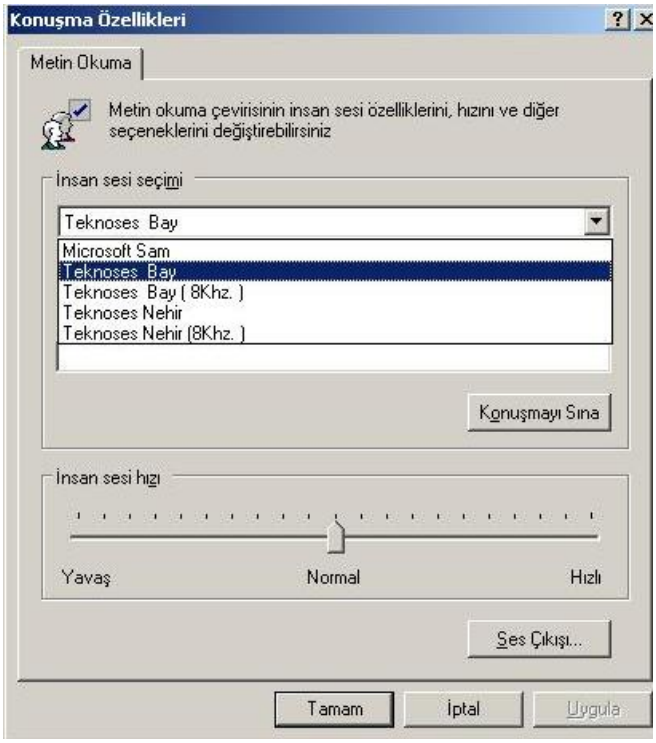
4) RTP mesaj gönderiminde Microsoft Research tarafından geliştirilen ve .Net uyumlu bileşen kullanıldığından bahsedilmişti. Öncelikli olarak MSR.LST.Net.Rtp bileşeni proje kodumuza dahil edilmektedir.Windows işletim sistemiyle birlikte gelen RTP kütüphanesi dxmrtp.dll olup bizim kullanacağımıbileşen http://conferencexp.codeplex.com/ üzerinden indirilebilinir. Gerçekleştirim sırasında kullanılan kodların bir bölümü bu bileşenin yardım dokümanlarından ve örnek kodlar incelenerek oluşturulmuştur [42].

Bu iki yapıdan oluşturulan ve örnek bir VoIP görüşmesi gerçekleştiren kod CodeProject web sitesinde [44] bulunmaktadır. Ve gerçekleştirimde bu kod tabanı esas alınmıştır.

Bunlardan farklı Microsoft'un sistem API'sine benzer şekilde diğer firmaların da geliştirdiği Ör: SIPSDK, ConaitoSIP, Sip.net gibi bileşenler de proje kapsamında benzer biçimde kullanılabilir [61]

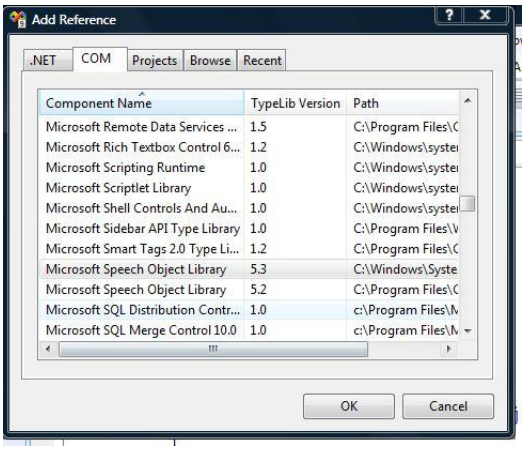
5.2 Metin Bilgisinden Ses Sentezlemesi

Çözüm önerisinin temel unsuru uygulamaların telefon açma özelliğinde istemcilerden alınan metin bilgilerinin sunucu üzerinde belirli insan sesine dönüştürülüp aktarılmasıdır. İşletim sisteminde metin bilgisinin insan sesiyle okunması için gerekli bileşenler mevcut olup; kullanıcı arabirimleriyle bu bileşenlere kod içerisinden erişim ve ses sentezlenmesi mümkündür. Windows İşletim sisteminde Türkçe konuşma desteği olmadığı için gerçekleştirim kapsamında Teknoses [39] firmasının geliştirdiği ses sentezleme bileşenleri kullanılmaktadır. Böylelikle işletim sistemine Türkçe ses sentezleme özelliği kazandırılmış olup gene işletim sistemiyle gelen kullanıcı arabirimleri sayesinde uygulama geliştirmek mümkündür.



Şekil 5.12 İşletim Sistemi Ses Yönetim Paneli ve Sisteme Eklenen Sesler

Şekilde Türkçe ses yüklendikten sonra sistemdeki sesler görülmektedir. Burada istendiği takdirde aynı uygulama içerisinde birden fazla ses seçimi mümkün olabilmektedir. Kullanıcılar isterse Bay-Bayan sesi seçebilmekte ve hatta servis farklı dillerde sentezlemeye olanak tanıyarak aynı uygulama aynı kod üzerinden birden fazla dilde hizmet verebilmektedir.



Şekil 5.13 Visual Studio'da Speech Object Library'nin eklenmesi.

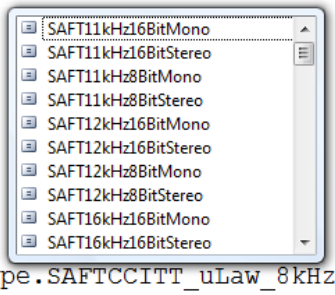
Ses sentezlemesini gerçekleştiren kodları incelersek; Ses dönüşüm bileşenlerinin kodumuz içerisine dahil edilmesi için Add Reference bölümünden Speech Object Library [40] projemize dahil edilir (Şekil 5.13)

Ses dönüşüm C# kodlarının genel yapısını inceleme durumunda ise, gerekli ad uzayı kodlara dahil edildikten sonra ses dönüşüm kodları Şekil 5.14'deki gibi

```
private void SesDonusum (string okunacakMetin, string dosyadi)
  { // Ses Dosyasının Oluşturulması
  SpFileStream dosya = new SpFileStream ();
  dosya.Open (dosyadi,
     SpeechStreamFileMode.SSFMCreateForWrite, true);
 //Metin Bilgisinin okunması ve dosyaya kayıt işlemleri
   SpVoice ses = new SpVoice ();
   ses.AudioOutputStream = dosya;
   ses.Speak (okunacakMetin,
          SpeechVoiceSpeakFlags.SVSFlagsAsync);
   ses.WaitUntilDone (1000);
   dosya.Close ();       }
```

olabilmektedir.

Şekil 5.14 Örnek Ses Sentezleme c# kodu [40]




```
af.Type = SpeechAudioFormatType.SAFTCCITT_uLaw_8kHzMono
spFileStream.Format = af
```

Şekil 5.15 Farklı Ses Formatlarında Ses Sentezleme İşlemlerinin Seçimi

5.3 Temel Web Servisi ile Gerçekleştirim

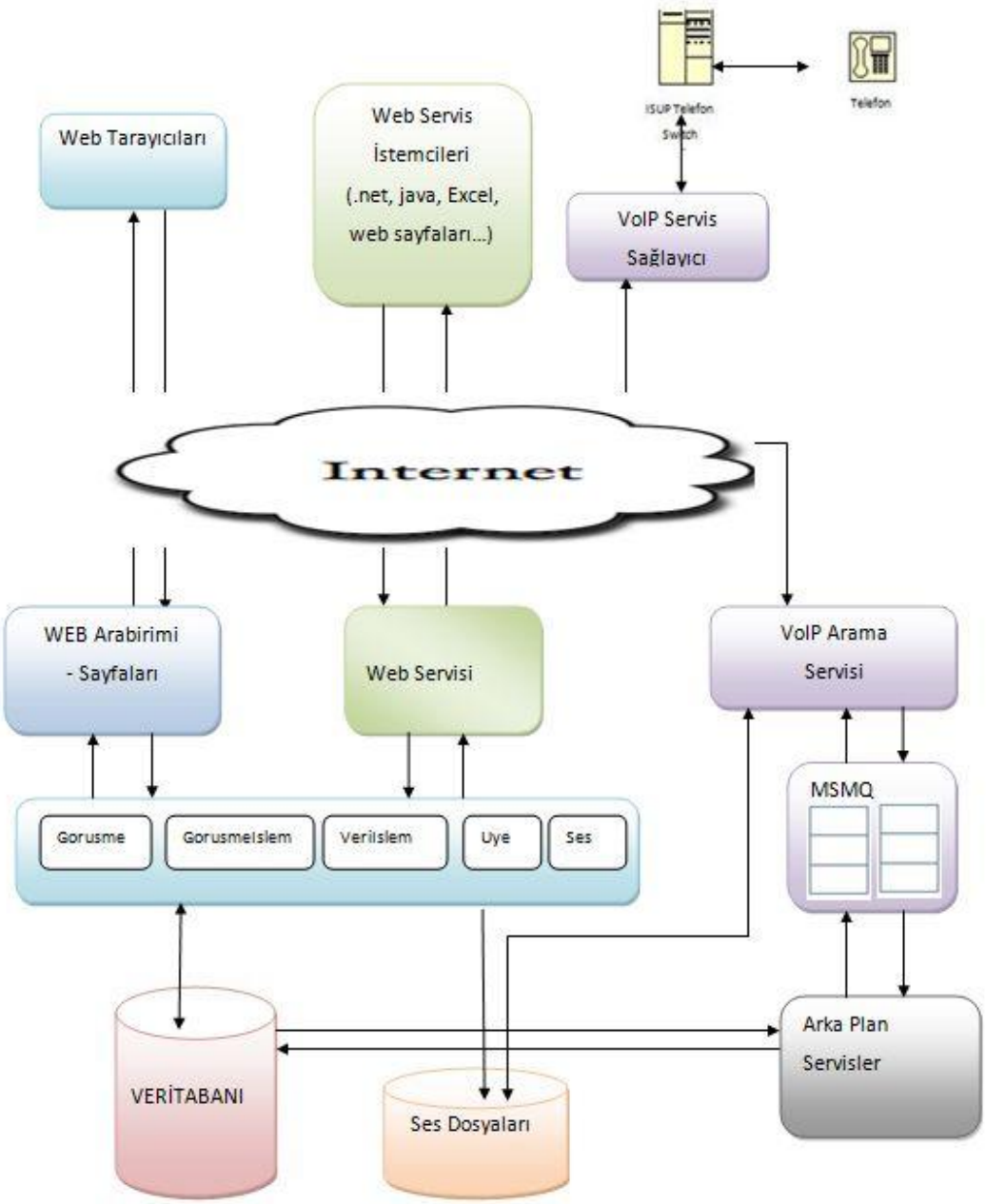
Beşinci bölümün ilk iki maddesinde ses bilgilerinin nasıl oluşturulduğu, hedefe nasıl gönderileceği ve bu işlemleri yapacak olan kodların, bileşenlerin nasıl tasarlanabileceği üzerinde duruldu. Gerçekleştirimin bundan sonraki basamaklarında ise sunucu tarafında kullanıcılarla etkileşim içerisinde olacak kodların tasarlanması yer alacaktır. Burada servis kodları arka planda ilk iki bölümde tasarlanan VoIP bileşenlerine ve sunucularına erişilmesi ile kullanıcıların bu hizmetlerden en üst düzeyde yararlanmalarını sağlayan yapılar olacaklardır.

Bu aşamada ilk olarak temel web servisleri oluşturularak kullanıcıların bu web servisi kanalıyla Ws-I standartları çerçevesinde sistemi nasıl etkin bir şekilde kullanılabileceği üzerinde inceleme ve Microsoft Visual Studio 2008 [64] ile gerçekleştirim yapılmaktadır.

Belirlenen bir tarihteki arama işleminin gerçekleştirim basamaklarını incelemek bölüm başında daha açıklayıcı olabilecektir.

- Web Servisi istemcisi servisin WSDL dokümanlarını servis adresinden elde ederek gerekli vekil sınıfını kendi bünyesinde oluşturur. Bu vekil sınıf sayesinde uzak servis çağrılarını sanki yerel kod gibi kullanma şansına sahip olabilecektir.
- Sonrasında Servis metotları bu vekil sınıf üzerinden çalıştırılabilir. Ör: AramaYap2 metoduyla parametre olarak kullanıcı adı, şifre, aranacak numara, mesaj, gun, ay, yıl, saat, dakika gibi bilgiler gönderilerek servis işlemine başlanmış olunur.
- Servis tarafında öncelikli olarak gelen bilgilerden kullanıcı adı ve doğrulaması gerçekleştirilir. Bu işlem başarılıysa sonraki basamaklara başarısız olma durumunda ise metot sonucu olarak gerekli uyarı string veri türünde istemciye döndürülür.
- Telefon numarası uzunluğu ve geçerli tarih kontrolü yapılır.

- Bu işlemden sonra o arama için bir kod numarası oluşturularak, verilerin veritabanına işlenmesi için Gorusme sınıfının gorusmeGir metoduyla kayıt işlemine veriler gönderilir.
- Bu noktada bu veritabanı işlemleri veritabanındaki kayıtlı yordamlar üzerinden yağılacağı için kod üzerinde SQL sorgusu bulunmaz bağlantı adresleri, kayıtlı yordamlar, parametreler belirtilerek veritabanındaki kayıtlı yordam çağrılır.
- Eğer bu noktaya kadar bir problem olmamışsa sonraki basamak olarak metin bilgisinden ses sentezlemesi gerçekleştirilir. Buradaki işlemler hakkında 5.2 bölümde bilgiler verilmiştir. Fakat burada dikkat edilmesi gereken nokta istemcilerden internet üzerinden gelecek kodun çalıştırılmasının sistemde internet kullanıcısının yetkisinde olmasıdır. Bu durumda işlemin, gerekli bileşenleri kullanarak disk üzerinde belirlenen klasörde dosyayı oluşturma yetkisi dahilinde değildir. Bunu aşmak için Impersonation [45] olarak adlandırılan yöntemle işleme belirli bir sistem hesap yetkisi verilir ve bu yetkiyle ses sentezlemesi yapıldıktan sonra yetki geri alınır.
- İşlemler tamamlandıktan sonra istemciye görüşme kodu döndürülür.
- Görüşme kaydından sonra görüşmenin güncellemeleri, görüşme bilgilerinin alınması, görüşme kaydının silinmesi ve arama sonrasında tuş sonuçlarının alınması gibi işlemler servis metotlarıyla gerçekleşmektedir.
- Sonrasında ise arka plan Windows servisleri bizim belirleyeceğimiz aralıklarla veritabanında o an için geçerli arama olup olmadığını tarayacak ve arama varsa aramayla ilgili bilgileri VoIP bileşeninin uygun olduğu bir durumda alması için İşletim Sistemiyle gelen kuyruk yapısına aktaracaktır. Bu durum bölüm 5.3.4'de açıklanacaktır.
- Aynı şekilde VoIP arama işlemini gerçekleştirecek olan kodda müsait olduğunda kuyruktan sıradaki arama bilgilerini alacak ve aramayı gerçekleştirecektir. Bu durum bölüm 5.3.5'de detaylandırılacaktır
- Ayrıca tusSonuclari metoduyla da arama sonucunda aranan kişinin tuş yanıtları bu web servisi metoduyla servisten istenmektedir.

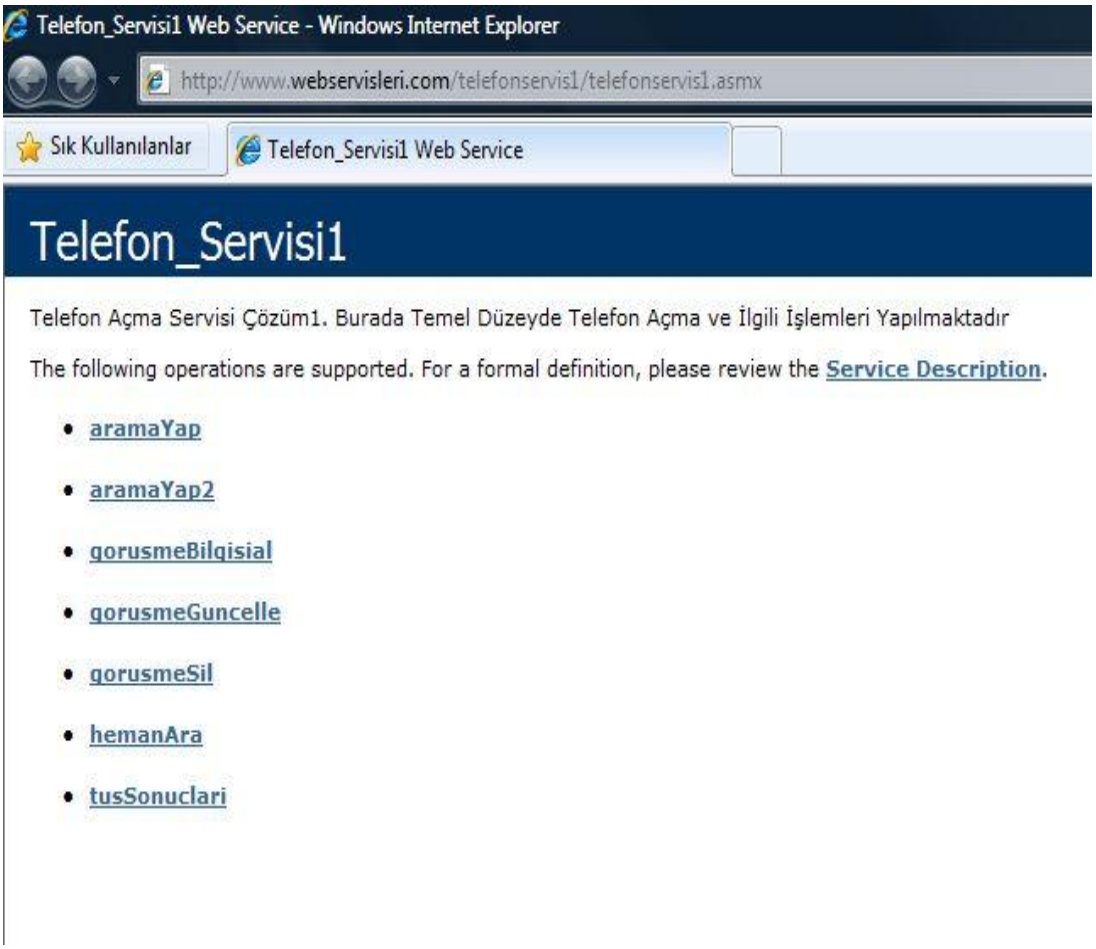


Şekil 5.16 Telefon Arama Servisinin Şemasal Gösterimi

5.3.1 Web Servisinin Yapısı

Temel işlemleri servis çözüm1 olarak adlandırdık. telefonservis1.asmx olarak adlandırdığımız bu servis içerisinde temel olarak altı metot bulunmakta ve bunların üç tanesi arama yapma ile ilgilidir. Şekilde Cozum1'in sınıf yapısı görülmektedir. Öncelikli olarak TelefonServis1sınıfı ve ilişkili olduğu sınıfları ile metotlarını incelersek:

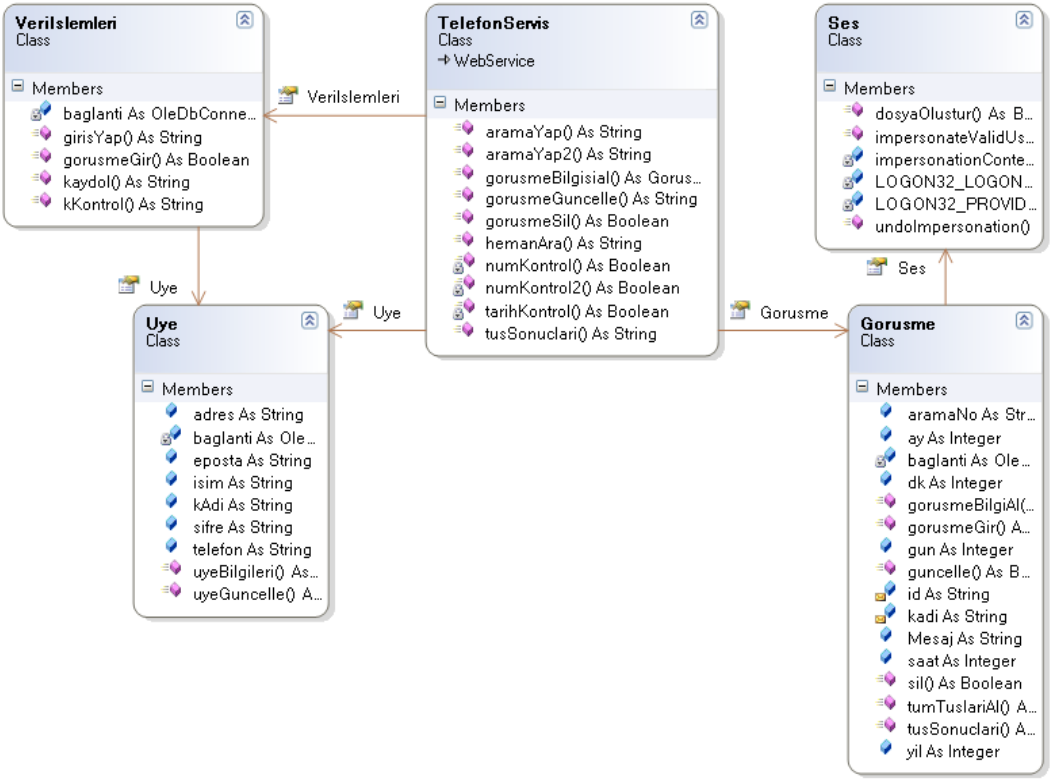
Telefonservis:



Şekil 5.17 Telefon_Servis1 metotlarının Gösterimi

- hemenAra metodu hızlı bir şekilde yapılacak olan aramalar içindir Veritabanına hızlı bir şekilde kayıt edilmektedir. Kayıt yapıldıktan sonra Arka Arama Kontrol servisleri tarafından kontrol edilip kuyruğa (MSMQ) hızlı bir şekilde aktarılarak arama işlemlerinin gerçekleştirilmesi sağlanmaktadır.

