T.C.
VAN YÜZÜNCÜ YIL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ
ANABİLİM DALI

SERVİS ODAKLI MİMARİ İLE TAŞINIR KAYIT TAKİP UYGULAMASININ GELİŞTİRİLMESİ

YÜKSEK LİSANS TEZİ

HAZIRLAYAN : Emine DOĞAÇ
DANIŞMAN : Doç. Dr. Rıdvan SARAÇOĞLU

VAN-2019

T.C.
VAN YÜZÜNCÜ YIL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ
ANABİLİM DALI

SERVİS ODAKLI MİMARİ İLE TAŞINIR KAYIT TAKİP UYGULAMASININ GELİŞTİRİLMESİ

YÜKSEK LİSANS TEZİ

HAZIRLAYAN: Emine DOĞAÇ

VAN-2019

KABUL VE ONAY SAYFASI

Elektrik-Elektronik Mühendisliği Anabilim Dalı'nda Doç. Dr. Rıdvan SARAÇOĞLU danışmanlığında, Emine DOĞAÇ tarafından sunulan "Servis Odaklı Mimari İle Taşınır Kayıt Takip Uygulamasının Geliştirilmesi" isimli bu çalışma Lisansüstü Eğitim ve Öğretim Yönetmeliği'nin ilgili hükümleri gereğince 05/07/2019 tarihinde aşağıdaki jüri tarafından oy birliği ile başarılı bulunmuş ve Yüksek Lisans tezi olarak kabul edilmiştir.

Başkan: Doç. Dr. Rıdvan SARAÇOĞLU İmza:

Üye: Dr. Öğr. Üyesi Murat KÖKLÜ İmza:

Üye: Dr. Öğr. Üyesi A. Oğuz KIZILÇAY İmza:

Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 26/07/2019 tarih ve 2019/40-1 sayılı kararı ile onaylanmıştır.

İmza

.....

Enstitü Müdürü

PROF.DR.SUAT ŞENSÖZ
Enstitü Müdürü

TEZ BİLDİRİMİ

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

(İmza)
Emine DOĞAÇ

ÖZET

SERVİS ODAKLI MİMARİ İLE TAŞINIR KAYIT TAKİP UYGULAMASININ GELİŞTİRİLMESİ

DOĞAÇ, Emine
Yüksek Lisans Tezi, Elektrik-Elektronik Mühendisliği Anabilim Dalı
Tez Danışmanı: Doç. Dr. Rıdvan SARAÇOĞLU
Temmuz 2019, 43 sayfa

Günümüzde, var olan kurumsal uygulamalar ile yeni geliştirilen veya satın alınan uygulamaların birlikte çalışabilirliğini sağlamak organizasyonların en fazla yatırım yaptığı alanlardan biridir. Otomasyonların kurum ve kuruluşlara en büyük maaliyet kalemleri; güvenlik, donanım, yeniden kullanılabilirlik ve lisanslamalardır. Otomasyon sistemi, onu kullanan işletmelerin ihtiyaçlar doğrultusunda şekillendirebileceği esnek bir tasarımla üretilmelidir. Geleneksel yazılım mimarisi ve platform bağımlı çalışan otomasyon sistemlerinden farklı olarak Servis Odaklı Mimari ile diğer projelerle etkileşime hazır bir yapı oluşturulabilmektedir. Bu çalışmada örnek bir uygulama alanı üzerinde hem geleneksel yazılım mimarisi hem de Servis Odaklı Mimari yapısına uygun bir çözüm geliştirilmiştir. Hazırlanan bu iki uygulama avantaj ve dezavantaj açısından birbiriyle karşılaştırılmıştır.

Anahtar kelimeler : Kayıt takip, Proje geliştirme, Servis odaklı mimari, Servis geliştirme

ABSTRACT

DEVELOPİNG MOVABLE RECORD TRACKİNG APPLİCATİON WİTH SERVİCE ORİENTED ARCHİTECTURE

DOĞAÇ, Emine
M. Sc., Electrical-Electronics Engineering
Supervisor: Assoc. Prof. Dr. Rıdvan SARAÇOĞLU
July 2019, 43 pages

Nowadays, providing interoperability between existing corporate applications and newly developed or purchased applications is one of the areas in which the organizations invest most. The biggest cost of automation to institutions and organizations; security, hardware, reusability, and licensing. The automation system should be produced with a flexible design that can be tailored to the needs of the enterprises using it. Unlike traditional software architecture and platform-dependent automation systems, a service oriented architecture can be built to interact with other projects. In this study, a solution has been developed on a sample application area in accordance with both traditional software architecture and Service Oriented Architecture. These two applications were compared in terms of advantages and disadvantages.

Keywords: Project Development, Record Tracking, Service Oriented Architecture, Service Development

ÖN SÖZ

Tezimin tüm aşamalarında hiç bir desteğini esirgemeyen danışmanım Sayın Doç. Dr. Rıdvan SARAÇOĞLU'na yürekten şükranlarımı sunmak istiyorum. Ayrıca aileme ve bana yardım eden değerli arkadaşlarıma teşekkür etmek istiyorum.

2019

Emine DOĞAÇ

İÇİNDEKİLER

Sayfa

Sayfa

ÇİZELGELER LİSTESİ

Çizelge **Sayfa**

ŞEKİLLER LİSTESİ

Şekil **Sayfa**

SİMGELER VE KISALTMALAR

Bu çalışmada kullanılan bazı kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

Kısaltmalar	Açıklama
CRUD	Create Read Update Delete
SOM	Servis Odaklı Mimari
APT	Gelişmiş Paketleme Aracı
BT	Bilişim Teknolojileri
IIS	Internet Information Services
HTML	HyperText Markup Language
MSSQL	Microsoft SQL Server
OGC	Web Map Service
SAP	System, Application & Products
SOAP	Simple Object Access Protocol
GYM	Geleneksel Yazılım Mimarisi
HTTP	Hiper-Metin Transfer Protokolü
ASP	Etkin Sunucu Sayfaları
IOS	İphone Operating System
PHP	Hypertext Preprocessor
XML	Extensible Markup Language
YM	Yazılım Mimarisi
WCF	Windows Communication Foundation
IIS	İnternet Information Services
JSP	Java Server Pages
CGI	Common Gateway Interface
VEK	Veri Erişim Katmanı
KAK	Kullanıcı Arayüzü Katmanı
İKK	İş/Kural Katmanı

1. GİRİŞ

Elektronik cihazların, tanımlanmış bir işi gerçekleştirebilmeleri için bilgisayar dilinde hazırlanan komutlar bütününe program denir. Yazılım ise dijital parçaların birbirleriyle iletişim halinde olarak, kullanan bireylerin iş verimliliğini arttırmasını, zaman tasarrufu yapmasını ve iletişimi kolaylaştırmasını sağlayan programlar bütünüdür. Yazılım üç farklı kategori altında toplanabilir. Bunlar; web yazılımı, masaüstü yazılımı ve mobil yazılımdır. İnternet ve akıllı cihaz kullanımının arttığı günümüzde masaüstü yazılım yerini mobil yazılım ve web yazılımına bırakmaktadır. Mobil yazılım ve web yazılım popüleritesini arttırmasının diğer bir sebebi de kolay kurulumu ve kullanıcısının ihtiyaçlarını hızlı ve net bir biçimde karşılamasıdır (Yeren, 2016).

İşletim sistemi yazılımları bilgisayar, tablet, akılı telefonlar gibi teknolojik aletlerin açıldığı anda itibaren, kapandığı zamana kadar görev yapan geniş çaplı yazılımlardır. Yani bu yazılımlar bilgisayar, tablet ve akıllı telefonların çalışabilmesi için gerekli olan temel sistemleridir. İşletim sistemi, kullanılan teknolojik aletler açılır açılmaz devreye girer ve otomatik olarak belleğe verileri yükler. İşletim sistemi bulunmayan, bilgisayar, akılı telefon veya tablet gibi aletlerin çalışması mümkün olamaz. Yani işletim sistemi yazılımları olmazsa, donanımlar çalışmamaktadır ve yapmak istenen hiçbir işlem gerçekleştirilmemektedir. İşletim sistemi yazılımların örnek verilmesi gerekirse; Windows 95/98/XP/7/8/9/10, Vista, Pardus, Linux, MacOS gibi bilgisayar sistemleri ve IOS, Windows Mobile, Android gibi akılı telefon ve tablet işletim sistemleri yazılımları bulunmaktadır (Anonim, 2015a).

Uygulama yazılımları, işletim sistemi yazılımları ile uyumlu bir şekilde çalışan ve bilgisayar, akılı telefon, tablet gibi uyumlu elektronik aletlere yüklenebilen, çalışan programlardır. Uyumlu olduğu sistem yazılımları ile entegreli ve bağlı bir şekilde çalışmaktadırlar (Anonim, 2015a).

Programlama yazılımları, işletim yazılımlarının ve uygulama yazılımlarının oluşturulması, bu oluşturulan yazılımlar ile uyumlu, kullanılabilir elektronik veya teknolojik aletler içerisine yüklenmesi ve kullanılmaya hazır hale getirilmesi için gerekli olan programlardır (Anonim, 2015b).

Web yazılımı, web sunucuları içerisinde çalışan web sitesinin tüm fonksiyonlarının oluşturulmasıdır. Web yazılımı, sadece server adı verilen web sunucuları üzerinde faaliyet gösterir ve server sayesinde gerekli tüm bilgiler internete aktarılır. Kısmen ya da tamamen web teknolojileri kullanılır. Kısacası web yazılım, yazılımın web ara yüzünden çalıştırıldığı, web siteleriyle ilgili veri alma işlemlerinin yapıldığı, web sitelerinin arka planda çalışan kodlama sistemidir. Software, yazılımın tüm dünyada ortak kullanılan adıdır. Yazılımların tamamında web teknolojileri kullanılarak bütün tarayıcılarda sorunsuz çalışmaları sağlanır (Anonim, 2014a).

Web yazılımı, aynı zamanda web programlama olarak da adlandırılır. Web yazılımı ile ilgili bilinmesi gereken en önemli husus programlama dilidir. Programlama dili bilgisayara ne yapması gerektiğini söyleyen komutlar bütünüdür. Bilgisayar, yapılması istenilen işlemin makine dili ile yazılmasını ister. Ancak bu işlemin bir programcı tarafından yapısal bir dil aracılığı ile bilgisayara çevrilmesi gerekir. Bu makine diline çevirme işlemine derleme (compile)ya da yorumlama (interpriting) denir. Web yazılım projelerinde C#, VB.NET, PHP, JAVA, ASP.NET, XML gibi web yazılım dilleri kullanılır. Sistem, yazılım makine dilini yazı diline çevirir. Ülkemizde en çok kullanılan PHYTON VE PHP web dilleridir. Kullanılan ASP dili ise yavaş yavaş kullanımdan kalkmaktadır. Teknolojinin hızla gelişmesi, web yazılım ve web teknolojisinin de değişiklikler göstermesine ve yeni dillere yol açmaktadır (Anonim, 2014b).

Kullanıcı arabirimine sahip uygulama yazılımı yapmak isteniyorsa arayüzü sürükle bırak mantığı ile daha kolay ve hızlı yapılabildiği için C#, Visual Basic veya Java tercih edilebilir (Anonim, 2018).

Yazılım mimarisi yapı ve davranışların yanı sıra kullanılabilirlik, fonksiyonalite, performans, esnekllik, yeniden kullanım, anlaşılırlık, ekonomik ve teknolojik kısıtlar gibi özellikleri de yansıtır. Tüm bu özellikleri ile “Yazılım Mimarisi” sistemin anayasasıdır. Tüm yazılım geliştirme sürecinin merkezinde durur ve her türlü faliyete kılavuzluk eder. Yazılım geliştirme süreci oldukça karmaşık ve zor bir süreçtir. Bu süreçte dağılmamanızı sağlıyacak şey, sürecin merkezinde duran şey yazılımınızın mimarisi olmalıdır (Dirik, 2012)

Tüm uygulamlar için atik, çevik, değişen ve dönüşebilen yeni taleplere kısa sürede adaptasyon sağlayabilen, heterojen yapılar arasında iletişimi sağlayabilecek bir alt yapı gereksinimi duyulmaktadır.