- aramaYap ve aramaYap2 metotları servis üzerinde aynı işlemi yapmakta fakat kullanıcıya metotları farklı bir şekilde kullanım seçeneği sunmaktadır. Kullanıcı aramaYap servisinde kullanıcı adı ve şifresinin yanında parametre olarak Gorusme sınıfını göndermektedir. Bu sayede istemci tarafında Gorusme sınıfı oluşturtabilmekte ve bunlar üzerinde kullanıcılar işlemlerini tanımlayabilmektedir. Veya aramaYap2 metodunda istemciler doğrudan görüşme bilgilerini parametre olarak göndermektedir.
- Sisteme gönderilen görüşme kayıtlarının alınması için gorusmeBilgisiAl metodu kullanılmaktadır. Burada aramaYap metotlarından geri dönen o görüşme kodu parametre olarak gönderilmekte ve Gorusme sınıfı istemciye geri döndürülmektedir. Tabi bu nokta sınıf belirli bir serializationdan geçerek sınıf içerisinden gerekli bölümler iletilir. gorusmeSil metodu ile de görüşme kodu girilerek görüşmenin veritabanı kayıtlarının silinmesi sağlanmaktadır. Benzer şekilde adından da anlaşılacağı gibi gorusmeGuncelle görüşme bilgilerini güncellemek içindir.



Şekil 5.18 TelefonServis içerisinde kullanılan Sınıf Diyagramları

Gorusme Sınıfı:

Web Servisi metotlarına gelen çoğu işlem bu sınıf metotlarına aktarılarak gerekli servis veritabanı ve dosya işlemleri gerçekleştirilmektedir. Ayrıca görüşme bilgileri de tekrar bu sınıf içerisinde tutulmakta ve kendi metotları üzerinden işlem yapmaktadır. Buradaki temel metotlar gorsumeBilgiAl, gorusmeGir, sil ve guncelle metotlarıyla gerekli veritabanı işlemleri gerçekleştirilir. Ayrıca Görüşme girilirken aynı görüşme koduna uygun olarak ses dosyasının sistemde metin üzerinden ses sentezlenmesi ve sentezlenen sesin disk üzerinde Gorusmeler klasöründe kaydedilmesinde Ses sınıfının dosyaOlustur metodu çağrılmaktadır.

Ayrıca sınıfın bazı bölümleri servis üstünden aktarılır. Belirli bir serileştirme işleminden sonra XML yapısına dönüştürülür ve WSDL içerisinde tanımlanmaktadır. Ve istemci tarafında oluşturulan vekil sınıfla beraber tanımlanmakta ve istemci içerisinde kullanılmaktadır.

VeriIslemleri Sınıfı:

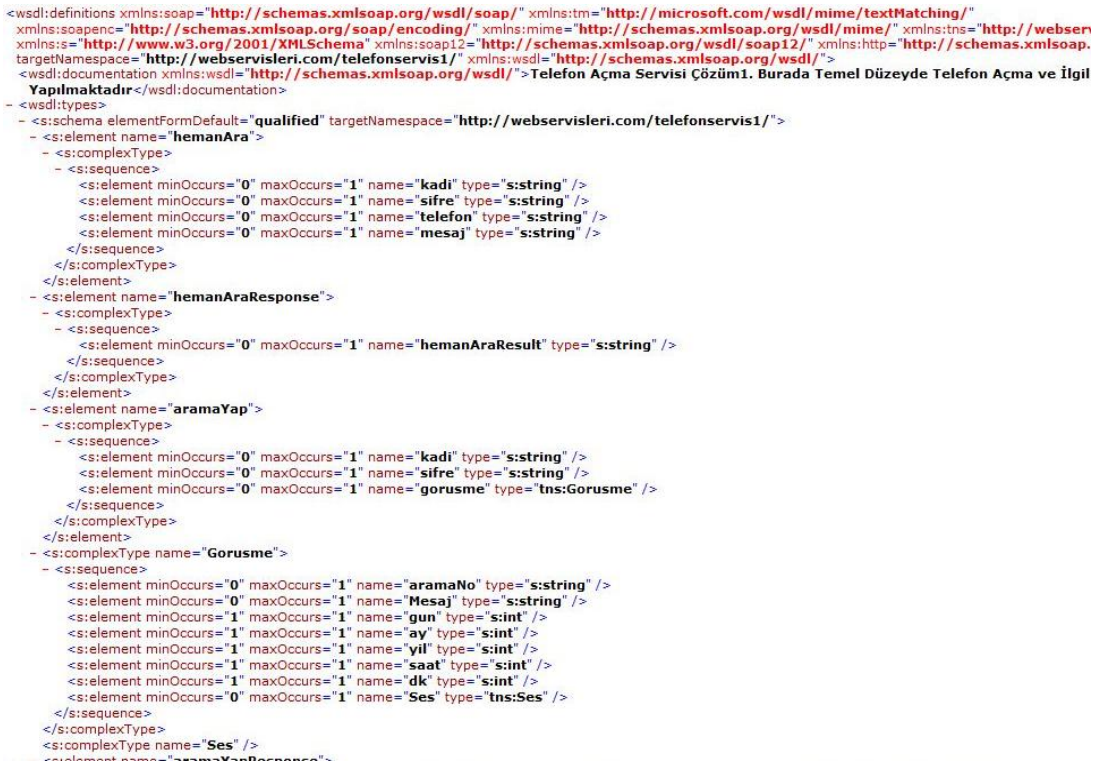
Bu sınıf temel olarak sistemin web ara yüzünü oluşturan uygulamada kullanılmasının yanında birlikte web Servisi içerisinde temel görevi metotlara parametre olarak gelen kullanıcı adı ve şifresinin giris metodu ile veri tabanından kontrol edilmesini ve sistemde bu kullanıcının kayıtlı olması durumunda diğer işlemelere devam edilmesini servis içerisinde sağlamaktadır.

Uye Sınıfı:

Uye sınıfı kullanıcı bilgilerinin servis içerisinde kullanımı ve güncellenmesi gibi temel üye bilgi işlemlerini yönetir.

Ses Sınıfı:

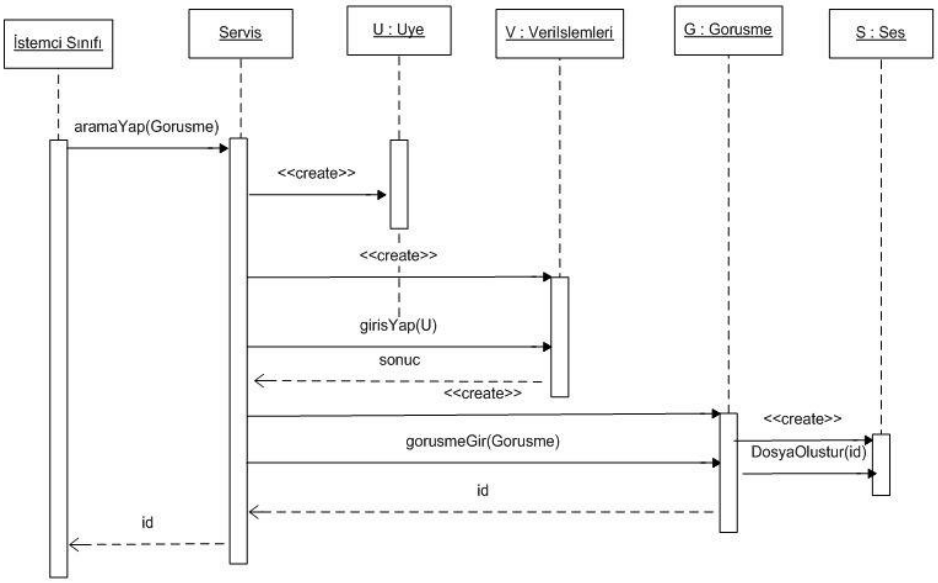
Bu sınıf içerisinde ses sentezlemesiyle beraber, ses dosyasının disk üzerinde oluşturulması gerçekleştirilmektedir. Bu noktada bu bölümün başında bahsettiğimiz Impersonation kavramıyla o an kullanıcıya ses sentezlemesi gerçekleştiren ve disk üzerinde belirlenen bir klasöre yazma izni veren kod uygulaması vardır. Buradaki kod içerisinde sistem kütüphaneleri kullanılmış ve. NET Framework dışarısına çıkılmıştır. Kod örneği Microsoft'un bilgi tabanlarında mevcuttur [45].



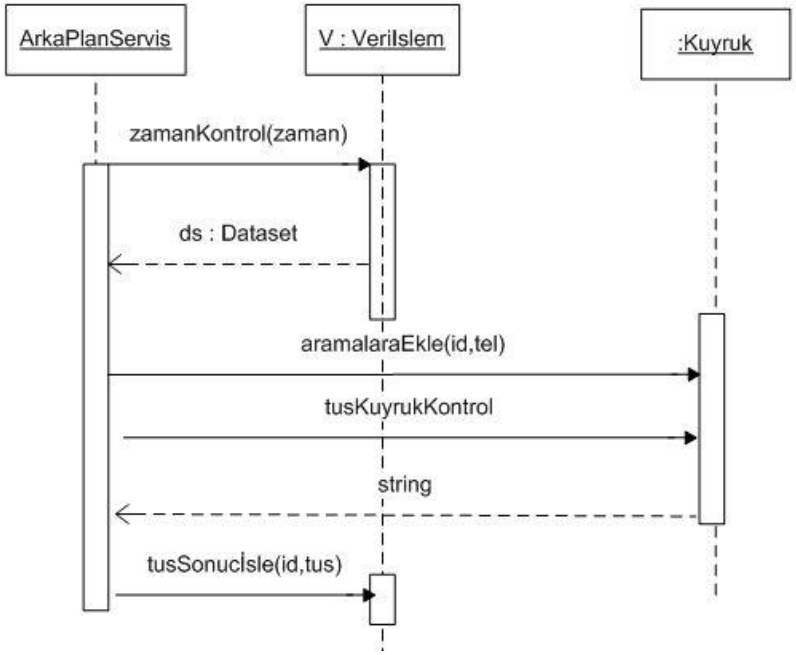
Şekil 5.19 Servis WSDL Dokümanının Başlangıç Bölümü

Şekil 5.20 Servis WSDL Dokümanın Bitiş Bölümü

5.3.2 Web Servisi Üzerinden Örnek UML Sıra Diyagramı

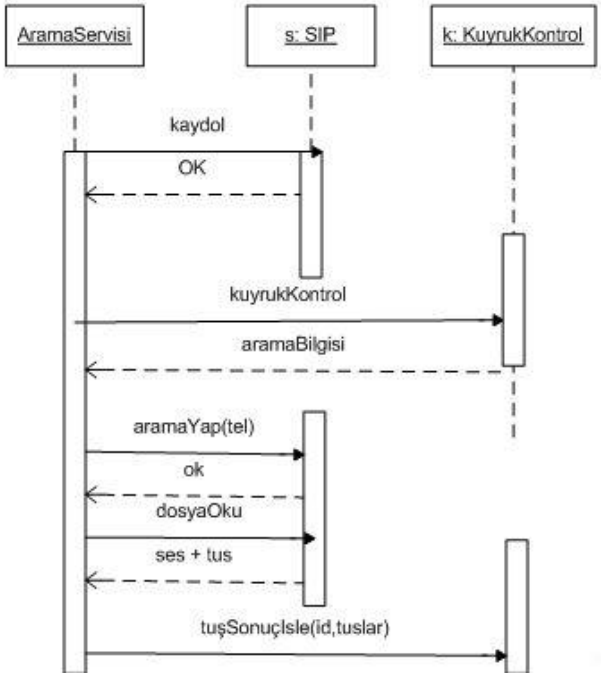


Şekil 5.21 aramaYap Servis Metoduyla Başlayan İşlemlerin Sıra Diyagramı



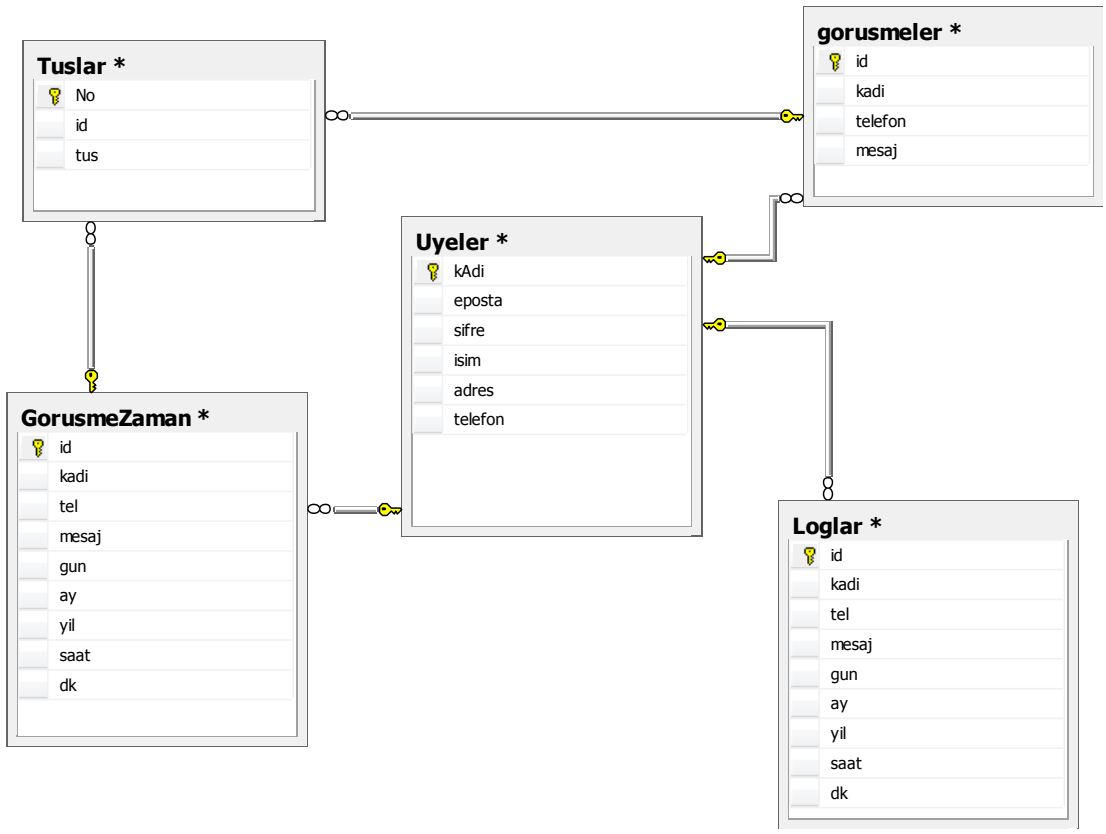
Şekil 5.22 Arka Plan Servisleri Veritabanı-Kuyruk Kontrol İşlemleri

Şekil **5.23**'daki işlemler belirleyeceğimiz belirli zaman aralıklarında gerçekleşmektedir.



Şekil 5.23 Arama Servis UML Sıra Diyagramı

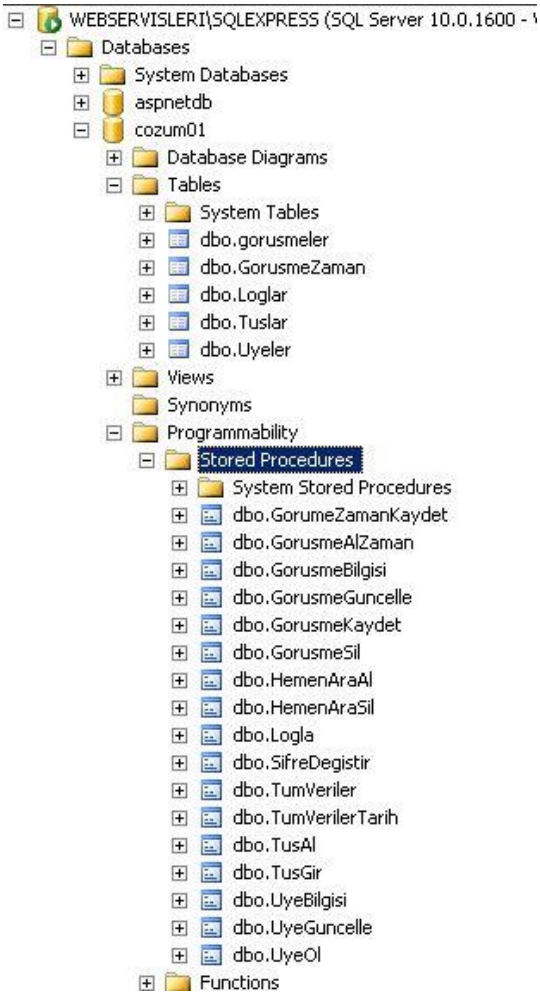
5.3.3 Veri Tabanı İşlemleri ve Tablolar



Şekil 5.24 Tablolar ve İlişkiler

Temel çözüm kapsamında sistemin çalışmasını sağlayabilecek tablolar Şekil 5.24'daki gibi en temel düzeyde olabilir. Gerek web ara yüzü içerisinden gerekse web servisinden kullanılan Gorusme ve VeriIslem bileşenleri işlemlerinde veri tabanında bulunan kayıtlı yordamlar aracılığıyla SQL işlemlerini gerçekleştirmektedir. Bu sayede veritabanı işlemlerinde tekrar kullanılabilirlik, güvenlik ve performans konularında belirli kazanımlar elde edilmiştir.

Veritabanı sunucusu olarak SQL Server 2008 kullanılmıştır. Bu kayıtlı yordamlar ürün içerisinde bulunan editör aracılığıyla oluşturulmuştur.



Şekil 5.25 Veritabanındaki Mevcut Kayıtlı Yordamlar

Şekil 5.25'de TelefonServis1 için oluşturulan veritabanı tablo ve kayıtlı yordamları görülmektedir.

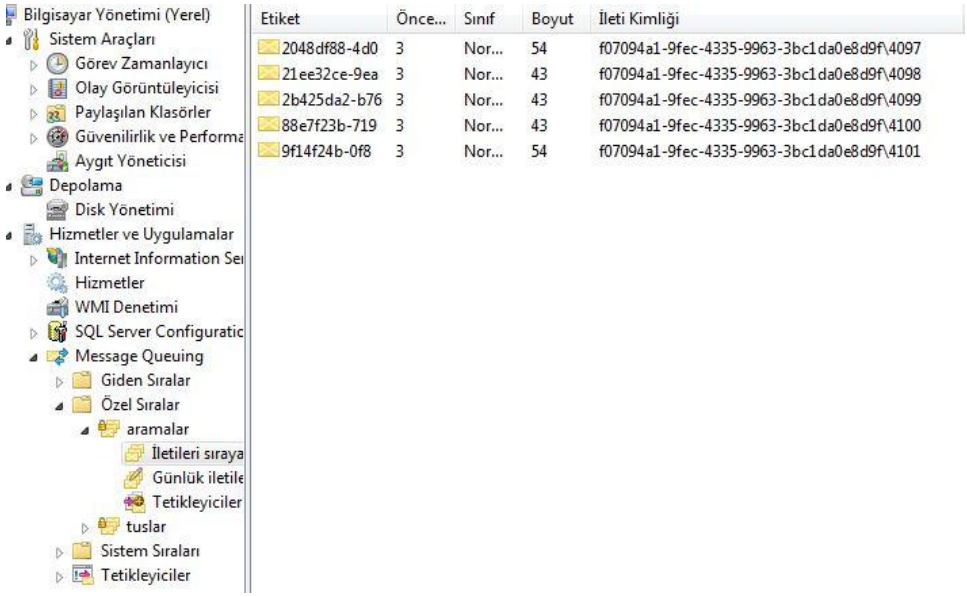
5.3.4 Arka Plan Windows Servisleri ve Kuyruk İşlemleri

Projemiz içerisinde görüşmeler belirlenen zamanlarda gerçekleştirilmektedir. Bunun için arka planda çalışan servisler o zaman dilimindeki görüşmeleri kontrol ederek veri tabanından görüşme bilgilerini alıp, bunları ses iletimini gerçekleştirecek olan bileşenlere iletebilmelidir. Arama yapılan kanallar sınırlı sayıda olacağından görüşme zamanında arama yapılamayacak ve belirli bir süre her görüşme kaydı kendi sırasını beklemek zorunda kalacaktır. Bunun için en uygun çözümlerden birisi

kuyruk yapısının kullanılması düşünülmüş ve işletim sistemi içerisinde bulunan MSMQ [46] ile çözüm geliştirmek için en uygun yapı olarak görülmüştür. Bu sayede arka plan servislerinin yeniden başlamasında ve sistemin çeşitli durumlarda kendini yenilemesi veya istenildiğinde yeniden başlatmalarda veriler kaybolmayarak kuyruk içerisinde güvenli bir şekilde depolanabilmektedir.

Şekil 5.26 Kuyruk Servis İşlemlerinin Gösterimi

Bu servisin görevi ayrıca ses iletim servislerden kuyruğa gelen tuş tonu yanıtlarının da alınmasıdır. Kuyruktan önceden belirlenen zaman aralıklarıyla tuş bilgileri alınıp veritabanındaki Tuslar tablosuna işlenir. Ayrıca arama yapan uygulama veritabanı işlemleriyle uğraşmamakta ve veritabanından daha hafif bir bağlantı sağlayan kuyruk yapısını kullanmaktadır. Böylelikle tuş bilgisi girilirken verilerin işlenmesinin denetlenmesi gibi konular arama servisinin kapsamında olmayacak ve yükü daha da azalacaktır.



Şekil 5.27 Kuyruk İçerisinde Bulunan Arama Sırasını Bekleyen Mesajlar

Gerçekleştirim içerisinde gerek arka plan servislerinin gerek arama servislerinin doğrudan kuyruklarla ilişkili olması ve bu bağlantının veritabanı bağlantısına göre daha hafif olması performansı arttıracaktır [46]. Ayrıca Ölçekleme açısından sisteme farklı arama işlemini gerçekleştiren sunucular rahatlıkla eklenebilmektedir. Bu sunucular veritabanı işlemleriyle uğraşmadan doğrudan kuyrukta bekleyen aramayı alarak belirli bir sıra düzeninde arama işlemini gerçekleştirebilirler.