Servis Odaklı Mimari (SOM) uzun vadede uygulamalar arasındaki entegrasyon maliyetlerini azaltmayı, kurumda iş süreçlerini kısaltmayı ve ekonomik maliyetleri azaltmayı sağlar. En önemlisi ise daha fazla kod ve tekrar tekrar kod yazmayı engeller. Uygulama sanallaştırmanın çalışma prensibinden gelen bu özellikler sayesinde bilgi, doğru kullanıcılara, doğru zamanda ve hızda ulaştırılmaktadır. Böylece bilgi güvenirliğine de çok iyi derece de katkı sağlamaktadır.

İşletme giderlerini büyük oranda düşürerek, işletmenin buradan yapmış olduğu tasarrufu bilişim sistemini geliştirmek için değerlendirmesi için olanak sağlar.

SOM'un elbette bilişim teknolojileri alt yapısındaki yazılımsal vizyonu dışında bir de iş birimine sağladığı faydalar söz konusudur. İş birimi açısından SOM'un faydaları ise şunlardır; servisler özerk tasarlandığı için işlem maliyeti düşüktür (Autonomy özelliği), Endüstriyel olarak standartlaşmış ilkeleri önerdiği için entegrasyon kolaydır, Bileşenlerin bakımı ve değişimi kolay olduğu için iş süreçlerindeki değişikliklere kolayca adapte olunur. Pazara yeni iş fonksiyonelliklerini hızla sunabilme imkanı vardır, Endüstri standartlarına uyan servis arayüzleri söz konusu olduğu için yeni sistemlere az eforla bağlanmak mümkündür ve iş ortağı firmalarla entegrasyon daha kolaydır (Şenyurt, 2015).

1.1. Amaç ve Tezin Önemi

Birden fazla otomasyon sistemi barındıran kurum/kuruluşlarda uygulamaların kendi içindeki modüllerinin/fonksiyonlarının başka uygulamalar tarafından kullanılabilecek şekilde tasarlandığı SOM yaklaşımını kullanmak amaçlanmıştır.

Otomasyon hizmetlerinin kolaylaştırılması, standardize edilmesi, müşteriye özgün çözümlerin kolayca modellenmesi/analizi, SOM'un kullanılması ile maliyetlerin aşağıya çekilmesi, müşterilerin Bilişim Teknoloji (BT) bölümlerinin otomasyon üzerindeki hâkimiyetlerinin artırılması, müşteriye kendi içinde yeni projelerin üretilmesine

destek olacak altyapının oluşturulması ve müşterilerin farklı firmalara ait programlarla platform bağımsız bir şekilde bütünleşme imkânlarının yaratılması amaçlanmıştır.

Kurum ve kuruluşlar uzun süre BT bölümlerini, içine işletme kaynaklarının akıtıldığı, yani daha fazla masraf kalemi olarak algılamışlardır. Genellikle yatırımın geri dönüşümü pozitif görülmüştür. Tahakkuk, tahsilat, sipariş, iş akışı, personel ve halkla ilişkiler gibi standart süreçlerin otomatikleştirilmesi, ölçeklendirme ve verimlilik kavramları konusunda önemli faydalar sağlanmıştır. Kâğıt ve personel yığınları, yüksek kapasiteli, her şeyi düzenli ve sabit tutan ana bilgisayar (Main Frame) ve sunucularla yer değiştirmiştir. Değişimin az veya kontrol edilebilir bir çevrede bu sorunsuz olarak görülen bir durum iken değişen dünya ve piyasalara adapte olmak isteyen kurumlarda, büyük ve ivmesi artan değişimler BT bölümlerini zorlamaya başlamıştır. Yönetimi zor olan, farklı teknolojileri ve firma ürünlerini kullanan, bir birinden kopuk bölümler, işlevler yeni veri siloları ortaya çıkarmaya başlamıştır. BT bölümlerinin oluşturacağı yeni projeler için de, oluşan bu yeni veri silolarına da erişmekte zorlanmakta dolaysıyla her yeni iş için yeniden kural motorlar, yeniden arayüz, yeniden analiz ve yeniden veri siloları oluşturmak zorundadır. BT bölümlerine yatırımın miktarı artarken geri dönüşümler azalmaya başlamıştır. BT bölümleri, kurumlarda değer yaratandan çok masraf yaratan algısı oluşturmuştur.

SOM ile kurumların kullanmış olduğu farklı platformlar için geliştirilen yazılımların tek bir platformda toplaması ile hem yazılımların takipleri kolaylaşacak hem de maliyet düşecektir.

Otomasyonlarda, kurum ve kuruluşlar içinde değişen ihtiyaçlara cevap verebilmek ve gerçek bir talebe dayalı yapıya sahip olmak için temel şart özellik; esneklik ve çeviklik özellikleridir. Bu iki hayati özellik SOM kavramı ile BT dünyasında ilk cevabını bulabilmiştir.

Dinamik tekrar yapılandırılabilir, tek başına veya başka servislerle bir arada kullanılabilen, yani birleşik işletme sorunlarını talebe dayalı çözmek için tasarlanmış yazılım mimarisi olan SOM ile oluşturacağımız otomasyonda, heterojen yapıya sahip kurumların otomasyonları arası haberleşme sağlanmış olacaktır. Bu yapıda her iş parçacığı bir servisin metodları olarak tasarlanacak ve kurumların diğer projelerinde de entegre olabilecektir. Küçük iş parçacıklarının servislere dönüştürülmesi ile tekrar

kullanılabilir bir yapı elde edilmiş olacaktır. Bu sebepten bakım maliyetleri çok aşağı seviyelere indirilmiş olacaktır. Veritabanı katmanı sadece servislerin güvenlik protokolleri üzerinden erişilerek maksimum güvenlik sağlanacak ve veritabanına BT bölümlerinin da erişiyor olmalarına olanak tanınarak büyük ve birbirinden bağımsız veri silolarından kurtulmuş olunacaktır. Geliştirilmesi planlanan bu sistem, servis odaklı esnek ve çevik bir sistem olacaktır.

SOM'un önemli özelliklerinden biri olan platform bağımsız özelliği ile oluşturulacak veritabanı iş ve servis katmanına servis katmanı üzerinde kullanıcı ve kurum yetkisi gözlenilerek bağımsız platformda erişiliyor olabilecektir.

Geliştirilmesi planlanan taşınır kayıt takip uygulaması ile kurum içine alınan tüm taşınırların ilk girişinden son kullanıcıya kadar takip edilmesinin sağlanması, taşınırların kuruma olan maliyetlerinin belirlenmesinin alyapısının oluşturulması, taşınırlarda doğacak sorunlarının/tamirlerinin kurum içerisinde birimlere/birim personellerine iş emri olarak aktarılmasının sağlanması, bu iş emirlerinin ilerleme durumlarının gözlemlenebilmesi ve aynı zamanda bu iş emirleriyle personellerin performanslarının tespitinin alt yapısının oluşturulması amaçlanmıştır.

2. KAYNAK BİLDİRİŞLERİ

Bu kısımda SOM üzerine yapılan çalışmalar, elde edilen sonuçlar ortaya konulacaktır. SOM üzerine yapılan çalışmaların büyük bir çoğunluğu İngilizce dilinde yapılan çalışmalardır. SOM konusu aslında çok da yeni bir konu değildir. 1980'lere baktığımızda bu konu nesne yaklaşımlı geliştirme çalışmaları ile başlamış ve 90'lı yıllarda portallerin de ortaya çıkışı ile SOM yaklaşımı şekillenmeye başlamıştır. Son dönemlerde ise SOM teknolojik altyapı olarak iş akışı ve süreç yönetim araçları ile ele alınan bir konu olarak karşımıza çıkmaktadır. Bu yöntemlerin tercih edilmesinin nedeni verimliliğinin ve başarının yüksek, maliyetlerin düşük olmasıdır.

Channabasavaiah ve ark. (2004) çalışmalarında kurum yöneticilerinin kurumsal kaynakların daha verimli kullanılması için bastırdığı fikrini ortaya atarak SOM'un bu probleminin en büyük çözümlerinden birisi olduğunu desteklemişlerdir.

Srinivasan ve Treadwell (2005) çalışmalarında web servisleri ve SOM tarzı, yeni bir temel olarak görmüştür SOM'nin temel kavramlarını, ilişkilerini ve yararlarını anlatmıştır.

Lee ve ark. (2005) SOM yapısının bir çok servisin bir araya gelerek oluştuğunu ve entegre uygulamlar oluşturmanın SOM yapısına dayandığı fikirini ortaya çıkarmışlardır.

Beklen (2006) çalışmasında SOM'un kendine özgü kuralları olan, finans, sigorta ve kamu gibi farklı alanlarda hayata geçirilmiş bir yaklaşım olduğunu ve servislerin iyi tanımlanabilmesi için detaylı bir analize ve modellemeye ihtiyaç duyduğunu belirtmiştir. SOM'un, tüm dünyada farklı sektörler için başarıyla uygulamaya alınmış olması mimarinin olgunluk seviyesini de yukarılara taşımış bulunmaktadır. Uygulama sorunlarına, katmanlar da soyutlama ile çözüm bulması ve platform bağımsız entegrasyon yöntemleri kullanıyor olması mimarinin en önemli avantajlarından olduğu belirtilmiştir. Büyük ölçekli yazılım projeleri, çeşitli sektörlerde farklı iş problemlerini çözümlemek üzere yazılım mühendisliği disiplinleri uygulanarak geliştirilmektedir. Ancak her kurum hayata geçirdiği uygulamaların alt yapılarında farklı teknoloji ve yaklaşımlar kullanmıştır. Bazı projelerde ise herhangi bir mimari yaklaşım kullanılmadan uygulamalar geliştirilmiştir. Bu tip durumlar zaman içinde daha da büyüyen ve karmaşıklaşan uygulamaları, birbirleriyle entegre olamaz, bakımı güçlükle yapılan, tekrarlanan işlerin yapıldığı

veya platformlara bağlı uygulamalara dönüştürmüştür. Bu tip sorunları, uygulama katmanlarını birbirinden soyutlayarak çözmeyi hedefleyen SOM, yaklaşımı, mimari, servis ve tanım gibi alt başlıklarda incelenmiştir.

Üstündağ (2006) çalışmasında günümüzde, var olan kurumsal uygulamalar ile yeni geliştirilen veya satın alınan uygulamaların birlikte çalışabilirliğini sağlamanın organizasyonların en fazla yatırım yaptığı alanlardan biri olduğunu söylemiştir. Bu çalışmasının sonucunda ortaya koyduğu birbirleri ile entegrasyona giren sistemler örnekte .Net ile geliştirilmiş bir uygulama ve var olan bir SAP uygulamasıdır. Kurumun büyüklüğü, iş hacmi ve süreçlerinin karmaşıklığına göre birbirleri ile etkileşim içerisinde olan uygulamaların sayısının arttığı düşünülecek olursa bu tip entegrasyon katmanlarının farklı katmanlar olarak geliştirilmesinin önemi daha iyi biri şekilde anlaşılabileceğini savunmuştur.

Jeng ve An (2007) çalışmalarında özellikle geniş çaplı SOM tabanlı bir projenin System Dynamics modelleme metodolojisi ile daha başarılı olduğunu savunmuşlardır. System Dynamics iş dünyası hedefleri ve sosyal sistemler gibi karmaşık geri beslemeli ortamları inceleyen ve yöneten bir metodolojidir. Bu çalışmada System Dynamics modellemeye dayanan çeşitli SOM senaryo analizleri yapılmıştır. Bu farklı çalışmalar SOM'nin değişik faktörlerin toplam etkisinin nasıl bir etkiye sahip olduğunu ve bunların uzun vadede ne tür sonuçlar vereceğini ele almaktadır. SOM'nin iki önemli özelliği olan hızlı geliştirme ve birçok çözüm dokuları ve geliştirmelerde yüksek derecede yeniden-kullanılabilirlik olarak belirlemişlerdir.

Altınbaş (2008) çalışmasında SOM'da amacın, ortak olarak kullanılabilecek mantıksal kodları bir araya getirebilmek ve bunları servis bağlamında bir ağ üzerinden diğer uygulamaların kullanımına açabilmesi olduğunu belirtmiştir. Doğal olarak böyle bir sorgulamada, sorgulara daha etkin yanıtlar alınabilecektir. SOM, işlevsel olarak iş süreçleri ve bilgi işlem altyapılarının kesişim noktalarında konumlanır. Tez kapsamında açıklanmaya çalışılan bu projenin önemi, tasarıma SOM yaklaşımı kullanılarak erişim olanağı yaratmaktır. Çünkü projede SOM'un kullanılması, Oracle platformunun SOM tamamlayıcı bir altyapı hizmet kümesi sunabilmesidir. Benzer şekilde iş süreci otomasyonunda sağladığı esneklik yanında düşük maliyet ve hızlı oluşu, bu teknolojinin avan-

tajları arasındadır. Aynı şekilde, Oracle'nin SOM mantığı ile tasarlanmış olması yanında, SOM'a olan erişimde ortaya koyduğu kolaylıklar gösterildiği fikrini savunmuştur.