5.3.5 Arama Servis Uygulaması

Arama servis uygulamasında bir görüşme için gerçekleştirilen işlem basamaklarını maddeler halinde incelersek:

- Her uygulama kendisi için özel olarak belirtilen yapılandırma bilgileriyle başlatılır. Bir makineden aynı anda birden fazla görüşme yapılabileceği için

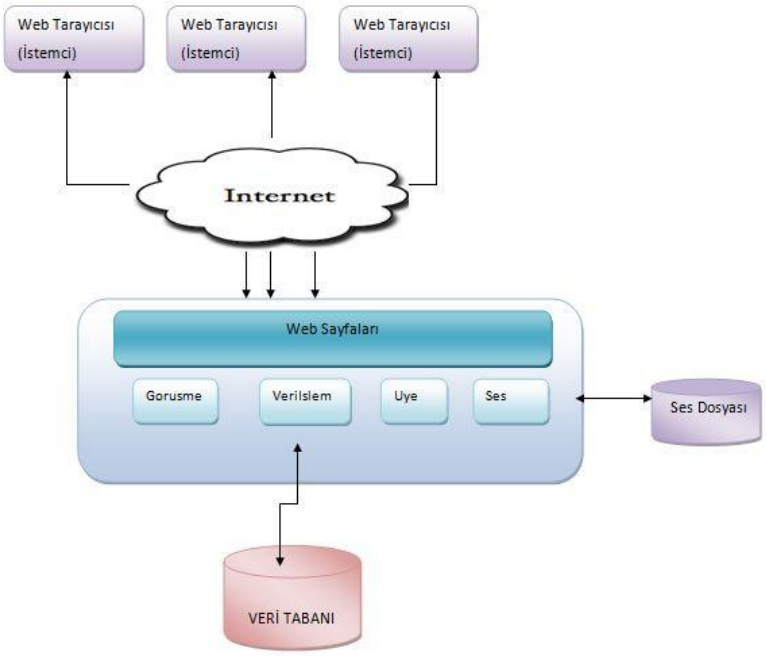
farklı görüşmeleri gerçekleştiren uygulamalar farklı bağlantı noktası adreslerine sahip olmalıdır.

- Uygulama başlatıldıktan sonra .Net framework içerisinde bulunan Timer nesnesi aracılığıyla MSMQ içerisindeki aramalar kuyruğunda mesaj olup olmadığını kontrol eder. Eğer mesaj varsa sıradaki mesajı alır ve mesaj içeriğine göre arama ile ilgili gerekli bilgileri aramayı gerçekleştirecek olan Fonksiyona iletir. Ayrıca kuyruğu belirli aralıklarla denetleyen zamanlayıcı devre dışı bırakılır.
- Arama Fonksiyonunda telefon numara bilgisine göre ve yapılandırmadaki ayarlara göre SIP mesajları oluşturulur. Bu mesajlar gene belirtilen bağlantı noktaları üzerinden VoIP hizmet sağlayıcıya iletilir. Böylelikle görüşme için ilk adım yapılır.
- 200 Tamam kodu geldikten sonra ve ACK ile bildirim gönderildikten sonra bağlantı RTP kanalları üzerinden iletim için hazır durumdadır. Bu kanallar üzerinden iletim sağlanıp görüşmeye başlandıktan sonraki ilk adım olarak RTP kanalı üzerinden ses verilerinin gönderimi yapılmaktadır. Ses bilgilerinin bu kanal üzerinden gönderimi tamamlandıktan sonra aranan kişiden bir yanıt gelmesi için belirli bir zaman beklenir.
- Bu zamanlamayı sağlamak için ikinci bir Timer nesnesi kullanılmıştır. Bu nesne aktif duruma geldikten sonra kullanıcıdan gelen yanıtlar bu süre içerisinde alınarak kuyruğa aktarılır. Süre dolunca bağlantı kesilir.
- Görüşme tamamlandıktan sonra Kuyruğun Arama için belirli aralıklarla denetlenmesini sağlayan Timer nesnesi tekrar aktif hale getirilir.
- Tuş sonuçları işlem performansı açısından veritabanı yerine kuyruğa gönderilir. Arka plan servisleri bunları kuyruktan alarak veritabanına işleyecektir.

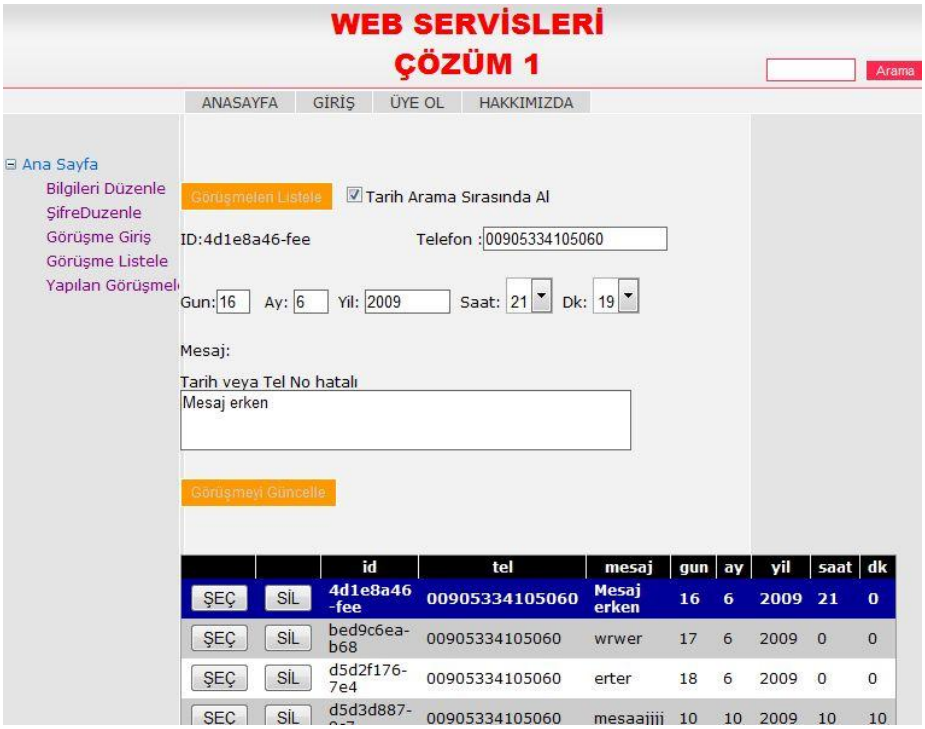
5.3.6 Web Sayfaları ve Yönetimi

Kullanıcılar ilk başta web üzerinden kaydolarak sisteme dahil olacaklardır. Ayrıca işlemlerini web üzerinden takip edip yönetebileceklerdir. Bu sayede servis kullanımı üzerinde genel bir kontrol sağlanacaktır. Ayrıca servis istemcileri olmadan da web sayfaları kullanıcılara işlem yapma yetkisi verebilecektir.

- default.aspx (Ana sayfa)
- giris.aspx (Sisteme Üye Girişi)
- uyeol.aspx (Üye Kayıt Yapılandırması)
- hakkimizda.aspx (Servisler ve Yapılanlar Hakkında Genel Bilgi)
- Uyeyonet.aspx (Üye Yönetim- Bilgi Güncelleme Sayfası)
- sifre.aspx (Şifre Yönetim ve değiştirme sayfası)
- gorusmegiris.aspx (Web Ara yüzünden Giriş Sayfası)
- gorusmelistele.aspx (Yapılacak Olan Görüşmelerin listelenmesi ve Düzenlenmesi)
- yapilangorusmeler.aspx (görüşme kayıtlarının tutulduğu ve sonuçların gerek ses dosyası gerek tuş sonuçları olarak elde edildiği sayfa)

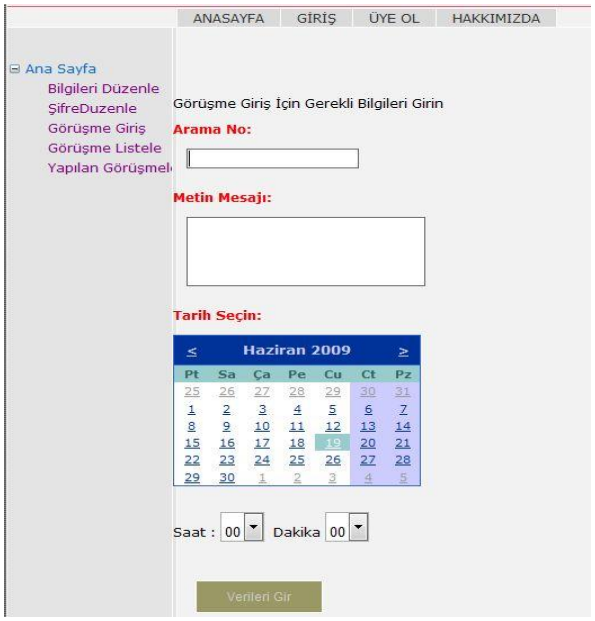


Şekil 5.28 Web Arabirim İşleyişinin Gösterimi



Şekil 5.29 Görüşme Listeleme Sayfası, Yapılacak Görüşmelerin Listesi

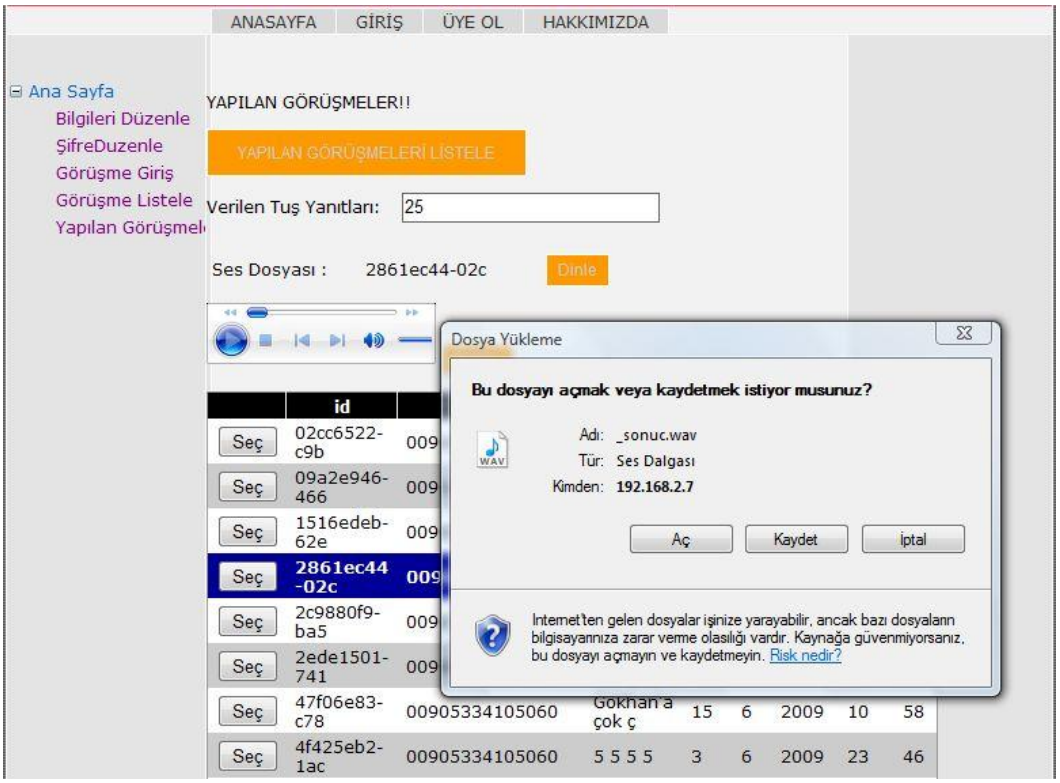
Şekil 5.29 üzerinde o kullanıcının sisteme girdiği mevcut aramaların listesi görülmektedir. Aramalar istendiği takdirde tarih kayıt ediliş sırası ve arama sırasına göre ayrı ayrı alınabilir. Bu arabirim üzerinden listedeki görüşmeler seçilerek daha detaylı bir inceleme yapılabilir ve istenirse sil seçeneğiyle görüşme iptali gerçekleştirilir. Ayrıca incelenen görüşmeler üzerinde arama öncesinde güncellemeler de bu ara yüz üzerinden yapılabilmektedir.



Şekil 5.30 Görüşme Giriş Sayfası

Şekil 5.30'daki arabirim üzerinden de görüşme kayıt işlemleri gerçekleştirilebilir. Şekil 5.31'de yapılan görüşmelerin listesi vardır. Bu ekran üzerinden kullanıcılar gerçekleştirilen aramaları görebilir. Aranan kişinin aramaya cevap olarak verdiği tuş yanıtları görebilir ve aynı zamanda karşı tarafın yanıtını da sesli olarak buradan dinleyebilir veya ses dosyası olarak indirebilir.

Tüm bu sayfalara erişim ana sayfa üzerinden başarılı kullanıcı adı ve şifre girişiyle sağlanabilmektedir. Adres çubuğundan yönetim sayfası adresleri yazılıp girilse dahi sistem bunu kabul etmeyecek kullanıcı giriş sayfasına yönlendirme yapacaktır.

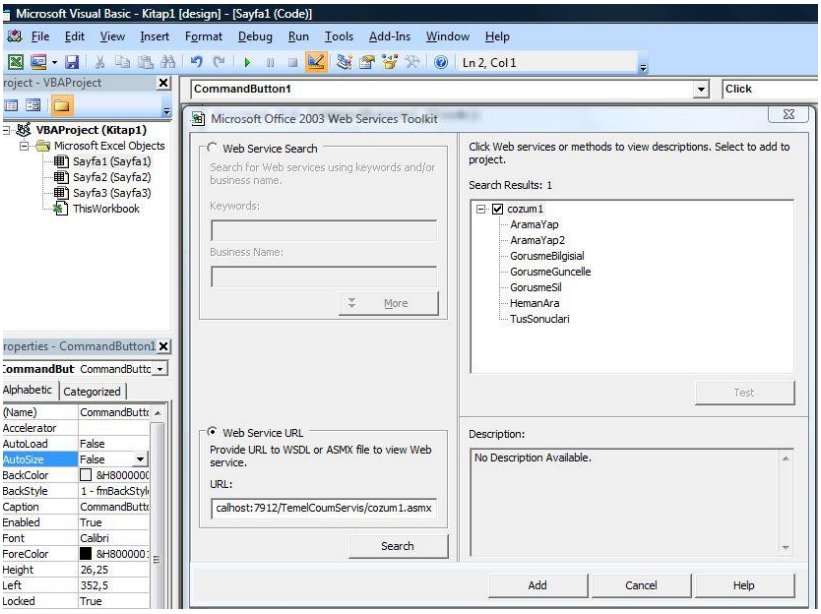


Şekil 5.31 Yapılan Görüşmelerin Listesi Tuş Sonuçlarının Gösterilmesi ve Görüşme Kaydının İndirilmesi

5.3.7 Web Servisi İstemcileri

Kullanıcılar web ara yüzü dışında işlemlerini takip etmek ve gerçekleştirmek için web servisi istemcilerini kullanabilirler. Bu sayede uygulamalarına servisin yeteneklerini dahil ederek servis işlemlerini kodlarının bir parçası haline getirebilirler. Böylelikle uygulama geliştiriciler için bu yeteneklerin kendi uygulamalarına hatta dokümanlarına eklenmesi mümkün olacaktır. Temel Servisler WS-I standartlarına dayalı olarak gerçekleştirildiği için istenilen programlama dilinden veya bunu destekleyen uygulamalardan bu servislere erişim mümkün olabilmektedir. Burada istemci örneğinde klasik .net veya java uygulamalarından farklı olarak Microsoft Excel içerisinden web servisine erişim sağlanmaktadır. Bu sayede Excel belgemize telefon açma ve yanıtlarını alma gibi karışık bir olay birkaç satırlık makro kodlarıyla eklemek mümkün olabilmektedir.

Bu işlemler esnasında Microsoft Excel içerisine öncelikli olarak Office dokümanları içerisinden web servislerine erişim sağlayan Office Web Services Toolkit projemize eklenmiştir (Şekil 5.32). Bu araç XP sürümü için yazılmış olmasına rağmen 2007 içerisinde de sorunsuz çalıştığı gözlemlenmiştir.



Şekil 5.32 Microsoft Office Web Services Toolkit [53]

VBA editöründe Tools ve sonrasında Web Service References Seçeneklerinden gerekli servis URL adresi girilerek servis kodu Excel dokümanına dahil edilir. Sonrasında geliştirici araç seçeneklerinden servis ile etkileşimi sağlayan Düğmeler çalışma sayfasına yerleştirilerek kullanıcının işlem yapmasına olanak tanınır.

	A	B	C	D	E	F	G	H	I	J
1	Aramaları Yap		Gerçekleşen Arama Tuş Yanıtları							
2										
3	Aranacak Kişi:	Telefon Numarası:	Mesaj	Gun	Ay	Yıl	Saat	Dk	id	TuşYanıtları
4	Gökhan	5334105060	birinci önemli mesaj	17	4	2009	14	40	248eef2c-9c8	234
5										
6										
7										

Şekil 5.33 Excel İçerisinde Verilerin İşlenmesi ve Tuş Yanıtlarının Alınması

5.4 SOA ile Çözüm Gerçekleştirimi

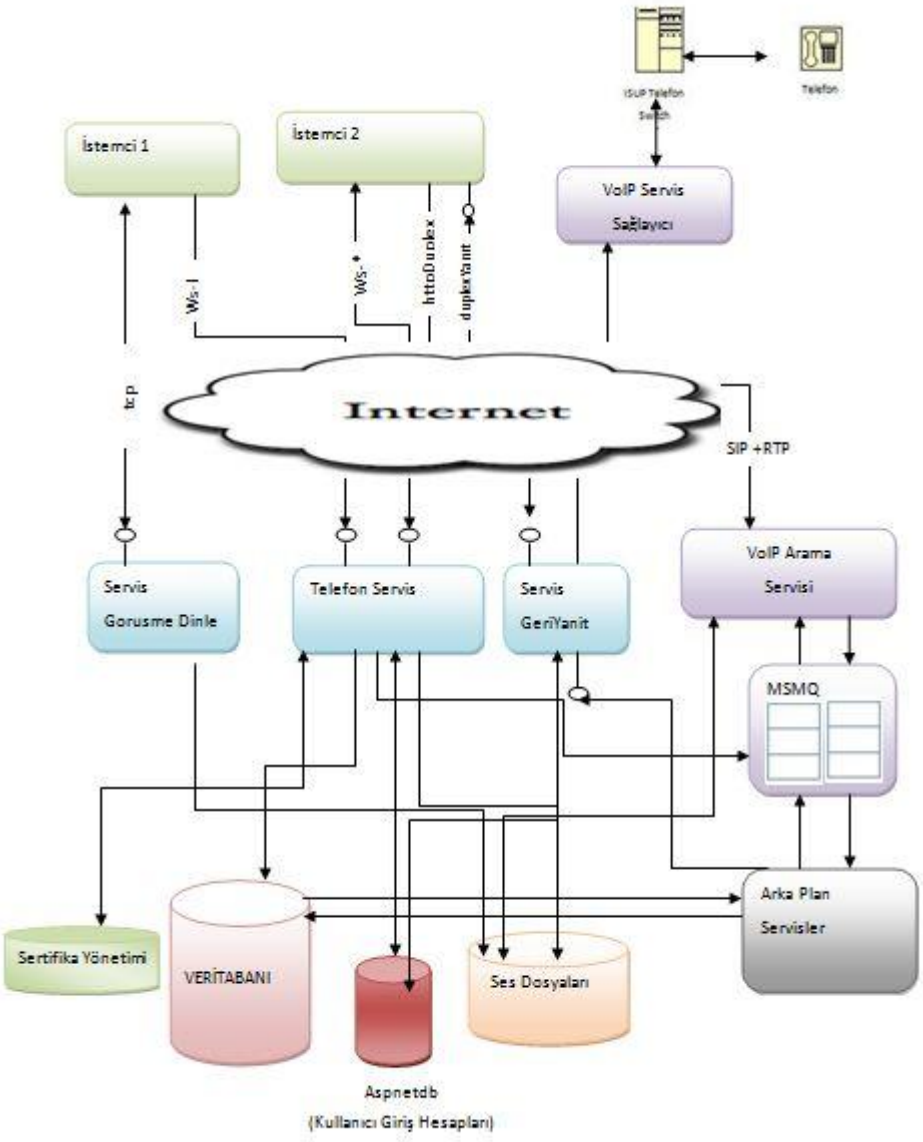
Bu bölümde 5.3 numaralı telefonservis1 çözümünün üzerine sistemi servis yönelimi mimaride tasarlanarak incelenmiştir. Servis yönelimli mimari kazanımları ve prensipleri üçüncü bölümde detaylı olarak incelenmiştir. Bu bölümde onun hem uygulamasını ve hem de temel çözüm üzerine getirdiği kazanımları inceleyeceğiz. Bu sistem Microsoft Windows Communication Foundation (WCF) [49] kullanarak tasarlanmıştır. Ve de bu yapı içerisinde gerçekleştirilen servisler, servis yönelimli mimarinin 4 temel özelliğini çok az bir düzenlemeyle, rahatça barındırabilmektir.

Sistem servis yönelimli mimaride tasarlanan ve geçen bölümde gerçekleştirdiğimiz servislere ek olarak elde edilecek olan kazanımları maddeler halinde incelersek;

- Servis Yönelimli Mimari prensiplerinde tasarlanarak bileşenlerden elde edilen verim üst düzeyde olmuştur. Bu sayede servis bileşenleri farklı mimariler içerisine daha rahat uyum sağlayabileceklerdir.
- Ws-Security Protokolünün kullanımıyla beraber serviste kullanıcı girişleri ve verilerin gerekli bölümlerinin şifrelenmesi sağlanarak güvenlik ve güvenli haberleşme daha da gelişmiştir. Örneğin sadece mesajların şifrelenmesiyle telefon numaralarına göre ara kademelerde yönlendiriciler kullanabilecektir.
- Ws-Reliablemessaging protokolünün kullanımıyla beraber haberleşme sırasında çıkabilecek verilerin http üzerinden iletim esnasında kaybı ortadan kalkacaktır. Uzun mesajlarda veya ağ trafiğinin karmaşık olduğu servis işlemlerinde kullanışlı bir çözüm olarak servisimize eklenmiştir.
- MTOM protokolü sayesinde doğrudan ikili dosyaların servis içerisinden transferi mümkün olabilecektir. Örneğin metinden ses dönüşümü yerine servise doğrudan istemci tarafında hazırlanan bir ses dosyası parametre olarak eklenebilecek ve aramalarda kullanıcının gönderdiği ses dosyası telefona aktarılabilecektir. Ayrıca istemciye servis yanıtı olarak da görüşme sonucu ses dosyası istendiğinde aktarılacak; bu sayede kullanıcı gelen yanıtları web ara yüzüne gerek kalmadan dinleyip kendi kodu içine entegre edebilecektir.

- Bu servis yapısı WS-* protokollerini destekleyebildiği gibi servisi sadece WS-I standartlarında servisi kullanmak isteyenlere de farklı servis oluşturmadan erişim izni verebilecektir. Böylelikle tek bir servis içerisinde farklı bitiş noktaları (endpoint) tanımlayarak farklı istemci profillerine erişim imkanı verilmiştir.
- Duplex adı verilen ve verilerin çift yönlü iletimi sağlayan yapı da mimari içerisinde kullanılabilmektedir. Duplex sayesinde servis istemci yanıtlarına cevap veren bir yapıdan farklı olarak kendisi de gerekli durumlarda istemciye mesaj gönderebilecektir. Örneğin belirli bir zamanda arama tamamlandığında veya arama sonucunda tuş değerleri istenilen değerler olduğunda istemci uygulamaya durumu bildiren mesajı servis kendisi gönderebilecektir.
- Bu tür servisler kurumsal yapılarda LAN ortamında kullanılmak istenirse gene servis kod yapısına dokunmadan ve ek servis geliştirmeden ek bağlantı noktaları oluşturularak kullanılacaktır. HTTP yapısının yanında TCP bağlantı noktası da rahatlıkla eklenebilecektir. Ayrıca ek bir bağlantı da oluşturarak kuyruk hizmetlerine erişimde ek bir katmana gerek kalmadan doğrudan MSMQ servisine erişim sağlanabilir. Böylelikle LAN ortamında ses hizmetlerini kullanan kullanıcılar performans artışı elde edecektir.
- WS-Addressing ve MEX gibi protokollerle beraber servis kendisi ve kuralları hakkında daha detaylı bilgi verebilecek ayrıca gerekli durumlarda farklı servislere de adresleme yapabilecektir. Yük dengeleyici veya ses hizmetlerinin daha dağıtık bir yapıda olması gibi durumlarda kullanım imkanı daha kolay olabilecektir.
- Servislerin yayınlama ortamları temel servislere göre daha esnektir. Temel servisler yayınlama için bir web sunucusu gerektirirken WCF ile geliştirilen servislerde bu yayınlama için ayrı bir web sunucusuna gerek kalmadan yayınlama yapılabilecektir. Servisler kendi kendilerini yayınlayarak servis yönelimli mimarinin temel prensiplerinden otonomiyi gerçekleştirmiş olacak ve ses hizmetlerinin kullanımı yayınlama konusunda daha da esnek bir yapı sunacaktır.

- Servis seviyesi anlaşmalara uygun bir şekilde servis belirli kullanıcılara ayrı bir instance olarak sunulabilecektir.



Şekil 5.34 Genel Servislerin Sistem Mimarisinin Gösterimi

Şekil 5.34 üzerinde sistemin genel mimari gösterimi bulunmaktadır. Bu sistemin yapısını madde madde incelersek;

- Servis yönelimli mimaride servislerin gerçekleştiriminde öncelikli olarak geçen bölümde gerçekleştirimini incelediğimiz telefonservis1 servisinin metotlarını servis yönelimli mimariye taşıyarak aynı mimariyi servis yönelimli olarak gerçekleştirmekteyiz.
- Bu servisten farklı olarak servis görüşmelerini dinleme için ayrı, geri yanıtların alınması için de ayrı bir servis oluşturulmaktadır. Bu servislerin diğer Telefon Servisinin dışında tutulmasının nedeni, sadece temel işlemleri kullanmak isteyen servislere istemci tarafında ek bir yük getirmemek ve daha hafif bir vekil sınıf oluşturmaktır.
- Telefon Servisi'nin iki tane bitiş noktası vardır. Bu sayede hem WS-I uyumlu hem de WS-* protokollerini kullanabilecek olan istemcilere hizmet verebilir. WS-I olarak kullanıldığında yetenekleri bir önceki bölümde incelenen TelefonServis1 seviyesinde olacaktır. Bu bölümde elde edilen kazanımları sağlamak için WS-* protokollerinin kullanılması gerekmektedir. Ws-* servisleri çoğu web servisleri geliştirme platformları tarafından desteklenmektedir.
- Sistemde mesajların şifrelenmesi ve imzalanması için sistemdeki sertifikaları kullanabilir. Kullanıcı girişlerinde web servislerinin doğrudan işlem yapabilmeleri için WCF ile tümleşik olan aspnetdb kullanıcı veritabanı kullanılmaktadır. Bu işlemler 5.4.7. numaralı bölümde incelenmektedir.
- Geçen bölümde incelediğimiz servisten farklı olarak servislerin belirli bir işlem sonucunda kullanıcılara, kullanıcının da servisi tetiklemesine gerek kalmadan istemciye bilgi göndermesi gerçekleşebilir. Bölüm 5.4.3. de detaylı bir biçimde incelenmektedir.
- Geriye kalan Arama ve Arka plan servislerinin çalışması diğer servisle aynı özellikleri göstermektedir.

Bu metotların yanında konu bölüm içerisinde servis yönelimli mimari ve WS-* protokollerinin getirdiği yenilikleri kullanacak yeni metotlar da servislerimize eklenmektedir.

```
Imports System.ServiceModel

<ServiceContract()> Public Interface Itelefon

    <OperationContract()> Function aramaYap2(ByVal kadi As String, ByVal tel As String, B

    <OperationContract()> Function aramaYap(ByVal kAdi As String, ByVal g As Gorusme) As

    <OperationContract()> Function gorusmeBilgisial(ByVal kadi As String, ByVal id As Str

    <OperationContract()> Function gorusmeSil(ByVal kadi As String, ByVal id As String) A

    <OperationContract()> Function gorusmeGuncelle(ByVal id As String, ByVal kadi As Stri

    <OperationContract()> Function tusSonuclari(ByVal kadi As String, ByVal id As String)

End Interface

<DataContract()> Public Class Gorusme
    <DataMember()> Public aramaNo As String
    <DataMember()> Public mesaj As String
    <DataMember()> Public gun As Integer
    <DataMember()> Public ay As Integer
    <DataMember()> Public yil As Integer
    <DataMember()> Public saat As Integer
    <DataMember()> Public dk As Integer
End Class

<ServiceContract()> _
Public Interface ItelefonBasic
    <OperationContract()> Function aramaYapB2(ByVal kadi As String, ByVal sifre As Strinç
```

Şekil 5.35 Servis Kontratlarını Oluşturan Ara yüzlerin Tanımlanması

Şekil 5.35'de Itelefon ara yüzü temel olarak Telefon servisinin WS-* özelliklerini kullanmak isteyecek olan istemciler içindir. Benzer şekilde aynı servisin WS-I uyumlu versiyonu da ItelefonBasic arayüzünden oluşacak kontrat üzerinden tanımlanabilir böylelikle tek bir servis üzerinden farklı bağlantılar sağlanabilmektedir. Gorusme sınıfı bir DataDontract tipi yani XML serileştirmesi sonucunda oluşacak bir karmaşık tip tanımlamasıdır. Bu tanımlama servis metotları içerisinde kullanılabilecektir.

Yapılan akademik ve sektörel projeler [47][48] incelendiğinde çalışmaların temel düzeyde bahsettiğimiz seviyede olduğu ve birçoğunun bu bölümde bahsettiğimiz getirileri kapsamadığı görülmüştür. Burada da sistemimizin diğer yapılan çalışmalardan farkı görülmektedir. Bu bölümün sonraki kısımlarında bu özelliklerin nasıl ve hangi protokoller yardımıyla kazandırıldığına değinilecek ve son olarak servis yönelim prensiplerini nasıl kapsadığından bahsedilecektir.

5.4.1 Ws-Reliablemessaging ile Mesajların Eksiksiz Taşınması

Http protokolü yapısı gereği kaybolan verilerin tekrar gönderimini desteklememektedir. Bölüm 3.3.7. de Ws-Reliablemessaging bölümünde anlatıldığı gibi mesajların eksiksiz olarak ve gönderildiği sırada iletimi için bu protokol rahatlıkla kullanılabilir. Bunun için servis üzerinde yapılacak olan yapılandırma ayarları endpoint yani bağlantı noktası düzeyinde yapılmaktadır.

```
<wsHttpBinding>
 <binding name="RMWsHttp">
      reliableSession enabled="true " ordered="true"
      inactivityTimeout="00:10:00"/>
 </binding> >
</wsHttpBinding>
```

Şekil 5.36 WS-ReliableMessaging Yapılandırma Ayarları

Şekil 5.36'de verilen yapılandırmada mesaj sıralamasının yapılacağını ve zaman aşımı süreleri belirtilmiştir. Diğer yapılandırma ayarları ise onay mesaj gönderimi varsayılan olarak 0.2 sn, kuyruk tampon bölgesi büyüklüğü 8 mesaj, tekrar gönderim miktarı 8 adet olarak belirlenmiştir. Gerekirse özel bağlantı yapılandırması tasarlanarak bu varsayılanlar üzerinde değişiklikler ve protokolün hangi sürümünün kullanılacağı belirtilir. TCP üzerinde işlemler protokol düzeyinde yapılacağından bu protokolün kullanılmasına gerek yoktur. Fakat TCP düzeyinde kullanım sadece. NET uygulamaları için geçerlidir ve güvenlik duvarı ayarları buna göre yapılandırılmalıdır.

5.4.2 WS-I Protokol Uyumluluğu

5.3. numaralı bölümde oluşturulan web servisleri WS-I uyumluluğunda tasarlanmaktadır. Servisi kullanan istemciler yeni ek özellikleri kullanmak istemeden temel özellikleri kullanmak isterlerse farklı bir servise bağlanmadan aynı servis üzerinden temel düzeyde erişebilirler. Burada da servis yönelimli mimarinin prensiplerinden yeniden kullanılabilirliği mimarimiz içerisine yerleştirmiş oluyoruz. Bu durumda WS-RM ve MTOM protokol desteği ve diğer WS-* protokoller ortadan kalkmış olacaktır. Servisler içerisinde bu farklı durumları desteklemek için temel servis işlemlerini içeren ayrı bir arayüz, arayüze bağlı farklı servis kontratları ve bağlantı noktaları oluşturarak bu gerçekleştirim yapılabilir.

```
<ServiceContract (Name:="TelefonTemelDuzey")> _
Public Interface ItelefonBasic
    <OperationContract ()> Function aramaYapB2 (ByVal kadi As
String, ByVal sifre As String, ByVal tel As String, ByVal
mesaj As String, ByVal gun As Integer, ByVal ay As Integer,
ByVal yil As Integer, ByVal saat As Integer, ByVal dk As
Integer) As String

    <OperationContract ()> Function aramaYapB (ByVal kAdi As
String, ByVal Sifre As String, ByVal g As Gorusme) As String

.....
```

Şekil 5.37 Temel Seviye Servis Kontratı

Fonksiyonları oluşturan kodlarımız her düzeyde aynı olmasına karşın anlaşma sunuş biçimleri, protokollerdeki farklılığı kontratlar düzeyinde tanımlayacağından servis yönelimli mimari prensiplerinde anlaşmalar kontratlarla yapılır prensibine uygunluk göstermektedir.

```
<endpoint address="basic" binding="basicHttpBinding"
bindingName="basicbinding" name="GenelServis.BasicTelefon"
contract="GenelServis.ItelefonBasic">
</endpoint>
```

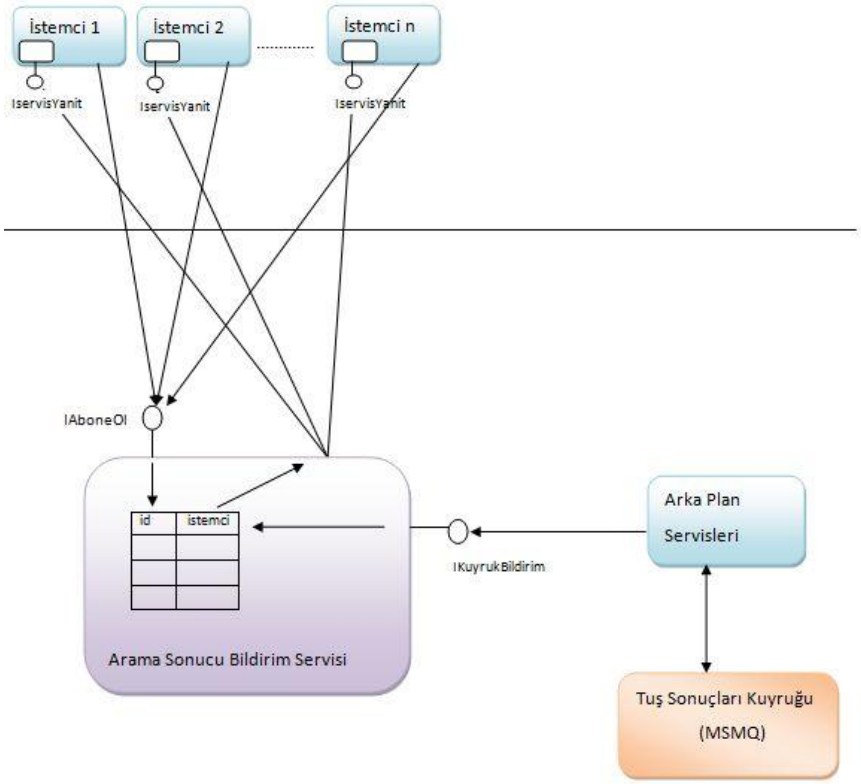
Şekil 5.38 Temel Seviye Bağlantı Noktası Yapılandırması

5.4.3 Çift yönlü Servis –İstemci İletişimi

Bu bölüm de bölüm 3.1. bahsedilen temel mesajlaşma kavramlarından kompleks çift yönlü mesajlaşma yayınla-abone ol mesajlaşma örüntüsünün özel bir formu içinde incelenip uygulanmıştır. Eş zamanlı çalışma servis ve istemci açısından önemlidir. Servisten istemciye giden mesajlar bu tür haberleşmelerde tek yönlü mesajlaşma örüntüsü olarak düzenlemek mesajlaşma içerisinde çıkabilecek olan hatalar karşısında sistem çalışmasını sürdürecektir. Böyle bir yapıda servislerin istemcinin yaşam döngüsünü kontrol etmesi söz konusu değildir [20]. HTTP üzerinden yapılacak olan mesajlaşmalarda istemci tarafında ayrı bir bağlantı noktası oluşturulmalı ve istemci kodu servisten gelen mesajları buradan alıp kendi iç süreçlerinde kullanabilmelidir.

Şekil 5.39 üzerinde bu sistemin çalışma şekli görülmektedir. Burada birden fazla istemci görüşme idlerini Bildirim Servisine kayıt ettirerek arama sonuçlarından haberdar olmak istediklerini bildirmektedir. Servis bu kayıt esnasında takip edeceği id'lerle birlikte sonuçların geri döndürülmesinde kullanılmak üzere istemci bilgilerini de servis tanımlanan ara yüz çerçevesinde saklar. Bu bilgileri saklamak için. net framework içerinden mevcut anahtar değer eşleşmesini sağlayan Dictionary veri yapısı kullanılmaktadır.

Belirli görüşmeler için abone olunduktan sonraki basamak görüşmelerin tamamlanmasını beklemek olacaktır. Arka Plan Servisleri görüşme sonucunu veri tabanına işlerken aynı zamanda bu servisin KuyrukBildirim metodunu kullanarak yapılan görüşme kodunu ve sonucunu servise bildirir. Eğer gelen kod, Dictionary içerisinde mevcutsa istemci bilgileri alınarak istemciye geri dönüş yapılır.



Şekil 5.39 Çift-Yönlü İletişim Yapısını Kullanarak Servilerden Bilgi Edinme

Yapısı gereği TCP/IP gibi HTTP çift yönlü iletişimi desteklemez bu yüzden bağlantı noktaları için wsDualHttpBinding bağlantı türü kullanılmaktadır. Fakat arka plan servisleri çift yönlü iletişim yapmayacağından bu bağlantı noktasından farklı ve daha hızlı çalışabilen bir bağlantı noktasını kullanması daha uygundur. Şekil 5.39'de bu iki farklı bağlantı noktaları görülmektedir. Servis kodlarını incelediğimizde istemcilere tek bir servis referansı verilmekte fakat yapılan anlaşmalar sayesinde istemci tarafında servis sonucu bildirmek için gereksinim duyulan ayrı bir bağlantı noktası oluşmaktadır. Burada servis kontratı aynı zamanda istemci tarafında geri dönüş için gerekli yapıyı oluşturacak bilgileri de içermektedir.

Servis kontrat ara yüzlerini ve sistemin Visual Basic kodlarını incelersek;

```
<ServiceContract (callbackContract:=GetType (IServisYanit))>_
 Public Interface Igeriyanit

    <OperationContract (isOneway:=True)> Sub AboneOl (ByVal id
As String)
End Interface

Public Interface IServisYanit

    <OperationContract (isOneWay:=True)> Sub Degerler (ByVal
deger As String)
End Interface
```

Şekil 5.40. Çift Yönlü Kontratların Oluşturulması

Bu kodlarda temel olan servis kontratı Igeriyanit olup, bu kontratta CallbackContract özelliği içerisinde IServisYanit tanımlanmakta ve istemci bu servisi referans ettiğinde servisten gelecek olan bağlantı ve metot yapısını görmektedir.

```
Implements Igeriyanit, IKuyrukBildirim

Public liste As New Dictionary (Of String, IServisYanit)

Public Sub AboneOl (ByVal kAdi As String, ByVal sifre As String,
ByVal id As String) Implements Igeriyanit.AboneOl

  Dim callback As IServisYanit =
OperationContext.Current.GetCallbackChannel (Of IServisYanit) ()
  liste.Add (id, callback)

End Sub

Public Sub Bildirim (ByVal id As String, ByVal degerler As
String) Implements IKuyrukBildirim.Bildirim
        Dim donus As IServisYanit
        donus = liste.Item (id)
        donus.Degerler (degerler)
End Sub
```

Şekil 5.41 Kontratların ve Servis Kodlarının Bir Bölümünün Gerçekleştirimi

Servise yapılan ve servisten dönen tüm çağrıların ve bildirimlerin tek yönlü yapılmasındaki amaç; istemcilerin servise bilgiyi aktardıktan sonra ek bir yanıt beklememektedir. Bu durum herhangi bir tarafta çıkabilecek problemleri engelleyecektir.

Servis istemciye referans edildikten sonra istemci tarafında oluşturulacak kod örneği şekildeki gibi olabilir.

```
Dim s As ServiceReference1.IgeriyanitClient
Dim context As New InstanceContext (Me)

s = New ServiceReference1.IgeriyanitClient (context)

s.AboneOl ( "..")

Public Sub Degerler (ByVal deger As String) Implements
ServiceReference1.IgeriyanitCallback.Degerler

        ' Gelen değerle yapılması istenen işlem
End Sub
```

Şekil 5.42 Örnek İstemci Kodu ve Servis Çağrısına Yapılacak İşlemin Gerçekleştirimi

Burada ayrı bir InstanceContext yaratılmakta ve servisten gelen çağrıları takip etmesi için kullanılmaktadır.

Tüm bu kavramların dışında son olarak dikkat edilmesi gereken noktalardan birisi de servislerin sunucu üzerinde oluşturulması şeklidir. Eğer servis çağrı bazlı veya kullanıcı bazlı oluşturulursa servis üzerinde oluşturulan dictonary veri yapısı tüm istemciler veya çağrılar için ayrı ayrı oluşturulacaktır. Bu durumda kuyruktan gelen bilgilerle istemcilerin belirttiği idlerin veri yapısı üzerinden eşleşmesi gerçekleşmeyecektir. Tüm servis kullanıcıların ortak bir servisi ve veri yapısını kullanması için servis tekil olarak oluşturulmak zorundadır.

Bunun için servis gerçekleştirim kodunda Sınıfın başına tabloda görüldüğü gibi tanımlama yapılmaktadır.

```
<ServiceBehavior (InstanceContextMode:=
InstanceContextMode.Single)>Public Class ServisYanit

   'Kodlar

End Class
```

Şekil 5.43 Servis Kodunun Tek Bir Instance Üzerinden Hizmet Vermesi

Ws-DualBinding içerisinde varsayılan olarak Ws-ReliableMessaging ve Ws-Security etkin durumdadır.

Şekil 5.44 Çift Yönlü İletişimin Örnek Gerçekleştirimi

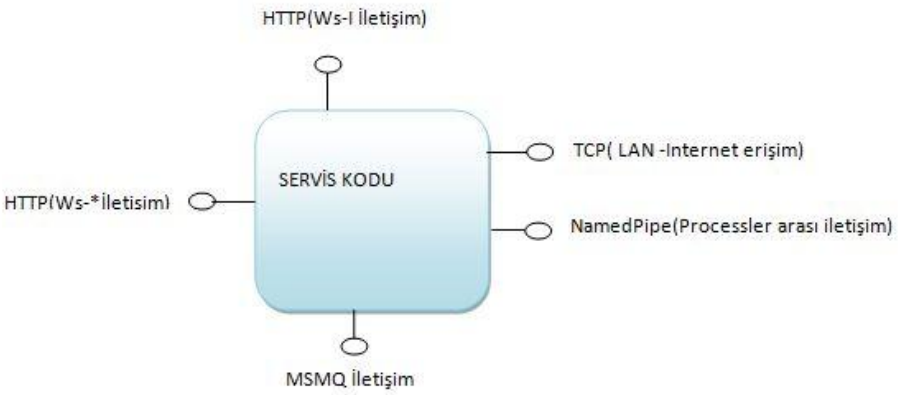
Şekil 5.44'de bu çalışmanın bir örneği görülmektedir. Bu örnekte bir AramaYap ile Genel Telefon servisine bağlantı kurulmakta ve belirtilen mesaj belirtilen telefon numarasına, belirtilen tarihte arama yapılmaktadır. Ve AramaYap metodunun sonucu olarak görüşme kodu gelmektedir. Bu görüşme koduyla örneğe bağlı olan "Sonuç otomatik olarak gelsin tıklanarak" GeriYanıt servisine Görüşme kodu ve istemcideki geri dönüş bilgileri iletilmektedir. O kodlu görüşme gerçekleştirildikten

sonra sonuçlar (124 tuş yanıtları) servisten örnek uygulamanın gerekli kod bölümüne otomatik olarak işlenmektedir.