O'Brien (2009) çalışmasında bir SOM projesinin yönetimi sırasında kapsam, maliyet ve harcanacak çaba tahminleri ile ilgili çalışmalardan olan yönetimsel aktivitelerin SOM projelerinde kullanılmasını önermektedir. Teknik, sosyal, kültürel ve kurumsal açıların yanında organizasyonun tecrübesi ile birlikte olgunluk seviyesini de inceleyerek, birbirinden farklı SOM projelerinin tiplerini de göz önüne alarak kapsam, maliyet ve çaba kestirilmeye çalışılır fikrini savunmuştur.

Kaya (2009) çalışmasında SOM'un özellikleri ve avantajları sayesinde çok değişik alanlarda kullanılmakta olduğunu belirtmiştir. Değişik platformlara çok kolaylıkla adapte edilebilmektedir. SOM'u uygulamalarda kullanmak geliştirme aşamasında zaman açısından çok büyük avantajlar sağlar, ancak sürecin başlangıcında servisleri oluşturmak zaman açısından çok avantajlı değildir. SOM'un en belirgin özelliği tekrar kullanılabilirliğidir, bir servis hiçbir değişiklik yapılmadan değişik uygulamalarda rahatlıkla kullanılabilir. Bir servisin açıklamasına sahip olmak onu kullanmak için yeterlidir. Bir servisi kullanmak için gerekli değişkenleri göndermek yeterlidir.

Kang ve ark (2010) çalışmalarında SOM'un değişen gereksinimlere hızlı bir şekilde cevap verebilmek için kurumsal çözümlerin toplanması, düzenlenmesi ve bakımını kolaylaştırdığını belirtmişlerdir. SOM kullanılan bir uygulamada yeniden kullanılabilirlik üst seviyededir. Oluşturulan her servis farklı projelerde dahi olsa yeniden kullanılabilmektedir. İş gereksinimlerinde bir değişiklik olduğunda bu değişikliğin uygulama tarafındaki etkisi sadece bu serviste olur. Yani ilgili değişiklik sadece bu serviste yapılır ve bu servisi kullanan uygulamalar değişiklikten büyük oranda etkilenmez fikrini ortaya çıkarmışlardır.

Çopur (2011) çalışmasında, SOM'un web gibi bir ağda sunulan hizmetleri kullanan yazılım uygulamaları oluşturmak için bir mimari tarzı olduğunu savunmuşlardır. SOM'daki uygulamalar hizmetlere dayanarak oluşturulur. Servisler arasında gevşek bağlantıyı teşvik eder, böylece yeniden kullanılabilirler. Bir hizmet, iyi tanımlanmış bir iş işlevselliğinin bir uygulamasıdır ve bu tür hizmetler daha sonra farklı uygulamalarda veya iş süreçlerinde müşteriler tarafından tüketilebilir. SOM kullanarak işletmeler, geliştirme maliyetlerinde önemli tasarruflar elde edebilir ve yeni uygulamaların geliştiril-

mesinde mevcut hizmetleri yeniden kullanarak ve yeniden yapılandırarak değişen iş koşullarına hızla adapte olabilir. SOM, önceden yalıtılmış uygulama siloları ve eski sistemler de dahil olmak üzere kurumsal BT kaynaklarının daha iyi entegrasyonunu sağladığını ortaya koymuştur.

Herand (2013) çalışmasında yazılımların arzuladığı esnek ve çevik olabilmeleri için, kırılgan olmayan, taşınabilir, karmaşık olmayan, gereksiz tekrar içermeyen, anlaşılması kolay, düzenli ve kontrol edilebilir bir yapıda geliştirilmeleri gerekmekte olduklarını ifade etmiştir. Arzu edilen bu yapı, değişim etkisinin minimuma indirilmesini hedefleyen SOM'un oldukça dikkatli ve hedeflerinin iyice anlaşıldıktan sonra belirli bir metodolojinin takip edilerek uygulanması ile elde edilebilir. Aksi takdirde SOM'un kullanım şekli okunmadan kullanılmış bir ilaç etkisi yaratabildiğini düşünmüştür. Kurumlarda SOM'a geçiş öncesi kullanılan eski sistemlerin yararlı kısımlarının, yeni geliştirilen SOM tabanlı sistemde de kullanılması sağlanmalıdır. Bu sayede büyük maliyetlerle elde edilmiş olan eski sistemlerin, tamamen kullanılmaz bir hale gelmelerinin önüne geçirilir. Bu çalışmada SOM'un ayrıca uygulama esnasında, strateji belirleme ve seçiminin önemi vurgulanmıştır. Kavramsal servislerin belirlenmesinden sonra, bu servislerin geliştirilebilmesi için stratejilerin belirlenmesi, belirlenen stratejiler arasında teknik ve finansal analizlerin yürütülmesi ve bu analizler doğrultusunda stratejilerden birinin uygulanmasına karar verilmesi sağlanmıştır. Bu şekile karar aşamasında hata yapma ihtimali düşürüldüğünü savunulmuştur. SOM tabanlı servis geliştirme ve yönetme işlemi gerçekleştirirken, SOM yönetim platformu seçiminin önemini vurgulanmıştır.

SOM ve web hizmetleri teknolojisini birleştirmek, gerçek kurumsal uygulama entegrasyonunu mümkün kılar, çünkü web servisleri SOM uygulamalarına farklı bir programlama dilinde yazılan, farklı işletim sistemi üzerinde çalışan ve farklı donanım platformuna yerleştirilen uygulamalarla kolayca bütünleştirme becerisi sağlar. Tüm araştırmalar sonucunda SOM ile birden fazla uygulamanın aynı katmanları ayni servis üzerinden kullanabildiği, bu durumda bir revizyon yapılması gerektiğinde istemci uygulamalara müdahale edilmeden tek bir servis üzerinden değişiklikler yapılabildiği, farklı platformlarda geliştirilen uygulamalar için aynı altyapı kullanılabildiği, bir uygulamanın Windows, Linux, mobil, Mac gibi versiyonları var ise hepsi için aynı altyapı kullanılabildiği, uygulamaların farklı katmanlarını farklı sunucularda barındırarak daha kolay

yönetim ve performans sağlanabildiği ve uygulamalar için uygulama geliştirme arayüzü oluşturarak her türlü uygulamanın entegre olmasının sağlanabildiği sonuçları ortaya çıkmıştır.