Bu bölümde servis yönelimli mimarinin temel özelliklerinden servislerin gerektiğinde durum özelliklerini saklayabilmeleri incelenmiş ve bununla birlikte servislerin çift yönlü mesajlaşma örüntüsüne uygun olarak çalışabilip abone ol – yayınla mimarisine uygunluğu gösterilmiştir.

5.4.4 Farklı Bağlantı Noktası Seçenekleri Tasarımı

Bölüm 5.4.2 de farklı bir bağlantı seçeneği kullanarak WS-I yapısında servislerimizi kod değişikliği yapmadan kullanabilmeyi incelemiştik. Servis yönelimli mimari içerisinde bileşenler web servislerinin getirilerine ek olarak iş mantığının ana bölümünü oluşturmakta ve her türlü bileşenle HTTP den farklı olarak iletişime geçebilmektedir. Farklı noktalarla iletişim anlaşmaları aynı kontratların farklı bağlantı noktalarına bağlanmasıyla gerçekleşmektedir. Servis yönelim dışında eski teknolojilerde aynı işi yapan her bir bağlantı noktası için farklı kodları tasarlamamız ve gerçekleştirmemiz gerekmekteydi. Web servisi üzerinden HTTP bağlantısı ve HTTP üzerinden iletişim için ayrı bir kod, CORBA, DCOM gibi uzak makinelerdeki kodları RPC üzerinden çalıştırmak için farklı bir kod tasarımı gerekmektedir. Servisimiz bu yapısıyla farklı bağlantılarını gerektiğinde sadece yapılandırma ayarlarını yaparak gerçekleştirebilir. Bu sayede servisimiz proje kapsamında aynı işi farklı bağlantı seçenekleriyle beraber sunarak servislerin tekrar kullanılabilir olması gibi bileşen tabanlı ve servis yönelimli mimari maddeleriyle uygunluk gösterebilmektedir. Fakat bir servis için bu bağlantı noktalarının tümünü bir istemciye açmak sıklıkla uygulanabilen bir durum değildir.



Şekil 5.45 Servis Üzerinde Kullanabilecek Muhtemel Bağlantı Noktaları

5.4.5 Ws-Addressing Kullanımı Ve Keşfedilebilirlik

Sistem gerçekleştirimi kapsamında detaylı incelenmesine karşın oluşturulan servisler gerektiğinde Ws-Addressing kapsamında incelediğimiz şekilde gelen mesajların yönlendirilmesi yanıtlanması ve bilgilendirilmesi gibi konuları servis tanımlarında gerçekleştirilecektir.

Servis yönelimli mimarinin temel noktalarından birisi de servislerin keşfedilebilir olmasıdır. WSDL dokümanlarının yanı sıra servis yönelimli mimarinin önemli parçalarından birisi olan MetadataExchange protokolünün de servislerimiz içerisinde gerçekleştirimi yapılmış ve bu sayede istemciler dinamik olarak bu bilgilere erişebilir duruma gelmiştir. MetadataExchange bilgilerinin elde edilmesi ayrı bir bağlantısı üzerinden Şekil 5.46'deki gibi gerçekleştirilebilir.

```
<endpoint address="mex" binding="mexHttpBinding"
contract="IMetadataExchange"/>
```

Şekil 5.46 Servis Üzerinde MetadataExchange Protokolünün Aktif Hale Getirilmesi

Tablodaki kod içerisinden de görebileceğimiz gibi mex http üzerinden elde edilebilmektedir. Web istemcileri içerisinde bunun doküman olarak elde edilmesi için yapılandırmaya <serviceMetadata httpGetEnabled="true"/> olarak eklenmektedir.

Bu özellikler servis yönelimli mimaride servis kompozisyonları oluşturabilmeyi ve servis anlaşmalarıyla uyumluluğun sağlanması gibi servis yönelimli özellikleri sağlamaktadır.

5.4.6 Farklı Yayınlama Ortamları

Web servisleri temel olarak web sunucularında barındırılmakta ve yayınlanmasından web sunucuları sorumlu olmaktadır. Bunun başlıca nedenlerinden birisi web servisi kodlarının istemcilerle temel HTML gibi HTTP üzerinden mesajlaşan XML temelli protokolleri kullanmasıydı. Bu durumda güvenirliliği ispatlanmış olan web sayfalarını sunan sunucuları kullanmak en mantıklı çözümlerden birisi haline gelmiştir. Servis yönelimli mimariyle birlikte servislerin otonom olması ve sınırlanırın belirgin olması gibi servis yönelim yaklaşımların; servislerin sadece web sunucularının içerisinde ve web sunucusunun kontrolünde yayınlanabilir olması kısıtlayıcı bir durumdur. Servis yönelimli mimari bölümünde belirttiğimiz gibi oluşturulan servisler taşındığı her ortamda yaşayabilmeli ve gerektiğinde yaşam döngüsünü ve yönetimini başka bileşenlerden bağımsız olarak kontrol edebilmelidir. Bu da bir bakıma kendi kendini yayınlama özelliğidir. Bu sayede bileşene ister LAN'dan ister internetten isterse aynı makinedeki uygulamalar servise erişirken servisin bulunduğu konum servisin gerçekleştirdiği kodların çalışmasını etkilemeyecektir.

Servislerin kendi kendilerini yayınlamasında ise servis kullanıcı tarafından başlatılıp, denetlenmesi ve izlenmesi işletim sistemi servislerinden bağımsız olarak kullanıcı tarafından gerçekleştirilmektedir. Servisin yayınlama seçenekleri servis iş kodundan bağımsız olmaktadır. Servis koduna ek bir yük kazandırmadan web sunucusunda yayınlanan bir servisin TCP ve NamedPipe gibi bağlantı durumlarında ise web sunucu kaynaklarının bu durumu riskli bulacağından kendisi proje kapsamında bu tür kodlarını kendini yayınlayan servisler olarak geliştirmek en uygun çözümlerden birisidir. Burada dikkat edilecek noktalardan birisi kuşkusuz güvenlik duvarı ayarlarının kendi kendini yayınlayan servis koduna gerekli izinleri vermesidir. Bu kapsamda Genel Telefon Servisimiz kendi kendini yayınlayabilecek durumdadır.

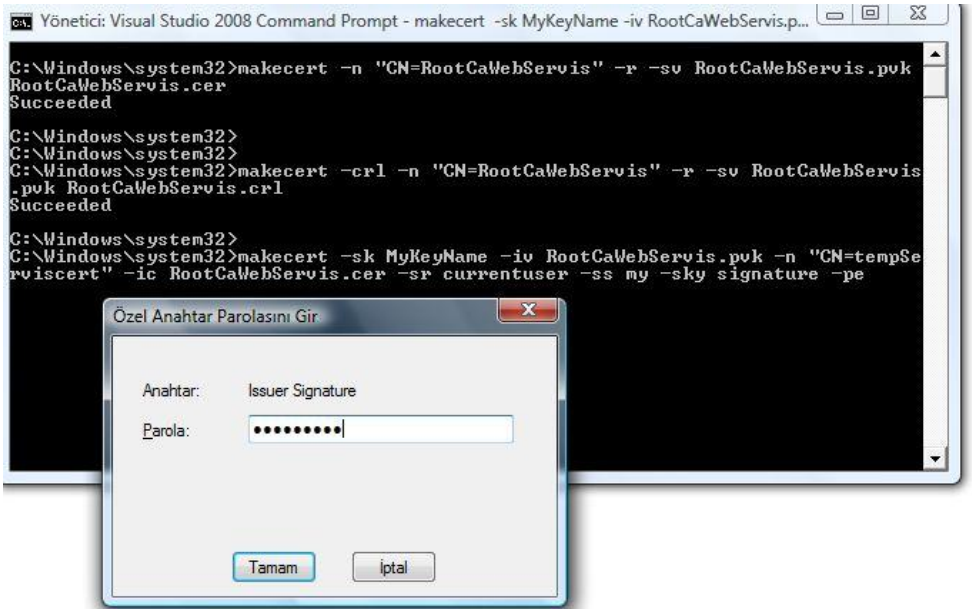
5.4.7 Güvenlik Ve WS-Security

Güvenlik web mesajlaşmalarındaki en önemli noktalardan birisi olduğundan servis yönelimli mimari içerisinde de önemli bir yer tutmaktadır. Servis yönelimli mimarinin temel özellikleri incelendiğinde karşımıza WS-Security protokolü temel bir nokta olarak çıkmaktadır. Burada 3.3.5. bölümde bahsettiğimiz WS-Securtiy protokolü ile beraber mesajların hedeflerine değişmeden ulaştığını belirlenmesi için Xml-Signature ve Xml-Encryption ile de verilerin şifrelenmesi yapılmaktadır.

Proje kapsamında temel olarak alındığında dışarıdan gelen istemciler, kullanıcı adı ve şifresi belirterek bir web uygulamasına giriş yapar gibi sisteme giriş yapmaktadır. Bununla birlikte bu kullanıcılar belirli rollere sahip olabilmekte ve kod içerisinde belirli işlemler belirli rollere sahip kullanıcılar tarafından çalıştırabilmektedir.

Proje kapsamında mesajlar içerisindeki imzalama ve şifreleme işlemleri sertifika tabanlı gerçekleştirilmektedir. Proje içerisindeki bazı mesajlar sadece şifrelenip veya sadece imzalanabilmelerine karşın tüm mesajlarımız hem imzalanıp hem şifreleme işlemlerine tabi tutulmuştur. Servislerimizde kullanılmak üzere geçici sertifikalar makercert.exe komutu kullanılarak X.509 standardına uygun şekilde oluşturulmuş ve oluşturulan bu sertifikalar işletim sisteminin güvenilen kök sertifikaları yetkilileri bölümüne eklenmiştir.

Şekil 5.47 oluşturulan örnek sertifikalardan birisinin gerçekleştirimi görülmektedir.



Şekil 5.47 Makecert Komutu ile Örnek Sertifika Oluşurumu

Böylelikle servis kodlarımız kullanacakları sertifikalara işletim sistemi kaynaklarından erişmektedirler.

Microsoft Root Certificate Auth...	Microsoft Root Certificate Authori...	10.05.2021	<Tümü>	Microsoft Root Cert...
NO LIABILITY ACCEPTED, (c)97 ...	NO LIABILITY ACCEPTED, (c)97 V...	08.01.2004	Zaman Damgalaması	VeriSign Time Stam...
RootCaWebServis	RootCaWebServis	01.01.2040	<Tümü>	<Yok>
Secure Server Certification Auth...	Secure Server Certification Author...	08.01.2010	Sunucu Kimlik Doğ...	VeriSign
Starfield Class 2 Certification A...	Starfield Class 2 Certification Auth...	29.06.2034	Sunucu Kimlik Doğ...	Starfield Class 2 Cer...

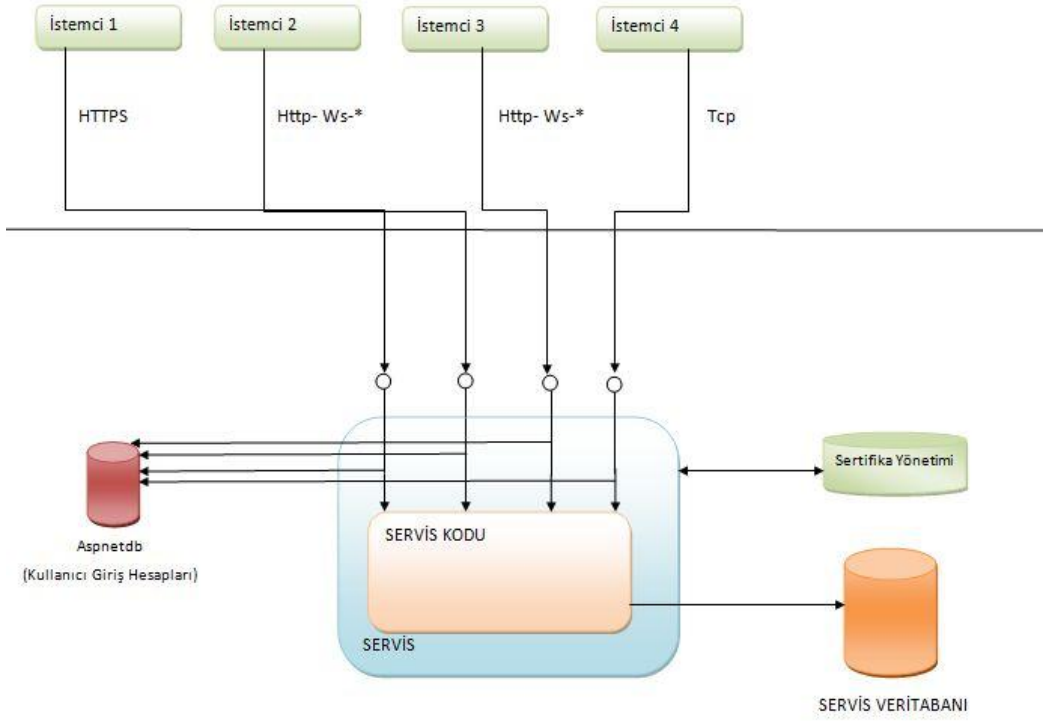
Şekil 5.48 Oluşturulan Sertifikanın İşletim Sistemine Eklenmiş Durumu

Servislerdeki kullanıcı girişleri ve yetkilendirilmesi konusunda ise ASP.net programlama platformuyla tümleşik olan SQLMembership yapısı kullanılmıştır. Bu sayede servislerimizi kullanacak olan her türlü istemciler için kullanıcı girişleri bu noktadan yapılabilmektedir.

WS-I kullanarak temel düzeyde servisleri kullanacaklar güvenlik açısından HTTPS protokolünü kullanacaklardır. Bu sayede güvenlikleri iletim katmanında olacaktır. Fakat iletişim sadece 2 nokta arasında olacağından ara servislere gerek duyulması durumunda geçersiz olacaktır. HTTPS olarak servislerin yapılandırılması için web

sunucusu üzerinde gerekli yapılandırmalar servis için hazırlanan sertifikanın web dizin güvenliğine eklenmesiyle sağlanmış olur.

İletim katmanı dışında WS-Security ve ilintili protokollerin kullanılabilmesi için wsHttpBinding bağlam türü bitiş noktası ayarı olarak seçilmiş olup WS-* protokollerinin sağlayacağı avantajlar kullanılabilmektedir. Böylelikle oluşturduğumuz servis çeşitli servis kompozisyonlarında güvenli bir şekilde kullanılmakta ve servis yönelimli mimarinin prensiplerinden bir tanesine daha uygunluk göstermektedir.



Şekil 5.49 Servis'in Tüm Bağlantılarında Ortak Kullanıcı Veritabanı Kullanımı

Şekil 5.49'de görüldüğü gibi kullanıcı girişlerinde istemcilerden gelen kullanıcı adı ve şifreleme doğrulama işlemleri servis kodu yerine WCF içerisinde otomatik olarak gerçekleşmektedir. Bu sayede Temel Çözüm'de gördüğümüz kullanıcı adı ve şifrelerinin web metodu içerisine yerleştirilmesine gerek kalmadan servis iş mantığından soyutlanması gerçekleştirilmiştir. Bu da çok servis yönelimli mimariden önceki mimarilerin temelini oluşturan soyutlanma yapısına uygun düşmektedir.

Servislerde kullanıcı adı ve şifreleri SOAP zarf başlığında taşınmaktadır. Bu bilgilerin SOAP zarf başlıklarında taşınması bizlere kullanıcı bilgilerinin bir kez girilmesinin yanında birden fazla servis metodunu şifresiz çağırma özelliğini kazandırmıştır. Birden fazla mesajın gönderiminden sorumlu olan protokol arka planda Ws-SecureConversation olacaktır.

Ws-Security ile kullanıcıların Giriş işlemlerini gerçekleştiren yapılandırma kodlarını incelersek:

Öncelikli olarak Kullanıcı Giriş Bilgilerinin bulunduğu web sunucusu bağlantı ayarları ve Kontrol işlemlerini gerçekleştiren yapılandırma ayarları Şekil 5.50'da görülmektedir. Bu ayarlarla birlikte servis doğrulama için gerekli veritabanı bağlantılarına sahip olmuştur.

```
<connectionStrings>
    <add name="MySqlConnection" connectionString="Data
Source=WEBSERVISLERI\SQLEXPRESS;Initial
Catalog=aspnetdb;Integrated Security=True" />
</connectionStrings>

<membership defaultProvider="SqlProvider"
userIsOnlineTimeWindow="15">
      <providers>
        <clear />
    <add
          name="SqlProvider"
       type="System.Web.Security.SqlMembershipProvider"
          connectionStringName="MySqlConnection"
          applicationName="cozum1"
          enablePasswordRetrieval="false"
          enablePasswordReset="true"
          requiresQuestionAndAnswer="false"
          requiresUniqueEmail="true"
          passwordFormat="Hashed" />
     </providers>
    </membership>
```

Şekil 5.50 Kullanıcı Hesaplar Veritabanına Bağlantı Yapılandırma Ayarları

Yapılandırma ayarlarının bölümünde ise servisin kullanıcı girişlerinde kullanacağı yöntemle birlikte servis sertifikasının konumu yapılandırma ayarlarının Behavior bölümünde görülmektedir.

```
<behavior name="GenelServis.TelefonBehavior">
  <serviceMetadata httpGetEnabled="true"/>
  <serviceDebug includeExceptionDetailInFaults="false"/>
  <serviceAuthorization
     principalPermissionMode="UseAspNetRoles"/>
  <serviceCredentials>
    <userNameAuthentication
     userNamePasswordValidationMode="MembershipProvider"
     membershipProviderName="sqlProvider"/>
      <serviceCertificate findValue="RootCaWebServis"
           x509FindType="FindBySubjectName"
     storeLocation="LocalMachine" storeName="Root"/>
  </serviceCredentials>
</behavior>
```

Şekil 5.51 Membership ve Sertifika Yapılandırması

Şekil 5.52'deki yapılandırma bölümünde ise wsHttpBinding üzerinden security mode, message seçilerek güvenliğin SOAP zarfının gövde kısmı için geçerli olacağı belirilmektedir. Bu sayede başlık kısmı şifrelenmediğinden WS-* protokollerine bağlı kazanımlar servis yönelimli bir ortamda gerektiğinde elde edilebilecektir.

```
<wsHttpBinding>
<binding  name="wsHttpEndpointBinding"  >
  <security mode="Message">
   <message clientCredentialType="UserName"
     establishSecurityContext="true"
     negotiateServiceCredential="false"/>
  </security>
 </binding>
 </wsHttpBinding>
```

Şekil 5.52 Message Security Yapılandırması

Bu servisi kullanan istemci tarafında bu yapılandırma ayarlarına karşılık gelen istemci yapılandırma ayarlarıyla birlikte sertifika bilgisi bağlantı noktası ayarlarında gelmektedir. Bu noktada dikkat edilmesi gereken noktalardan birisi de istemci ve sunucu arasındaki zaman farkıdır. Varsayılan olarak bu fark 5 dk. Daha fazla olma durumunda Ws-Security içerisindeki wsu:Timestamp özelliğinden dolayı sistemde hata meydana gelecektir.

```
s.ClientCredentials.UserName.UserName = "tobb2"
s.ClientCredentials.UserName.Password = "P@ssword"

s. AramaYap (……)
```

Şekil 5.53 Muhtemel İstemci Kod Örneği

Kullanıcı adı ve şifresinin bildirimi Şekil 5.53'deki gibi görülmektedir.

```
<identity>
     <certificate encodedValue="AwAAAAEAAAA . .
. . . . . VHhoVbTtOA==" />

</identity>
```

Şekil 5.54 İstemci Servisi Referans Etmesi Sonucunda Oluşan Sertifika Bilgileri

5.4.8 Servis Gerçekleştiriminin Servis Yönelim Prensiplerine Uygunluğu

Bu gerçekleştirdiğimiz sistemde 3.2. bölümde bahsedilen Servis Yönelimli Mimari prensiplerini hızlıca tekrar edersek. 3.2.1. belirtildiği gibi sistemde servis anlaşmaları belirli WSDL, Mex gibi standart kontratlar üzerinden yapılmaktadır.

Servisler kontratlar düzeyinde anlaştıklarından birbirlerine olan bağımlılıkları en az düzeydedir. Servislerin veritabanı, dosya sistemi gibi iç sıkı bağımlıkları kontratlar sayesinde azalır ve servis dış dünyayla gevşek bir bağlaşım içerine girer. Bu da 3.2.2 deki gevşek bağlaşım prensibine uygun düşmektedir.

Servis kodları soyutlanmıştır burada servislerin kod detayları dış dünyadan saklanmıştır. Böylelikle servislerde meydana gelebilecek teknoloji değişiklikleri, veritabanı, kuyruk, sertifika değişiklileri servise bağlı sistemler 3.2.3. bölümde bahsedildiği gibi birbirini etkilemeden yapılabilmektedir.

Şekildeki servis kodların istemcileri sadece web üzerinden WS-* servis istemcileri olmayabilir. Aynı kodlar farklı web ara yüzü geliştirilirken de kullanılarak kod

tekrarı engellenmiş olunabilir. Böylelikle 3.2.4. bölümde bahsedilen servislerin tekrar kullanılabilirliği özelliğine uygunluk sağlanmıştır.

Servisler farklı platform veya sunuculara taşınsa da birbirleriyle herhangi bir şekilde bağlantı kurdukları takdirde çalışmalarını sürdürebilmektedir. Bu da servislerin 3.2.5 bölümde bahsedilen otonomi prensibine uygundur. Bu özelliği sağlayan en önemli etmenlerden birisi servislerin kendi kendilerini yayınlamasıdır.

Web uygulamalarında genellikle durum yönetimi her istemci için ayrı ayrıdır. Her istemci için ayrı bir oturum açılmakta ve bu oturum içerisinde durum bilgileri saklanmaktadır. Fakat GeriYanıt servisinde böyle bir durumun kullanılması söz konusu olmayacaktır. Burada tüm istemciler ortak bir veri yapısına bilgilerini işlemekte ve geri dönüş bu sayede olmaktadır. Bu da 3.2.6. bölümünde bahsedilen servislerin durumları özelliğini prensibini uygun düşmektedir.

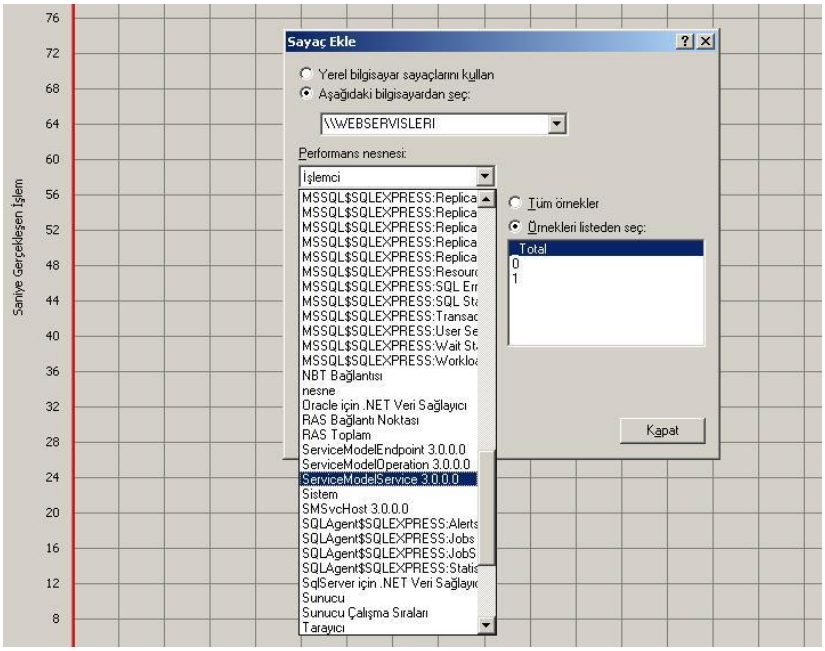
Servisler Mex-WSDL sayesinde dinamik olarak çalışma zamanında keşfedilebilir. Bu özellik 3.2.7. bölümde bahsedilen Servislerin keşfedilebilirliğine uygundur.

Servisler farklı servislerle iletişim kurup bir kompozisyon oluşturabilmektedir. Bu da 3.2.8. bölümdeki servis kompozisyonları prensibine uygunluk göstermektedir.

5.5 Performans Değerlendirmesi

Servislerin üzerinde çalıştığı sunucunun işlemcisi P4 3.0 Ghz , Ram miktarı 1 GB ve İşletim Sistemi Windows 2003 Server'dır.

İşletim sistemine kayıtlı performans sayaçlarından sistem performansını izlemek için öncelikle denetim masası içerisinden sistem performans bölümden. ServiceModelService 3.0 seçilir. Bu performans sayacı web servislerimizi oluşturan WCF'nin temel kütüphanesinin sayacı olup servis denetimi buradan yapılabilir.



Şekil 5.55 ServiceModelService 3.0 WCF Servis Sayacı Eklenmesi

İstemcinin servisin aramaYap2 metodunu çağırma sayıları ve servisin bu metotları işlem süreleri Çizelge 5.1'deki gibidir.

Çizelge 5.1 Servislerin aramaYap2 metodunu işlem süreleri

İşlem Miktarı	telefonservis1 Süre (sn)	telefonservis2(WS-*) Süre (sn)
1000	31	62
5000	154	300
10000	290	605

1000 işlemi temel servis 31 saniyede işlemiştir. Ws-* ise yaklaşık 2 katı bir sürede bu işlemleri gerçekleştirmiştir. İstemciler servise istemleri peş peşe göndermiştir. Yani bir önceki kaydın yanıtı geldikten sonra diğer istem gönderilmektedir. 5000 ve 10000 kayıt içinde süreler Çizelge 5.1'de görülmektedir.

Bu noktada kısacatelefonservis1 için aramaYap2 metodunun işlemlerini tekrarlarsak: Kullanıcı kontrolü (isim+şifre) , veritabanına kayıt, dosya yazma için yetki verilmesi, ses dosyası oluşturulması, yetkinin geri alınması ve görüşme kodunun istemciye gönderimi olarak sıralayabiliriz.

TelefonServis2'de kod olarak aynı kodları kullanmıştır.. Fakat yapılandırmasında Ws-Security kullanımı, Ws-ReliableMessaging protokolünün getirdiği performans kaybı ve de SOAP başlık kısımlarının daha büyük olması ve işlenmesinin zaman alması gibi servis kod farkları işlemlerin sonuçlanma sürelerine yansımıştır. Ayrıca kodlar belirli kütüphaneleri kullandıklarından dolayı bu kodları kendimiz de hızlıca temizleyerek bellek kullanımını düşük düzeydetuatbiliriz. Örneğin W3wp.exe altında çalışan temel telefonservis1.asmx her istekte 800kb civarında bir bellek alanı işgal etmesi ve isteklerin çok sayıda, peş peşe olma durumunda bu alanın normalde çalışma zamanı tarafından temizlenmesini beklemeden temizlenmesi, bellek kullanımını belirli bir düzeyde tutmaya yarayacaktır.

Temel Seviye Servis telefonservis1.asmx'de 1000 kayıt için gönderilen saniyede gerçekleşen işlem miktarını gösteren grafik Şekil 5.56'da görülebilir. Ve test 31 saniyede sonlanmıştır

Şekil 5.56 Saniyede gerçekleşen İşlem Miktarı

Sistem 10 dakika boyunca tabloda belirtilen istemci sayısında test edilmiştir ve toplam işlem miktarları görülmektedir. Burada internet ortamında TOBB içerisinden

Samsundaki sunucuya erişim sağlanmıştır 512 kb/s çıkış hızı olan sunucuya erişim sağlamıştır.

Çizelge 5.2 10 dk. süresince istemcilerden gönderilen istekler ve yanıt miktarları

1 İstemci	3 İstemci	5 İstemci	7 İstemci	10 İstemci
6933	13366	17649	17228	17390

Buradan elde edilen sonuç sistemin belirli bir düzeyde donanımsal işlem kapasitesinin olduğu ve bu kapasite kapsamında tüm istemcilere eşit miktarda yanıt döndürdüğüdür. Kapasite kullanımı en düzeye çıkana kadar istemci sayısının artması toplam işlem miktarını da artmıştır. Donanımsal iyileştirme bu kapasiteyi de arttıracaktır.

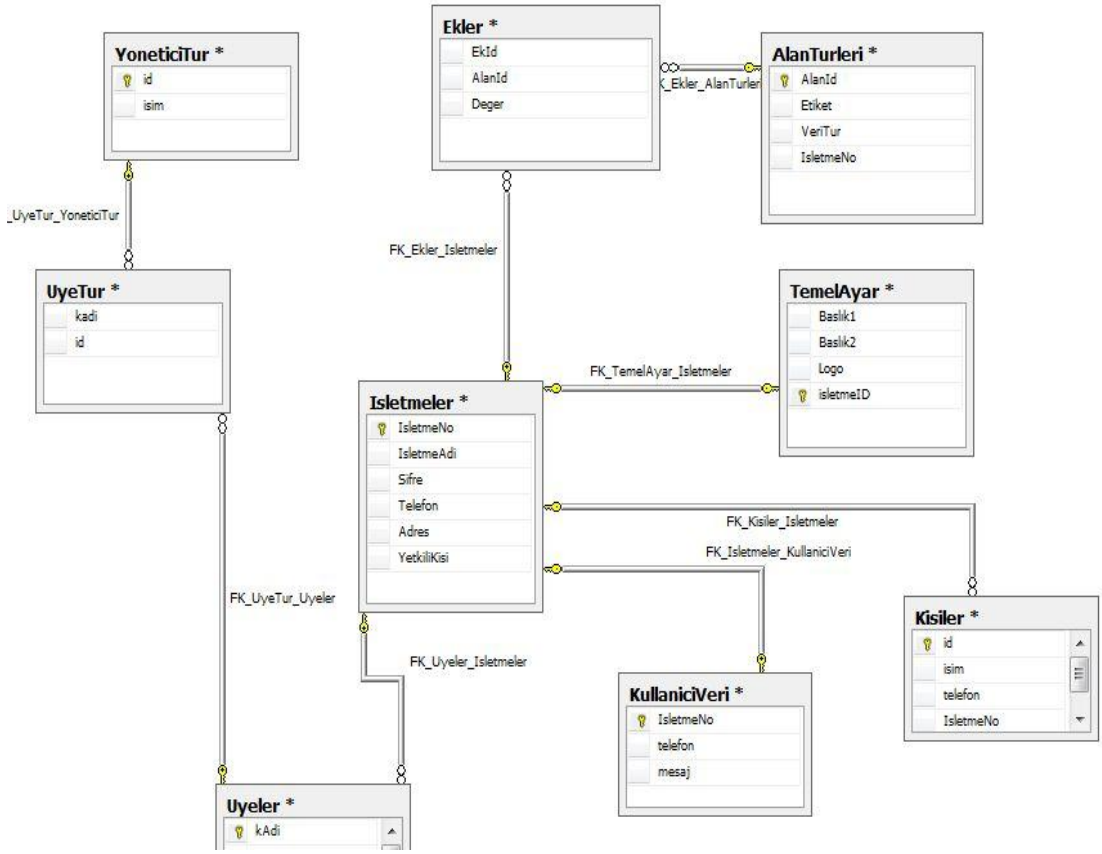
5.6 SaaS Olarak Hizmetlerin Sunulması

Bu bölümde gerçekleştirdiğimiz web servisinin SaaS modeline göre gerçekleştiriminin nasıl olabileceği incelenmektedir. Servisimiz çoklu kiracı modeline göre gerçekleştirilip, arabirimimizde ona göre yeniden düzenlenmiştir. Birden fazla işletme kullanıcısına yazılımın kendileri için tasarlanmış bir yazılım olduğunu hissettirmek temel amaçlardan birisidir. Yazılımımız SaaS olgunluk seviyelerinden üçüncü basamaktadır ve tek veritabanı tek şema yapısını kullanmaktadır. Kullanıcılar kendi web arabirim yazılarında basit firmasal düzenlemeler yapabilmekte kendi logolarını yükleyebilmektedirler. Bunların dışında kullanıcılar için kelime uzunluğu, alan kodu gibi belirli sınırlamalar ve onay mekanizmaları tanımlayabilmektedir. Ayrıca firmalar kendi müşteri bilgilerini tutarken yazılım tarafından sunulan alanların yetersiz olması durumunda; yönetici arabirim üzerinden kendisi alan ekleyebilmekte ve o işletmenin kullanıcıları bu tanımlanan yeni alanları rahatlıkla arabirimlerinde kullanabilmektedir.

Bu noktada yazılımı kullanan tüm işletmeler arka planda tek bir yazılım ve veritabanı kullanmaktadır. Bu durum SaaS seviye 3' e uygundur. Bu durumda tüm işletmelerin arama bilgileri aynı tabloda saklanacaktır. Bu durum işletmeler için güven sorununu da beraberinde getirecektir. SaaS uygulamasıyla beraber Servis Yönelimli Mimari birlikte kullandıklarında bu durumu etkin bir şekilde çözebilir. Bu durumda arama öncesinde işletmeler için gizli olabilecek ve aranacak telefon numarası ve mesaj gibi bilgiler işletmelerin kendi Excel gibi Office yazılımlarında saklanarak, gerektiğinde (Arama öncesinde), SaaS servisi istemciden bu bilgileri talep ederek gizli bilgileri elde edecek ve arama işlemlerini bu bilgilerle gerçekleştirecektir. Bu sayede sunucu tarafında işletmeler için gizli bilgiler saklamadan servis etkin bir şekilde kullanılacaktır. Bu konunun detayları bölüm içerisinde detaylı olarak bahsedilecektir.

5.6.1 SaaS Uygulaması Veritabanı Tablo ve İşlemleri

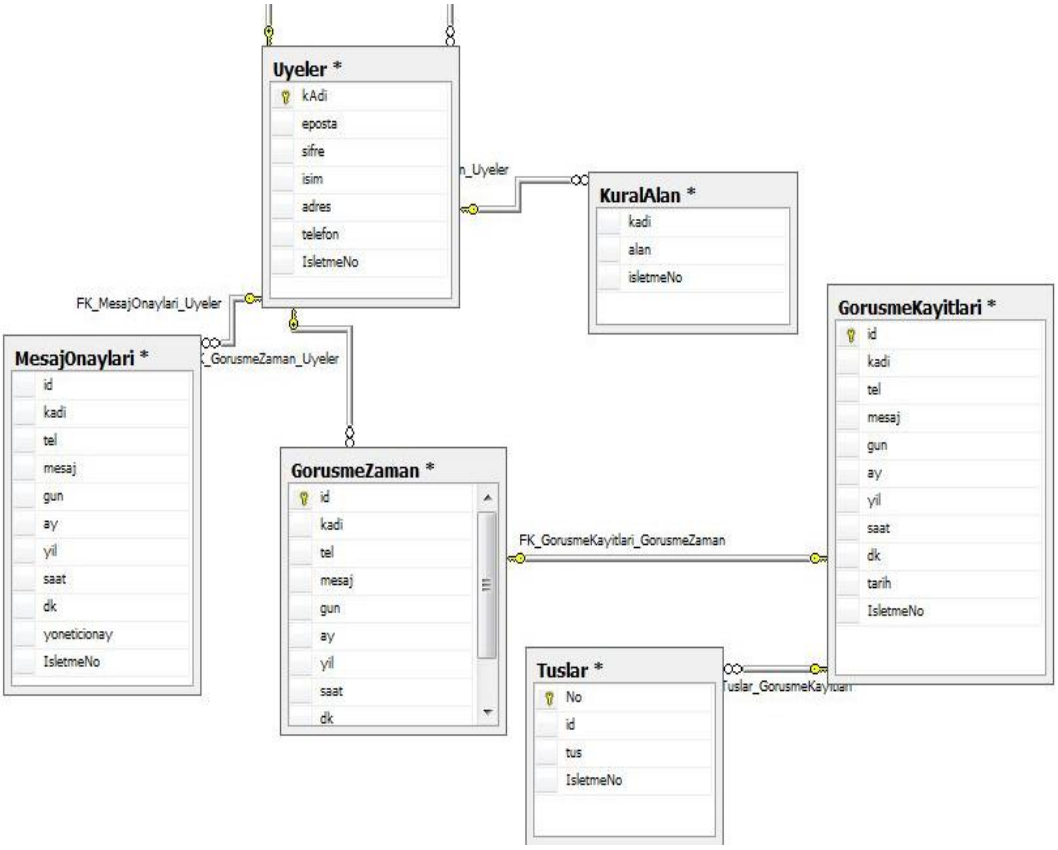
Şekil 5.57 Tabloların Listesi



Şekil 5.58 SaaS Veritabanı Tabloları 1. Bölüm

SaaS uygulamasında gerçekleştirdiğimiz veritabanı diyagramının birinci bölümü Şekil 5.58'deki gibidir. Burada çoklu kiracı modeline göre veritabanı düzenlemeleri gerçekleştirilmiş ve SaaS uygulamasıyla beraber uygulamanın kazandığı özelliklere bağlı tablolarda veritabanına eklenmiştir.

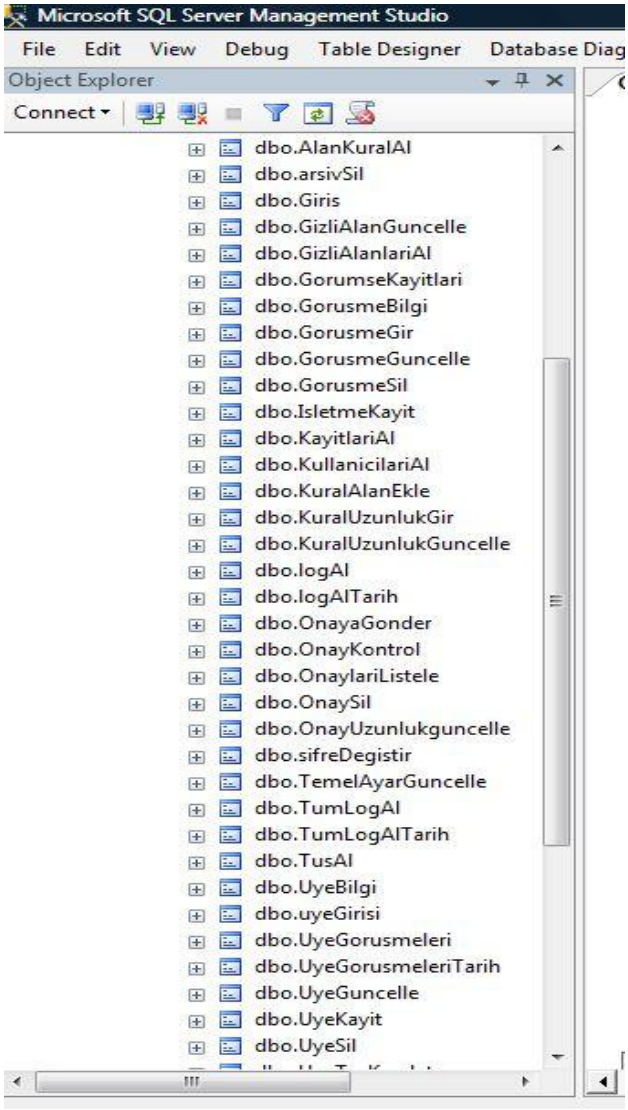
Veritabanı yapısı incelendiğinde tekli veritabanı, tekli şema yapısına uygun olarak tasarlandığı görülmektedir. İşletmeler, Ekler ve AlanTur tabloları incelendiğinde bu yapının SaaS kullanıcı veritabanı genişletmesinde bölümünde incelenen yapıya uygundur. Uyeler tablosu işletmelerin kullanıcı bilgilerini tutmakta ve buna bağlı olan UyeTur ve YoneticiTur tabloları da bu kullanıcıların ne tür yetkilere sahip olabileceğini belirlemektir. Bu, ileriki çalışmalarda kullanıcıların kendi birimlerini tanımlamalarına olanak sağlayacak şekilde tasarlanmıştır.



Şekil 5.59 SaaS Veri Tabanı Tabloları 2. Bölüm

TemelAyar tablosu her işletme için tanımlanacak arabirimi sağlayacaktır. Bu tablo sayesinde farklı işletmelere farklı arabirimler sunulmaktadır. Kişiler tablosu arama işlemlerini gerçekleştirmeden önce işletme kullanıcılarına aranacak kişi hakkında çeşitli bilgiler sunmaktadır. KullanıcıVeri tablosu ise hangi bilgilerin istemciden alınacağını işletme için ayrı ayrı tutmaktadır. KuralAlan Tablosu kullanıcı kısıtlamalarını ifade eder MesajOnayları onaya giden mesajlar, görüşme zaman ise aramayı bekleyen görüşmeler, Tuşlar tablosu kullanıcı tuşları, GörüşmeKayıtları da kullanıcıların yaptığı görüşmeler hakkında bilgi veren tablolardır.

Şekil **5.60** üzerinde veritabanı işlemlerinin birçoğunu gerçekleştiren kayıtlı yordamların bir bölümü görülmektedir. Bu sayede güvenlik, performans gibi birçok kazanım elde edilmiştir

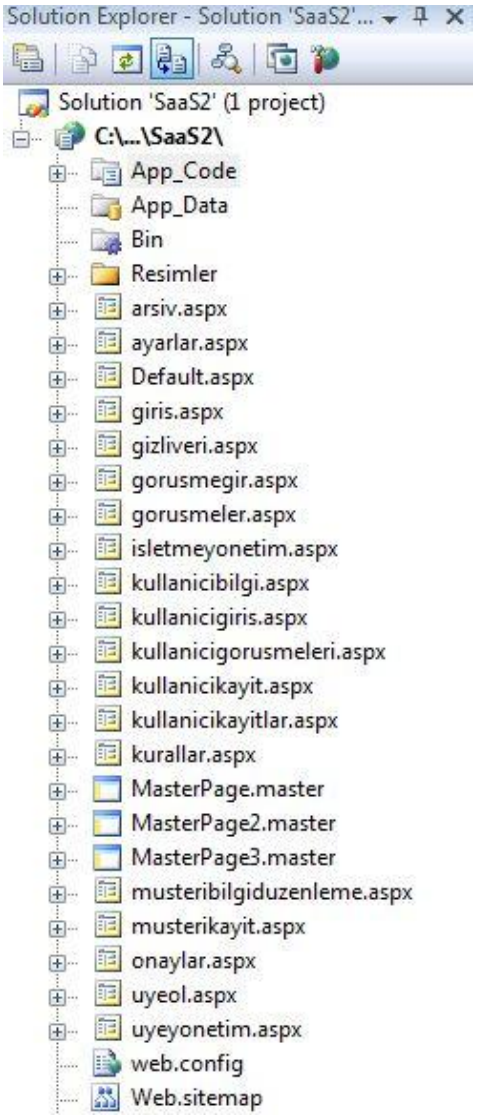


Şekil 5.60 Veritabanı Kayıtlı Yordamları

.

5.6.2 SaaS Uygulamasının Çalışması ve Web Arabirimleri

SaaS Uygulamamız diğer projemizden farklı bir yapıya sahiptir. Bu yapının ilk bölümü olarak Visual Studio içerisindeki sayfa oluşturulan Şekil 5.61'de görüldüğü gibidir.



Şekil 5.61 Web Sayfalarının Genel Listesi

Web sayfaları Şekil 5.61'de toplu olarak görülmesine karşın 2 temel arabirimden oluşmaktadır. Bunlardan ilki Yönetim Arabirimi diğeri ise işletme kullanıcıları arabirimidir. Bu web sayfaları aynı alanda olmalarına karşın yapılacak olan yetkisiz girişler ve istemler sayfalar arasında engellenmiştir.