Bussler (2013) çalışmasında SOM yapısının verilerin artışı, servis odaklı teknolojilerin gelişimi, entegrasyon çalışmaları, işlem yönetimi, güvenlik ilkeleri ve kurumsal bilgi sistemleri gibi problemlerin çözümü için kolaylıklar sağladığı fikrini ortaya çıkarmıştır.

Kreger (2003) Web servisleri ile SOM servis arayüzlerini tanımlayarak uygulama karmaşıklığını azalttığını, ayrıca, Web hizmetleri, tam zamanlı (just-in-time) entegrasyonu ve eski uygulamaların birlikte çalışabilirliğini sağladığı fikirini ortaya çıkarmıştır.

3. MATERYAL VE YÖNTEM

Bu tez çalışmasında Visual Studio 2017 geliştirme ortamından yararlanılmıştır. Servislerin geliştirilmesi için de WCF mimarisi kullanılmıştır, ASP.net teknolojisi kullanılarak arayüzler geliştirilmiştir. Ayrıca kurum içinde yapılan analizler için balsamiq mockups uygulaması kullanılmıştır. Aşağıda bu tezde kullanılan yöntemler ve aşamalar hakkında bilgiler verilmiştir.

3.1. Mimari Nedir?

Yazılım Mimarisi (YM), bir sistemin ana yapısını gösterdiği ve her bir paydaşın ihtiyacını dikkate aldığı için yazılım geliştirme sürecinin temel taşlarından biridir (Tekinerdoğan, 2014).

Yazılım da mimari yazılım sistemini oluşturan ana katmanların, bileşenlerin, sunucuların ve çeşitli türlerdeki istemci uygulamaların iletişim biçimini göstermek için yapılan çalışmaların hepsini kapsar. Kullanılan teknolojiden bağımsız olarak yapılan bu çalışmalar bir anlamda bütüne bakma çabasıdır denebilir.

Yazılım geliştirme sürecinin ilk adımlarında problem analizi yapılarak sistemde nelerin yapılması gerektiği ortaya konur. İhtiyaçların nasıl karşılanacağı, nasıl çözümler geliştirileceği ise dizayn disiplinin konusudur. Dizayn geliştirme boyunca çözümler için alınan kararlar öncelikle üst seviyede alınır ve giderek detaylandırılır (Dirik, 2012).

YM "sistemi tanımlayan en üst seviye kavramlar" olarak nitelendirilebilir.

YM şunları içerir:

•Sistemin organizasyonu hakkında önemli kararlar

•Sistemin önemli yapısal elementleri ve bunların arayüzleri ile birbirleri arasındaki etkileşim

•Sistemin önemli yapısal ve davranışal elemanlarının altsistem'lere dağılımı (Dirik, 2012).

YM yapı ve davranışların yanı sıra kullanılabilirlik, fonksiyonalite, performans, esneklik, yeniden kullanım, anlaşılırlık, ekonomik ve teknolojik kısıtlar gibi özellikleri de yansıtır. Tüm bu özellikleri ile "YM" sistemin anayasasıdır. Tüm yazılım geliştirme sürecinin merkezinde durur ve her türlü faliyete kılavuzluk eder. Kısaca, yazılım mima-

risi bize sistemin karmaşıklığını yönetmek ve bütünlüğünü korumak için , kontrol edilebilir bir yapı sunar (Anonim, 2014b).

3.2. Servis Nedir

SOM'un temel fikri servistir. Servis, servis anlaşmasıyla yapılabilen iş işlevselliğinin ayrık bir birimi olarak tanımlanır. Servis antlaşması, servis sunucusu ve servis kullanıcısı arasındaki etkileşimi belirtir. Bunlar:

- Servis arayüzü
- Arayüz dokümanı
- Servis kuralları
- Servis kalitesi
- Performans

SOM servisleri çoklu platform ve protokollere izin vererek standart tanımlama dilinde açıklanırlar. Ortak bir arayüze sahip olurlar. Benzer işlemleri birlikte destekleyerek faaliyetlerin yürültülmesini birbirleriyle iletişim kurarak yaparlar (Francis ve ark. 2004).

SOM'da her bir fonksiyon servis olarak tanımlanır (Arsanjani, 2002). Tüm servisler özerk çalışır (Papazoglou ve ark. 2007).

3.3. SOM Nedir

SOM birbirinden farklı servislerin birleşip belirli yapılar oluşturarak uyumlu çalışmasını sağlayan bir yaklaşımdır. Yani servis içinde servistir. Normalde çok katmanlı uygulamalarda hep bir katmanın diğer bir katmanı çağırması gerekir. Bu yaklaşımda hiyerarşinin düzgün olması gerekmektedir. Basit ve klasik bir örnek vermek gerekirse projenizde bir data katmanı bir de business katmanı olduğunu varsayalım. Business katmanı data katmanını çağırarak veri tabanı işlemlerini halleder, tek taraflıdır ve proje bazlı olmuş olur fakat data katmanını bir servis olarak oluşturursanız tek taraflı olmaktan çıkar ve istenilen her yerden kullanıma izin verir. Bu şekilde uygulamanın nereden ve/veya hangi platformda çalıştığının önemi kalmaz. SOM noktadan noktaya entegrasyonlardaki bağımlılıkları ortadan kaldırır (Avcı, 2016).