Yönetim arabirimini oluşturan ve bununla ilişkili sayfaları incelediğimizde:

- Uyeol.aspx:

Şekil 5.62 İşletme Kayıt

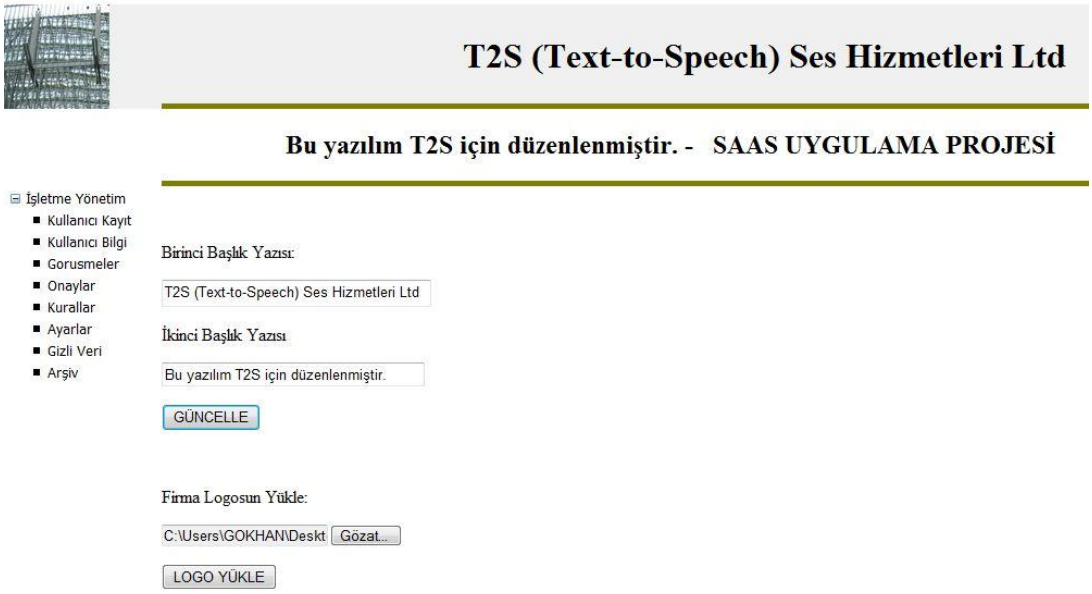
Bu sayfa işletmelerin ilk kayıtları içindir. İşletmeler SaaS uygulaması için gerekli temel bilgileri girmekte ve kayıt sonuçlandığında işletmeye özgü bir işletme kodu işletmeye verilmektedir.

- Giris.aspx:

Şekil 5.63 İşletme Yönetim Giriş Sayfası

İşletme yönetimi Şekil 5.63'de görüldüğü arabirim üzerinden yönetim arabirimine giriş yapmaktadır.

- Ayarlar.aspx:

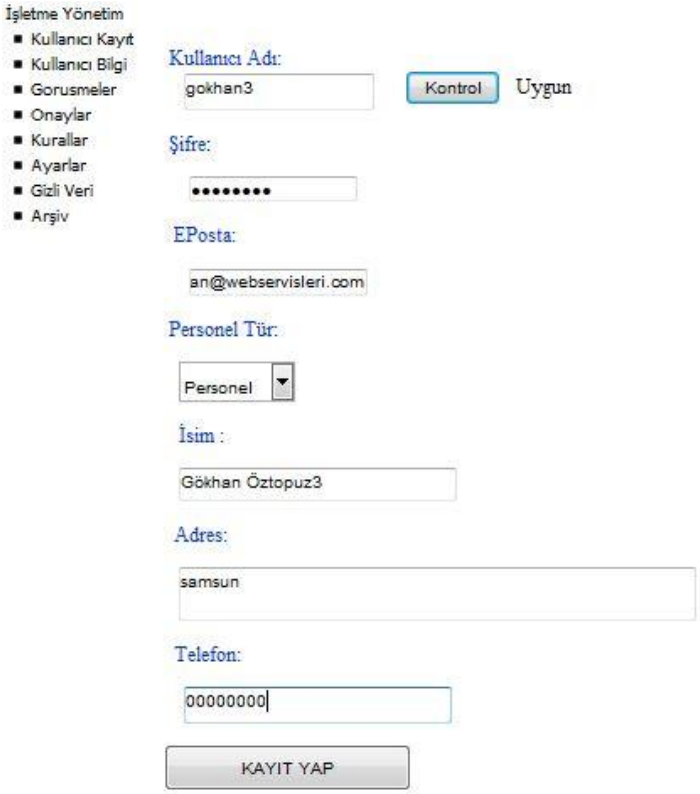


Şekil 5.64 İşletme Temel Ayarlar

Bu sayfa yönetim arabirimi üzerinde bulunmakta ve işletme için temel olabilecek 3 adet arabirim düzenlemesini gerçekleştirmektedir. Şekil 5.64'dan da görülebileceği gibi iki adet başlık ve şirket logosu-resmi işletme tarafından yüklenebilmekte ve tüm kullanıcılar arabirimlerinde bu tanımlanmış olan bilgileri kullanmaktadırlar. Bunlardan farklı olarak gerekirse müşteriler için yetersiz olan veri alanları için kendileri veritabanlarında kendi tanımlayacağı alanları da oluşturabilir.

Şekil 5.65 Veritabanında İşletme için Yeni alanların tanımlaması

- Kullanicikayit.aspx:



Şekil 5.66 Kullanıcı Kayıt Sayfası

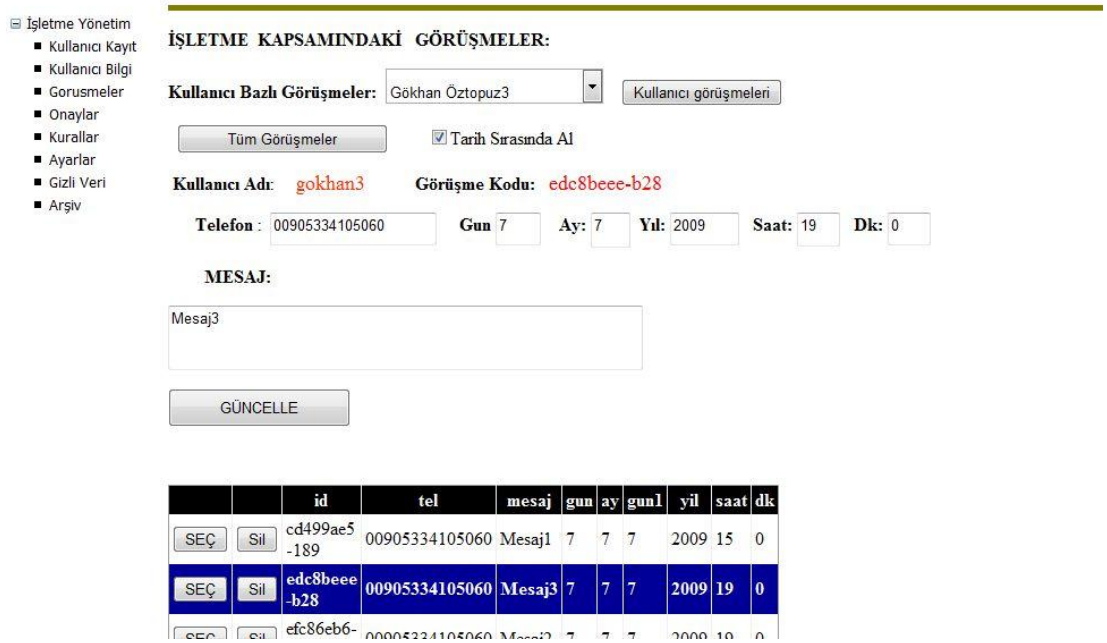
Şekil 5.66 üzerinden görülen arabirimde işletmeler kendi kullanıcılarını tanımlamaktadır. Bu kullanıcılar işletme için uygulama içerisinden web servisleriyle veya web arabiriminden sistemi kullanacak olan kişilerdir.

- Kullanicibilgi.aspx:

Bu sayfa o işletmedeki tanımlanan kullanıcıların listesini vermekte ve kullanıcıların bilgileri üzerinde çeşitli düzenlemeler yapabilmekle beraber kullanıcılar bu sayfa üzerinden silinebilmektedir. Şekil 5.67 üzerinde kullanıcıları bilgilerini düzenleyen bu web sayfasının görünümü yer almaktadır.

Şekil 5.67 Kullanıcı Bilgi Web Sayfası

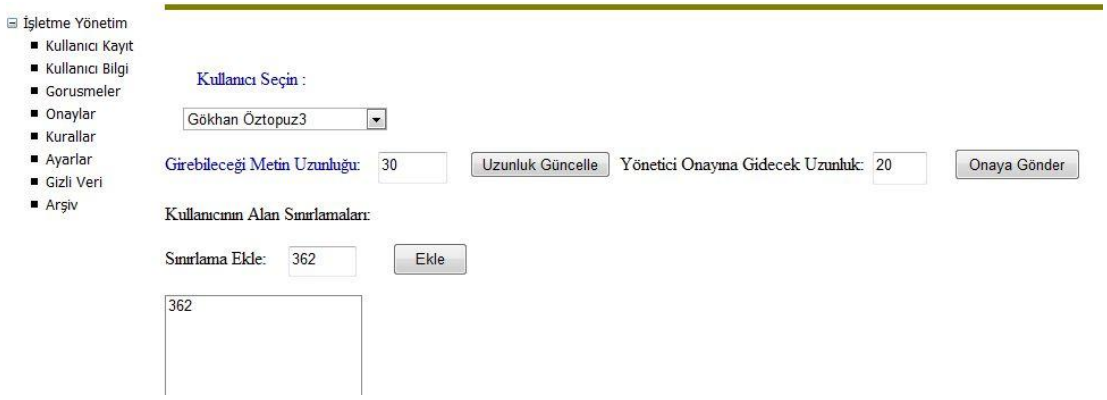
- Gorusmeler.aspx:



Şekil 5.68 Yapılacak Görüşmelerin Listesi

Bu web sayfası yapılacak olan görüşmelerin, işletme bazlı veya kullanıcı bazlı olarak listelenmesini sağlayacaktır. Bu görüşmeler Şekil 5.68'de görüldüğü gibi listelenebilmekte, yönetim tarafından düzenlenebilmekte ve silinebilmektedir.

- Kurallar.aspx



Şekil 5.69 Kullanıcılar için Arama Kurallarının Tanımlanması

Kurallar sayfasında her işletme kendi kullanıcıları için çeşitli sınırlamalar ekleyebilir ve bu sınırlamalar birbirini etkilemez. Şekil 5.69 üzerinde görüldüğü gibi işletmedeki

kullanıcılardan seçim yapılarak, o kullanıcı için sınırlamalar belirlenmiştir. Burada alan sınırlaması ile 362 alan kodlu telefon numaraları yönetici onayına tabi olacak, bu kullanıcı mesajlarında en fazla 30 karakter girebilecek ve 20 karakter sonrası yönetici onayına sunulduktan sonra arama işlemi gerçekleşecektir.

- Onaylar.aspx

Bu sayfa kullanıcıların aramaları girmesi sonucunda onaya giden görüşmelerin yönetici tarafından onaylandığı sayfadır. Yöneticiler burada görüşmeleri onaylayıp bu görüşmeleri normal görüşmelerin bulunduğu tabloya taşıyabilir veya görüşmeleri silebilir. Şekil 5.70 üzerinde Onay bekleyen görüşmelerin listesi görülmektedir. Burada tanımlandığı gibi 20 karakterden uzun karakterler ve alan kodu 362 olan aramalar yönetici onayına sunulmuştur.

Şekil 5.70 Yönetici Onayına Giden Mesajlar

- Arsiv.aspx:

Şekil 5.71 Arsiv.aspx web sayfası

Bu web sayfası kullanıcıların yaptıkları görüşmelerin listesini görüşmelerin kayıtlarını ve verilen tuş yanıtlarını listelemektedir.

Bu aşamadan sonra yönetici tarafından tanımlanan kullanıcıların erişebilecekleri sayfalar gelmektedir. Burada Yönetim bölümünde tanımlanan işletmeden farklı bir işletme tanımlanmış ve kullanıcı işlemleri bu işletme üzerinden yapılmaktadır.

- Musterikayit.aspx:

Şekil 5.72 Müşteri Bilgileri Sayfası

Şekil 5.72'de müşteri kayıt sayfası görülmektedir. Bu sayfada isim ve telefon bilgileri sabit olarak tanımlanmıştır. Adres ve Fax bu işletmeye özgü olarak işletme

yöneticisi tarafından tanımlanmıştır. Farklı işletmelerin bu ekrandaki bilgileri daha farklı olabilecektir.

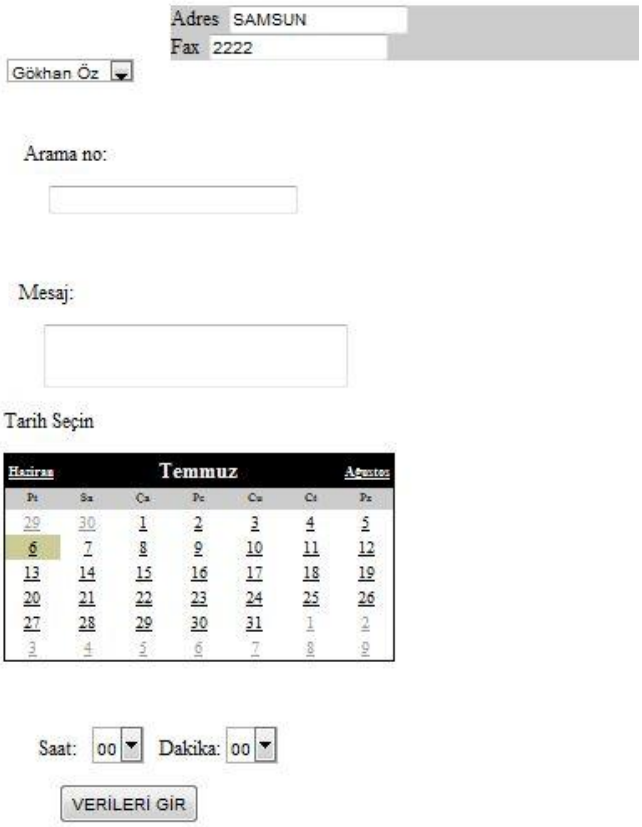
- Musteribilgiduzenleme.aspx:

Şekil 5.73 Müşteri Bilgileri Düzenlenmesi

Bu sayfa üzerinde müşteri bilgileri incelenip güncellenebilmektedir.

- Gorusmegir.aspx:

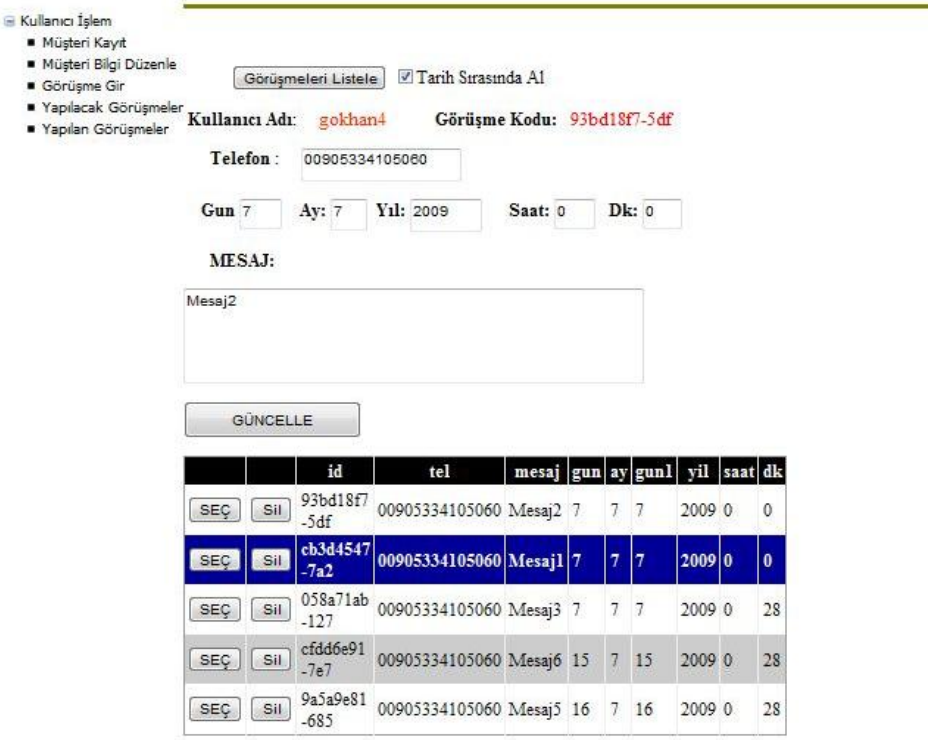
  Bu sayfa üzerinden arama işlemleri telefon, mesaj ve zaman bilgileri seçilerek yapılmaktadır. Ayrıca ufak bir ajax uygulaması bize müşteri hakkında bilgileri ekranda gösterecektir.



Şekil 5.74 Görüşme Giriş Sayfası

- Kullanicigorusmleri.aspx:

Kullanıcılar bu web sayfası üzerinden kullanıcılar sistemi gönderdikleri aramaları gönderimden önce inceleyip gerekli düzenlemeleri yapabilir gerektiğinde silebilir.



Şekil 5.75 Kullanıcı Görüşmeleri Listesi

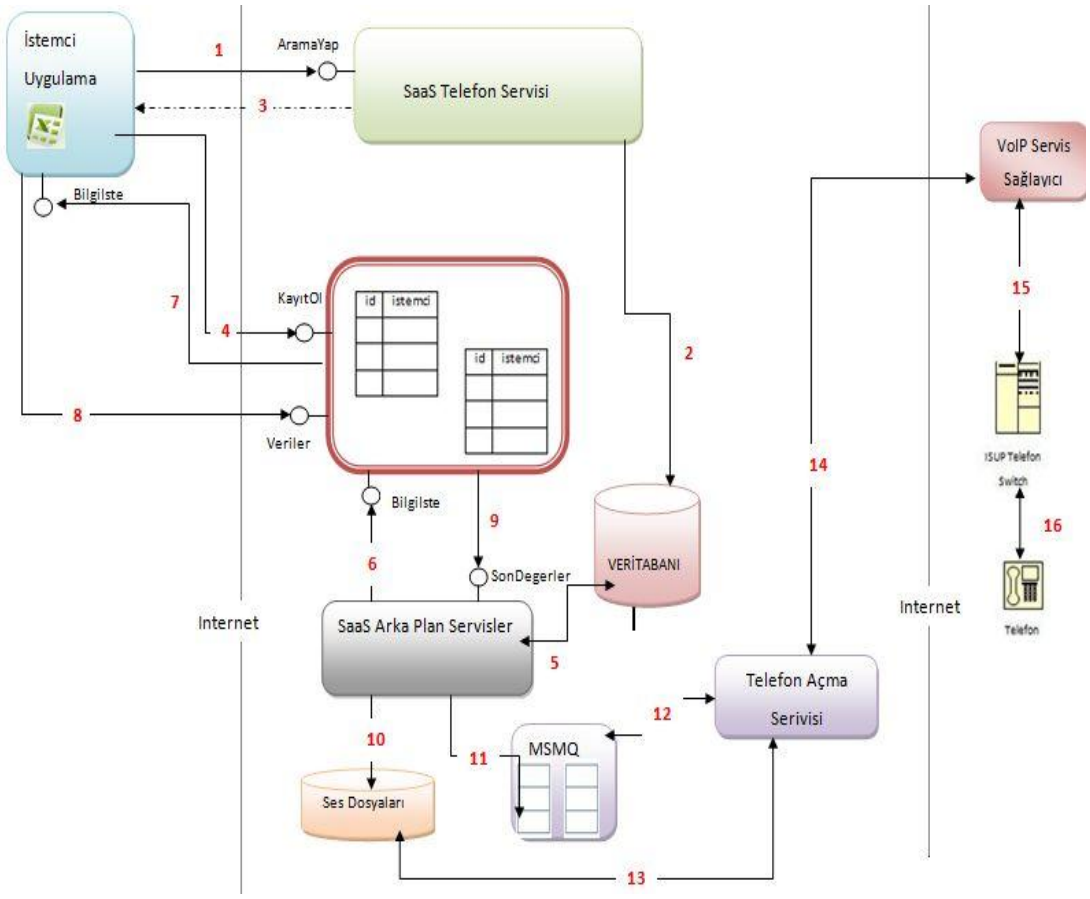
- Kullanicikayitlar.aspx

Şekil 5.76 Görüşme Kayıtları

Kullanıcılar görüşme kayıtları bu bölümden incelemektedir.

5.6.3 Gizli Bilgilerin Çalışma Zamanında İstemciden Elde Edilmesi

Bu bölümde tez içerisinde bahsettiğimiz SaaS uygulamalarında bilgilerin istemciden elde edilmesi incelenecektir. İşletmeler çeşitli durumlarda verilerinin diğer işletmelerin verileriyle alt alta sunucu bilgisayarlardaki veritabanlarında tutmak istemeyebilirler. Bu bölümde yaptığımız çalışma bu sorunun ortadan kaldırılmasına yöneliktir. Gerçekleştirdiğimiz ve önerdiğimiz bu çözümde işletmeler kendileri ek bir yatırım yapmadan başka yerde tutmak istemedikleri verilerini kendi ofislerinde bir Excel tablosunda tutabilmektedir. Aynı zamanda bu bilgileri SaaS uygulamalarıyla güvenli bir şekilde gerektiğinde paylaşabilirler.



Şekil 5.77 Sistemin Çalışması

Şekil 5.77 üzerinde sistemin çalışması görülmektedir. Buradaki işlemler şekil üzerinde belirtildiği gibi rakamsal sırada devam etmektedir. Bu işlemleri incelersek:

- İstemci örneğin bir Excel tablosu, SaaS Telefon hizmetinden web servisiyle kendi işletme ve kullanıcı bilgilerini girerek arama gerçekleştirmesini ister. Burada mesaj ve telefon bilgilerini gizlilik nedeniyle sunucuda bulundurmak yerine hesap tablosundaki gerekli hücrelerde tutar. Telefon numarası ve mesajı X karakteri ile aramaYap metoduna 1. basamaktaki gibi gönderir. Servis bu mesajı alır ve bilgileri 2. basamaktaki gibi veritabanına işler. Servis istemciye girilen görüşmenin kodunu 3. basamakta görüldüğü gibi istemciye geri gönderir.
- İstemci telefon ve mesaj bilgilerini sunucunun kendisi üzerinden elde edeceğini sunucuya iletmiştir. Bu sefer bu bilgileri göndereceği servise kendisini geri dönüş için tanımlar. Tanımlamayla birlikte görüşme kodunu 4. basamakta iletir. Böylelikle GizliBilgiler Servisi hangi kodlu görüşmenin bilgilerini hangi istemciden elde edeceğini bir liste şeklinde tutmaktadır.
- 5. Basamakta Arka Plan Servisi tanımlanan aralıklarla veritabanını kontrol ederek o an için mevcut arama olup olmadığını kontrol eder. Veritabanından alınan bilgilerden telefon ve mesaj bilgileri kontrol edilir ve bu bilgilerin x karakteri olma durumunda bilgilerin görüşme kodu alınır ve GizliBilgiler servisi üzerinde Bilgi Edinme metoduna 6. basamakta gönderilir.
- Bu durumda GizliBilgiler Servisi kendi listesinden gelen kod ile bu mesajı hangi istemcinin gönderdiğini bulur ve o istemciden gerekli bilgileri 7. basamakta ister.
- İstemci sunucudan gelen isteği alır ve gelen koda göre gerekli bilgileri göndermek için 8. Basamakta GizliBilgiler Servisinin Veriler metodunu çağırır.
- Servise bilgiler geldikten sonra servis gizli bilgileri isteminde bulunan ArkaPlan Servislerine, servisin üzerinde bulunan SonDegerler metoduyla görüşme için gerekli telefon ve mesaj bilgilerini 9. basamakta iletir.
- Bundan sonra 10. basamakta gerekli ses sentezlemesi ve dosya oluşturulması ve arama işlemlerinin yapılması; 11. basamakta ise gerekli bilgilerin kuyruğa iletimi yapılır.

- Basamak 12,13,14,15 ve 16'da gerçekleşen işlemler önceki bölümlerde incelediğimiz arama işlemleriyle aynıdır.

```
Imports System.ServiceModel
<ServiceContract(CallbackContract:=GetType(IServisbilgiIste))> _
Public Interface Igizli

    <OperationContract(isOneway:=True)> Sub AboneOl(ByVal kAdi As String, B
    <OperationContract(isOneway:=True)> Sub VeriYolla(ByVal kAdi As String,


End Interface

Public Interface IServisbilgiIste
    'excel de oluşacak metot
    <OperationContract(isOneWay:=True)> Sub Degerler(ByVal id As String)

End Interface

<ServiceContract(CallbackContract:=GetType(ISonBilgiler))> Public Interface

    <OperationContract(isOneWay:=True)> Sub BilgiIste(ByVal id As String)

End Interface


Public Interface ISonBilgiler
    <OperationContract(isOneWay:=True)> Sub SonDegerler(ByVal id As String,

End Interface

<DataContract()> Public Class GizliVeriler
    <DataMember()> Public telefon As String
    <DataMember()> Public mesaj As String
End Class
```

Şekil 5.78 Servis Kontratları

Arka planda gizli bilgilerin yönetilmesini sağlayacak olan servisin kontratlarının bir bölümü Şekil 5.78'de görülmektedir. Burada Gizli Bilgiler Servisi hem istemci uygulamayla hem de arka plan servisiyle çift yönlü iletişim halindedir. Bu çift yönlü iletişim sayesinde bu sistemde gerek istemci gerek arka plan servislerinde oluşacak herhangi bir hata diğerlerinin çalışmasını etkilemeyecektir. Arka plan servisi arama bilgilerini istedikten sonra yanıtını beklemeyip kendi çalışmasına devam etmekte ve yanıt geldiği takdirde yanıttaki bilgilere göre gerekli ses dosyasını oluşturup gerekli bilgileri kuyruğa göndermektedir. Çift yönlü iletişimin nasıl gerçekleştirildiği örnek olarak 5.4.4. bölümde incelenmiştir.

6. ÖNERİLER VE GELECEK ÇALIŞMALAR:

Bu bölümde tez kapsamında değinilen teknolojilerin geniş bir bölümünün böyle bir servis mimarisi içerisine nasıl eklenebileceği incelenmiş ve gerçekleştirilmiştir. Bu teknolojilerin kazanımları öneri olarak da sunulmuştur.

Servislerimizi SOAP protokolü temelinde çalışmakta ve WS-* servisleriyle birlikte kullanım özelliklerini genişletmektedir. Servislerin çeşitli durumlarda REST olarak hazırlanıp sunulmasıyla istemci tarafında daha hafif kodlar kullanılabilir. Böylelikle istemciler basit scirpt kodlarıyla bile servislere erişebileceklerdir. Bu REST istemcileri AJAX uygulamaları içerisindeki script kodlar, Windows Vista Kenar Çubuğu, Flash uygulamaları ve bu tür basit yapılar olabilir. Servislerin REST sürümü servislerin istemci profilini zenginleştirecektir.

SaaS uygulamalarının bilgileri gerektiğinde veritabanlarında gerektiğinde o işletme için oluşturulacak sertifikalarla şifrelenip saklanabilir.

SaaS uygulamalarında işletmeler kendi verilerini tanımlayabilmektedir. Aynı zamanda bu verilerin sistem kodları arasında nasıl işleyeceğini de BPEL kodları ile kontrol ederek, uygulamada kendi iş mantığını ve iş kurallarını oluşturabilir. Böylelikle uygulamayı standartlar çevresinde düzenleyip sanki kendisi için hazırlanmış gibi kullanabilir.

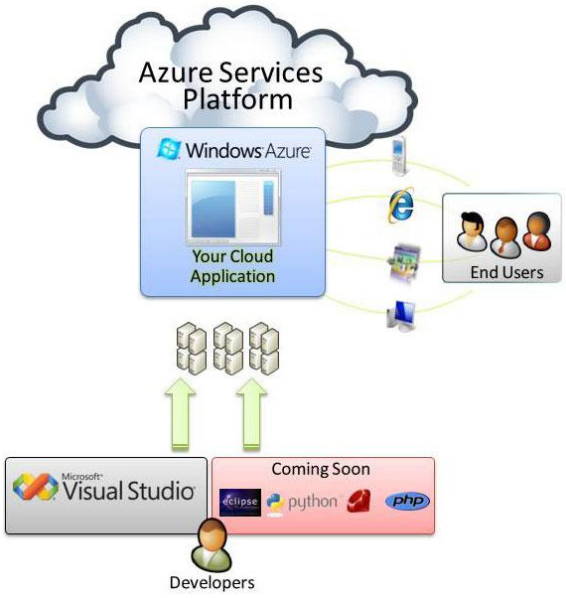
Servislerimizde ses sentezlemesinden önce metinle ses sentezlemesi ile ilgili bilgiler istemciden istenip, buna göre SSML dosyası oluşturulabilir. Böylelikle ses dosyasında kullanıcının istediği vurgular oluşturulur.

Aramanın bir diyalog şeklinde geçekleşmesi için VoiceXML ve ilişkili teknolojiler Telefon Açma Servisinde kullanılabilir. Böylelikle kullanıcının vereceği yanıta göre sistem tarafından yeni ses dosyaları üretebilir. Görüşme bir diyalog şeklinde kişi ile bilgisayar sistemi arasında gerçekleşebilir.

Arama için mutlaka VoIP üzerinden gerçekleştirim zorunlu değildir. Aramalar bunun yerine GSM Gateway adı verilen cihazlara gerekli sim kartlarının yerleştirilmesiyle aramalar bu cihaz üzerindeki sim kartlar aracılığıyla gerçekleşebilir. Böylelikle şirketlerin operatörlerle olan anlaşmalarına göre arama maliyetleri hafifletilmiş olur.

Çift yönlü iletişimde Java istemcileri kullanabilir fakat bunun için Microsoft'un kendisinin belirlediği şemaya uygun elle kod düzenlemeleri yapmak gerekir. Tüm sistemin standart olması istenirse iletişimde WS-Eventing [62] tercih edilebilir.

Yazdığımız servislerin bakımı ve donanımsal giderleri de önemli noktalardan birisidir. Bu noktada servisimizi kullanacak olan işletmeler servis seviyesi anlaşmalarıyla servislerin performansının kesintisiz olarak belirli bir düzeyde olmasını beklemektedir. Bizler de bu tür durumlara karşı servislerimizin yayınlanması bölümünde de bahsettiğimiz gibi servislerimizi belirli bir platformda sunarak donanımsal problemleri ortadan kaldırabiliriz. Bu sayede yük dengeleme, veritabanı performansı, felaket durumu senaryoları gibi işlemleri de hizmet aldığımız platforma devretmiş oluruz. Bizim geliştirdiğimiz servis yapısına en uygun platformlardan birisi de Windows Azure'dur [54]. Microsoft Windows Azure'u Bulut Bilişimi için hazırlanan işletim sistemi olarak tanımlamaktadır. Bu tür platformlarla sunucularımızdaki uygulamaları Windows Azure üzerine taşıyarak, sadece kendi servis kodlarımıza odaklanabilir ve tüm yönetimsel işlemleri bu platforma devredebiliriz. Böylelikle bizim de başlangıç için güçlü sunucular almamız gerekmez.



Şekil 6.1 Microsoft Web Sitesinde bulunan tanıtım resmi [54]

Platformu kendimiz servis olarak alabileceğimiz gibi platform üzerindeki uygulamaları da servis olarak alabiliriz. Örneğin kendi servislerimiz çeşitli hibrit çözümlerde 3.4. bölümde bahsettiğimiz gibi bir ESB içerisinde bulunabilir. Bu servislerimiz ve diğer uygulamalar arasında veri akışını sağlayacak ESB'yi de kendimiz bir SaaS olarak temin edebiliriz. Bu da bu işlemler için ek sunucu maliyeti ve yönetim maliyetini oldukça düşürecektir. Bu servislere örnekler olarak .net Service Bus [30] verilebilir.

7. SONUÇ

Çalışmamız bölümün giriş bölümünde hedeflenen noktalara ulaşmış bir çalışmadır. Uygulamaların ve yazılımların da gerekli gördüğü durumlarda telefon açabileceğini göstermiştir. Bunu yaparken gerekli Servis Yönelimli mimari kazanımlarının SaaS ile birleştirilmesiyle elde edilen sonuçlar da kazanımlara eklenmiştir.

Çalışmamız içerisinde web servisleri, servis yönelimli mimari, uygulamaların servis olarak sunulması, ses hizmetleri gibi konulara değinilmiş ve bu konular hakkında geniş bilgiler sunulmuştur. Burada elde edilen tecrübeler örneğin her voip operatörünün tuş tonu desteği vermediği gibi konular da aktarılmıştır.

Çalışmamız kapsamı oldukça geniş olmakla beraber üzerinde çalıştığımız konularla ayrı ayrı çalışmak isteyenlere yardımcı olabilecektir.

Tez içerisinde SaaS ve SOA konusunda birden fazla öneri olmasına rağmen bu tüm önerileri içeren dağıtık programla ve SaaS içerisinde veri güvenliği konusundaki yaklaşımımızın bu konuda yeni açılımlar sağlayabileceğini düşünmekteyiz.

Öneriler bölümünde değindiğimiz konularla beraber bundan sonraki çalışmalar ve daha kapsamlı gerçekleştirimler bir bulut bilişimi temel alınarak gerçekleştirilecektir.

KAYNAKLAR

[1] Laplante, P.A., Jia Zhang; Voas, J., "What's in a Name? Distinguishing between SaaS and SOA", IT Professional, Volume 10, Issue 3, May-June 2008, Page (s):46 – 50.

[2] Espadas, Javier; Concha, David; Molina, Arturo, "Application Development over Software-as-a-Service Platforms", Software Engineering Advances, 2008. ICSEA '08. The Third International Conference on 26-31 Oct. 2008 Page (s):97 – 104.

[3] Kwok, T.; Thao Nguyen; Linh Lam, "A Software as a Service with Multi-tenancy Support for an Electronic Contract Management Application", Services Computing, 2008. SCC '08. IEEE International Conference on Volume 2, 7-11 July 2008 Page (s):179 – 186

[4] Sathyan, J.; Shenoy, K., "Realizing unified service experience with SaaS on SOA", Communication Systems Software and Middleware and Workshops, 2008. COMSWARE 2008. 3rd International Conference on 6-10 Jan. 2008 Page (s):327 – 332

[5] Liao, HanCheng; Tao, ChangQi, "An Anatomy to SaaS Business Mode Based on Internet", Management of e-Commerce and e-Government, 2008. ICMECG '08. International Conference on 17-19 Oct. 2008 Page (s):215 – 220

[6] Goth, G., "Software-as-a-Service: The Spark That Will Change Software Engineering?", Distributed Systems Online, IEEE Volume 9, Issue 7, July 2008 Page (s):3 – 3

[7] "SaaS Data Architecture,ORACLE" erişim adresi: "http://www.oracle.com/technology/tech/SaaS/pdf/SaaS-data-architecture-whitepaper.pdf". Erişim Tarihi: Haziran 2008

[8] Wang, Zhi Hu; Guo, Chang Jie; Gao, Bo; Sun, Wei; Zhang, Zhen; An, Wen Hao, "A Study and Performance Evaluation of the Multi-Tenant Data Tier Design Patterns for Service Oriented Computing" e-Business Engineering, 2008. ICEBE '08. IEEE International Conference on 22-24 Oct. 2008 Page (s):94 – 101

[9] Dean Jacobs, Stefan Aulbach, "Ruminations on Multi-Tenant Databases", Technische Universität München, Institut für Informatik, Germany, 2007

[10] Sun, Wei; Zhang, Xin; Guo, Chang Jie; Sun, Pei; Su, Hui, "Software as a Service: Configuration and Customization Perspectives" Congress on Services Part II, 2008. SERVICES-2. IEEE 23-26 Sept. 2008 Page (s):18 - 25

[11] Petrie, Charles; Bussler, Christoph, "The Myth of Open Web Services: The Rise of the Service Parks" Internet Computing, IEEE Volume 12, Issue 3, May-June 2008 Page (s):96 – 95

[12] Lawton, G, "Developing Software Online With Platform-as-a-Service Technology", Computer Volume 41, Issue 6, June 2008 Page (s):13 – 15

[13] "Architecture Strategies for Catching the Long Tail", erişim adresi: http://msdn.microsoft.com/en-us/architecture/aa479069.aspx, erişim tarihi: Mayıs 2008.

[14] "Multi-Tenant Data Architect", erişim adresi: http://msdn.microsoft.com/en-us/architecture/aa479086.aspx erişim tarihi: Mayıs 2008

[15] "Software as a Service (SaaS): An Enterprise Perspective", erişim adresi:http://msdn.microsoft.com/en-us/architecture/aa905332.aspx erişim tarihi: Haziran 2008

[16] Mietzner, R.; Leymann, F., "Generation of BPEL Customization Processes for SaaS Applications from Variability Descriptors", Services Computing, 2008. SCC '08. IEEE International Conference on Volume 2, 7-11 July 2008 Page (s):359 - 366

[17] "Efficient Software Delivery Through Service-Delivery Platforms" erişim adresi: http://msdn.microsoft.com/en-us/library/bb735303.aspx. Erişim tarihi: Mayıs 2008.

[18] Pressman Roger, "Software Engineering A Practitioner's Approach", McGraw-Hill Science/Engineering/Math; 6 edition, 2004

[19] "Build a multi-tenant data tier with access control and security", erişim adresi:http://www.ibm.com/developerworks/db2/library/techarticle/dm-0712taylor/, erişim tarihi:Ekim 2008

[20] Bustamante Michele, "Learning WCF", O'Reilly Media, 2007

[21] Erl Thomas, "Service-Oriented Architecture: Concepts, Technology, and Design", Prentice Hall, 2005.

[22] "OASIS SOA Reference Model", erişim adresi: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm, erişim tarihi Ağustos 2008

[23] Pallmann David,"Programming Indigo", MSPress, 2005

[24] Klein Scott, "Professional WCF Programming",Wrox,2007

[25] Lowy Juval, "Programming WCF Services", O'Reilly Media, 2007

[26] Erl Thomas, "SOA: principles of service design", Prentice Hall, 2008.

[27] "MTOM Encoding", erişim adresi: http://msdn.microsoft.com/enus/library/aa395209.aspx , erişim tarihi: Ekim 2008

[28] "Web Services Reliable Messaging", erişim adresi: http://www.ibm.com/developerworks/library/specification/ws-rm/ , erişim tarihi: Eylül 2008

[29] "Microsoft BizTalk ESB Toolkit", erişim adresi: http://msdn.microsoft.com/en-us/library/dd897973.aspx, erişim tarihi: Eylül 2008

[30] ".NET Service Bus", erişim adresi: http://www.microsoft.com/azure/servicebus.mspx, erişim tarihi: Ocak 2009

[31] Eric van der Vlist, Danny Ayers, Erik Bruchez, Joe Fawcett, Alessandro Vernet, "Professional Web 2.0 Programming", Wrox,2006.

[32] "Representational State Transfer" erişim adres: http://en.wikipedia.org/wiki/Representational_State_Transfer , erişim tarihi : Ocak 2009

[33] "SIP", erişim adresi: http://www.voip-info.org/wiki/view/SIP, erişim tarihi Ekim 2007.

[34] Johnston Alan B. "SIP: Understanding the Session Initiation Protocol", "Artech House", 2004.

[35] "Speech Synthesis Markup Language (SSML)" erişim adresi: http://www.w3.org/TR/speech-synthesis/ erişim tarihi: Nisan 2008

[36] "VoiceXML", erişim adresi: http://www.w3.org/2004/03/voicexml20-errata.html , erişim tarihi: Aralık 2008

[37] "Voice Browser Call Control", erişim adresi: http://www.w3.org/TR/ccxml/, erişim tarihi: Aralık 2008

[38] "MSCML", erişim adresi: http://en.wikipedia.org/wiki/MSCML, erişim tarihi: Aralık 2008

[39] Teknoses Yazılım Ses Sentezleme Geliştirici Sürümü, adres: www.teknoses.com

[40] "Microsoft Speech API (SAPI)" erişim adresi: http://msdn.microsoft.com/en-us/library/ms723627 (VS.85).aspx, erişim tarihi: Eylül 2008

[41] "Communications Server 2007 R2 Server SDK Documentation" , erişim adresi : http://msdn.microsoft.com/en-us/library/dd146567 (office.13).aspx erişim tarihi: Ekim 2008

[42] "ConferenceXP", erişim adresi: http://conferencexp.codeplex.com/ , erişim tarihi: Mayıs 2008

[43] "RTC Client API", erişim adresi: http://msdn.microsoft.com/en-us/library/ms775893 (VS.85).aspx, erişim tarihi : Mayıs 2008

[44] "How to use the managed RTP API classes in .NET to create your multicasting systems", erişim adresi: http://www.codeproject.com/KB/IP/Using_RTP_in_Multicasting.aspx ,erişim tarihi: Mayıs 2008

[45] "How to implement impersonation in an ASP.NET application", erişim adresi : http://support.microsoft.com/kb/306158 erişim tarihi Ekim 2008

[46] "Microsoft Message Queuing", erişim adresi: http://www.microsoft.com/windowsserver2003/technologies/msmq/default.mspx erişim tarihi: Ocak 2008

[47] Across Communicatiıons, web adresi http://www.acrosscommunications.com/, erişim tarihi : Ocak 2008

[48] Phone Notify, erişim adresi: http://www.cdyne.com/products/phone-notify.aspx , erişim adresi: Ocak 2009

[49] "Windows Communication Foundation", erişim adresi: http://msdn.microsoft.com/en-us/netframework/aa663324.aspx , erişim tarihi: Haziran 2008

[50] CRM-Salesforce" ,erişim adresi: http://www.salesforce.com/ , erişim tarihi: Nisan 2008

[51] "Win-SaaS", erişim adresi: http://www.winSaaS.com/ , erişim tarihi: Mayıs 2008

[52] "Litware-HR" erişim adresi: http://www.codeplex.com/LitwareHR, erişim tarihi: Mart 2008

[53] "MSDN Office Developer Center", erişim adresi: http://msdn.microsoft.com/tr-tr/office/default (en-us).aspx, erişim tarihi: Ocak 2008

[54] "Windows Azure Service Platform" erişim adresi: http://www.microsoft.com/azure/ , erişim tarihi Ocak 2009

[55] "Ws-MetadataExhange", erişim adresi : http://www.w3.org/TR/2009/WD-ws-metadata-exchange-20090317/, erişim tarihi: Nisan 2009

[56] "Web Services Addressing (WS-Addressing)" erişim adresi: http://www.w3.org/Submission/2004/SUBM-ws-addressing-20040810/, erişim tarihi:Mayıs 2008

[57] "Web Services Policy", erişim adresi: http://www.w3.org/Submission/WS-Policy/ erişim tarihi: Mayıs 2008

[58] "Ws-Security", erişim adresi: http://www.oasisopen.org/committees/tc_home.php?wg_abbrev=wss , erişim tarihi: Ocak 2009

[59] "Web Services Reliable Messaging" , erişim adresi: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrm, erişim tarihi: Mayıs 2008

[60] "Call Centric", erişim adresi: http://www.callcentric.com/ erişim tarihi: Kasım 2009

[61] SIP Software Development Kit (SIP SDK) erişim adresi: http://www.pcbest.net/ erişim tarihi:Ocak 2009

[62] "Web Services Eventing", erişim adresi http://www.w3.org/Submission/WS-Eventing/ , erişim tarihi: Ocak 2009

[63] ÖZTOPUZ G, ÇORUH U "ONDOKUZ MAYIS ÜNİVERSİTESİ VoIP ve MOBİL ÇÖZÜMLERİ",Akademik Bilişim Konferansı Şubat 2007, Dumlupınar Üniversitesi, Kütahya

[64] "Microsoft Visual Studio 2008", erişim adresi http://msdn.microsoft.com/en-us/vstudio/default.aspx erişim tarihi, Kasım 2007

[65] Roy Thomas Fielding , "Architectural Styles and the Design of Network-based Software Architectures" REST Doktora tezi, erişim adresi http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm , erişim tarihi Kasım 2008

[66] "Software as a Service: Strategic Backgrounder" erişim adresi http://www.siia.net/estore/ssb-01.pdf , erişim tarihi: Haziran 2009

[67] "SaaS Licencing" , erişim adresi : http://blogs.technet.com/matt_deacon/archive/2007/02/06/2-saas-licensing.aspx , erişim tarihi: Haziran 2009

[68] "Communications as a Service" , erişim adresi http://msdn.microsoft.com/en-us/library/bb896003.aspx , erişim tarihi: Eylül 2008

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı	: ÖZTOPUZ, Gökhan
Uyruğu	: T.C.
Doğum tarihi ve yeri	: 18.12.1977 Samsun
Medeni hali	: Bekar
Telefon	: 0 (362) 431 95 53
Faks	: 0 (362) 230 94 96
e-mail	: gokhanoztopuz@hotmail.com

Eğitim

Derece	Eğitim Birimi	Mezuniyet tarihi
Lisans	Ondokuz Mayıs Ünv./Bilg. Ve Öğrt.Tek	2002

İş Deneyimi

Yıl	Yer	Görev
2002-2009	Milli Eğitim Bakanlığı-Samsun	Öğretmen
2006-2009	Ondokuz Mayıs Ünv.	Öğretim Görevlisi

Yabancı Dil

İngilizce