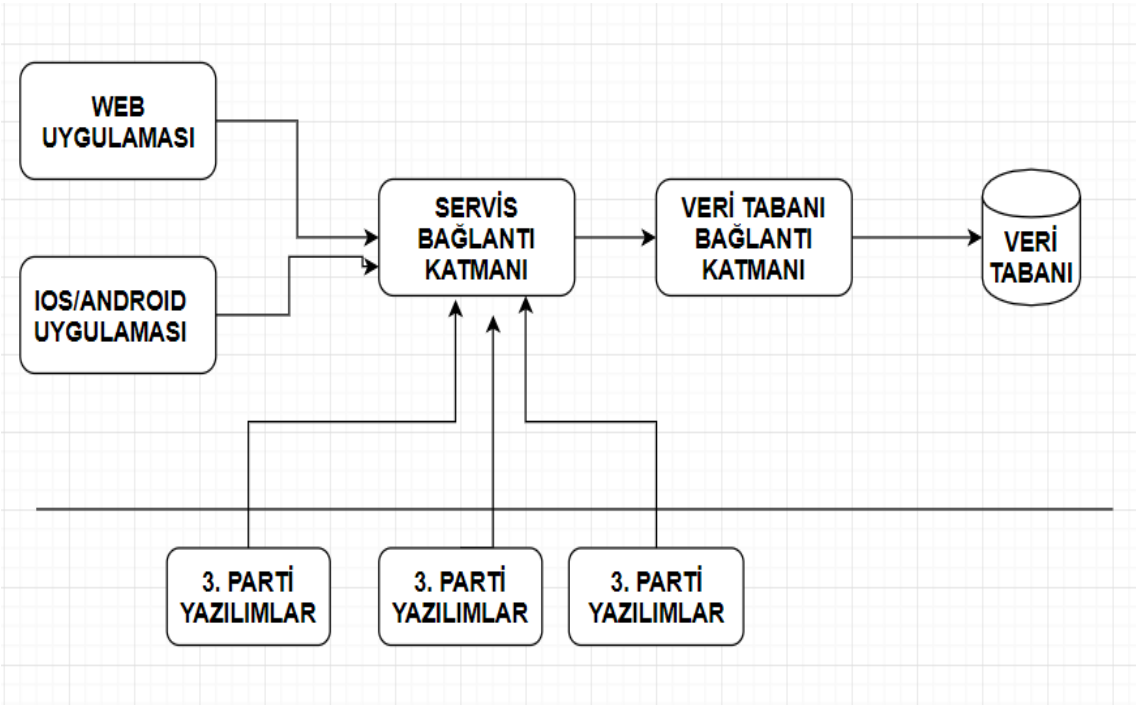
SOM'nin yaşam döngüsü şöyledir; modelle (gereksinimleri belirlemek, tasarlamak, analiz etmek), birleştir (oluşturmak, birleştirmek, test etmek), devreye al (kişiler, süreçler, bilgi entegrasyonu) ve yönet (uygulamaları ve süreçleri yönetmek, süreci izlemek, iş birimi ve bilişim teknolojileri koordinasyonu).

3.3.1. SOM'un amacı

SOM uzun vadede uygulamalar arasındaki entegrasyon maliyetlerini gerek hizmetleri tekrar tekrar kullanarak gerek standartları benimseyip uygulamalar arasında karşılıklı çalışabilirliği artırarak düşürmeyi; ayrıca gerektiğinde hizmetleri lego taşları gibi kullanarak iş süreçleri oluşturmayı ve böylelikle iş ihtiyaçlarını daha hızlı gerçekleştirebilmeyi amaçlar" (Hudayioglu, 2006).

3.3.2. SOM'un karakteristiği

- Servisler keşfedilebilir.
- Servisler kendini içerebilir ve modülerdir.
- Servisler birlikte çalışabilir.
- Servisler düşük bağımlıdır.
- Servisler ağda adreslenebilir bir ara yüze sahiptir.
- Servisler birleştirilebilir.
- SOM değişiklikleri destekler (Şenyurt, 2015).
- Kendi içinde anlamlı bir bütün olan, başka hizmetlerden bağımsız hizmetler
- Devamlı erişilebilir, hizmetlerin davranışları öngörülebilir
- Birleştirme maliyeti olmaksızın kullanıma hazırdır
- Hizmeti kullananlardan bağımsız
- Birden fazla hizmeti bir araya getirerek yeni hizmetler meydana getirme imkanı
- Hizmet kalitesi ölçülebilir
- Reusability (Yeniden Kullanılabilirlik) kodların bir kere yazılıp sonsuz kullanımı.
- Çevik



Şekil 3.1. SOM Genel Yapısı.

Şekil 3.1.' de birden fazla uygulamanın aynı servisleri çağırarak farklı arayüzlerden aynı iş parçalarını (metod) kullanmasının bir örneği gösterilmiştir. Bir seferliğine oluşturulan bu iş parçalarının sayısız uygulama tarafından kullanılmasının bakım ve yazılım maliyetlerine etkisi olmaktadır. Aynı zamanda veri tabanı tüm uygulamalardan ayrıştırılarak oluşturulan bu yapı ile veri tabanının güvenliğini de sağlamıştır. Heterojen yapıya sahip işletme /kurum ve kurululşların SOM ile tek bir veri tabanı kulanması ile birbirinden uyumsuz veri silolarından kurtularak tüm verileri aynı veritabanında barındırmasına olanak sağlamaktadır.

3.4 Veri Tabanı

Veri tabanı kavramı, 1946 yılında ilk bilgisayarın tasarlanması ile şekillenmeye başlamıştır. Yüksek kapasiteli veri işlemlerini yapabilen bilgisayarların çıktılarını saklamak için hard diskler oluşturulduysa da bu veriler ancak klasör olarak saklanabilmekte ve ancak isim listesine göre bulunabilmekteydi. İlerleyen aşamalarda erişilebilirliği arttırmak için her dosyanın içeriğine bir anahtar dizin eklendi, bu eklentiler karışıklığı ve düzeni sağlama açısından başarılı olamadılar, verinin doğru ve düzenli bir yapıda kay-

dedilmesini sağlayan veri tabanı sistemleri, bu istek doğrultusunda geliştirilmeye başlanmıştır.

Veri topluluğu bir "veri tabanı" olarak değerlendirilebilir. Veri tabanı, bir organizasyona ilişkin verilerin saklandığı ortamdır. Veri tabanı yönetim sistemleri ise, veri kümelerinin düzenli bir biçimde tutulduğu ve bu verinin çeşitli yazılımlar aracılığıyla yönetildiği bir ortam olarak düşünülebilir (Özkan, 2008).

3.4.1. Veri modelleri

Veri tabanı yapısının temelini veri modeli kavramı oluşturmaktadır. Model, kavram olarak "gerçek dünyadaki nesnelerin, olayların ve ilişkilerin tasviri" anlamına gelmektedir. Veri modeli ise, bir organizasyondaki verinin tanımlanması, manipüle edilmesi, ilişkilerin kurulması ve kısıtlamaların belirlenmesini birlikte sağlayan kavramdır. Bir başka deyişle veri modeli, veriyi mantıklı düzeyde düzenlemek için kullanılan kavramlar, yapılar ve işlemler topluluğudur (Özkan, 2008).

3.4.2 Veri tabanı yönetim sistemleri yazılımları

Microsoft SQL Server

Microsoft firmasının bir ürünü olan Microsoft SQL Server (MSSQL), iyi bir performansa sahiptir. En büyük dezavantajı, sadece Windows üzerinde çalışabilmesidir. Kullanım kolaylığı, güvenilirliği ve işlem gücüyle dikkat çekmektedir. Maliyeti diğer veri tabanlarına göre yüksektir. Tablo başına 4 TB veri depolayabilmektedir. "Hareket Kilitleme", "tetikleme" ve "saklı prosedür" özelliklerine sahiptir (Gündüz, 2002).

Microsoft Sql Server Management Studio programı ile Microsft Sql Server'a bağlanıp veri tabanları üzerinde işlemler yapılabilir.

3.5 WCF (Windows Communication Foundation)

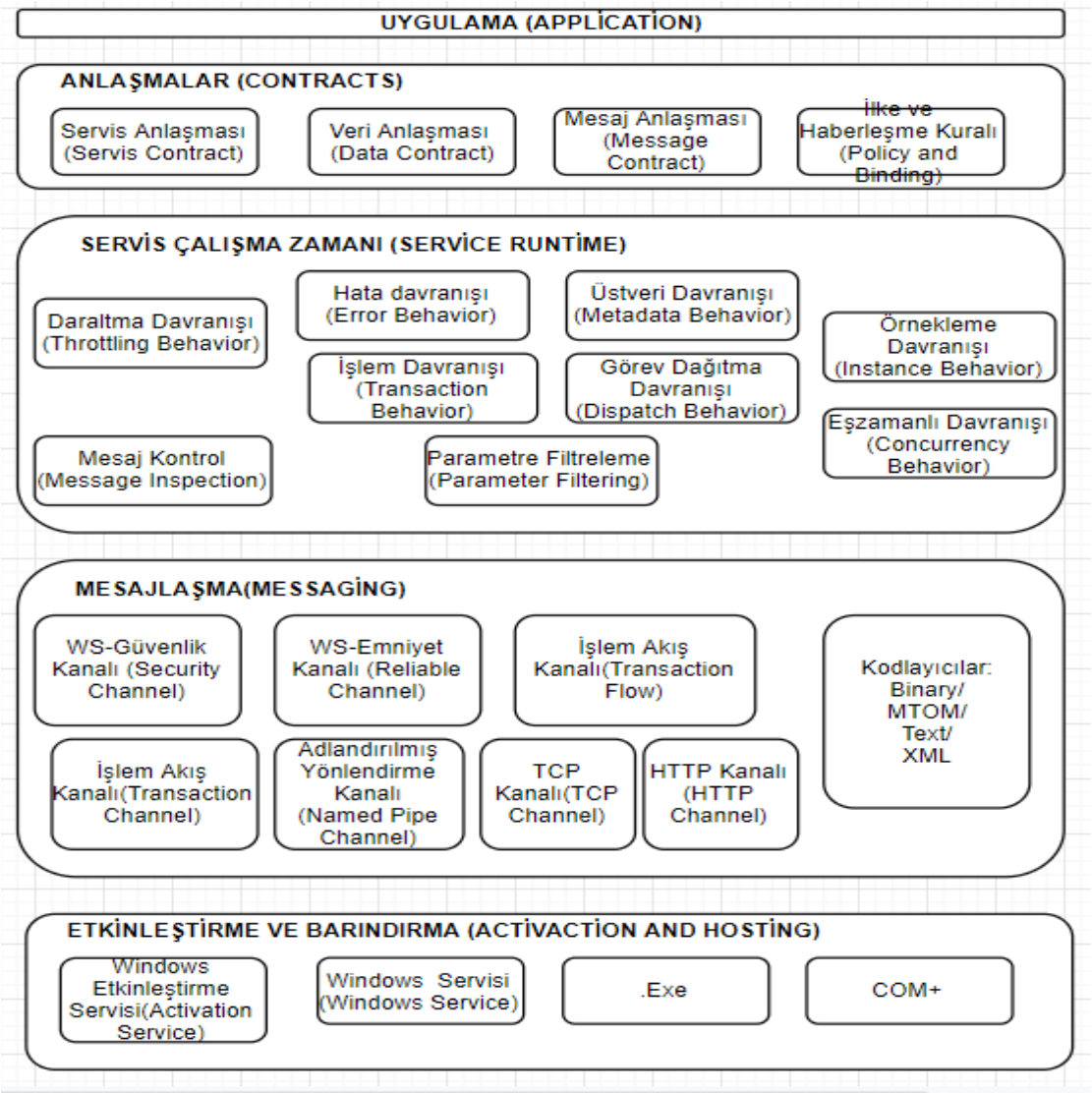
Windows Communication Foundation (WCF) hizmet odaklı uygulamalar oluşturmaya yönelik bir çerçevedir. WCF kullanarak veriler zaman uyumsuz bir ileti olarak bir hizmet uç noktasından diğerine gönderilebilir. Hizmet uç noktası, IIS tarafın-

dan barındırılan sürekli olarak kullanılabilir bir hizmetin parçası veya bir uygulamada barındırılan bir hizmet olabilir. İletileri tek karakter veya XML olarak gönderilen sözcük kadar basit ya da bir ikili veri akışı kadar karmaşık olabilir (Anonim, 2017).

3.5.1 WCF'nin özellikleri

- Hizmet yönlendirmesi,
- Birlikte çalışabilirlik,
- Birden çok ileti desenleri,
- Hizmet meta verileri,
- Veri anlaşmaları,
- Güvenlik,
- Birden çok aktarımları ve kodlamaları,
- Güvenilir ve kuyruğa alınmış iletileri,
- Dayanıklı ileti,
- İşlemler,
- AJAX ve REST desteği,
- Genişletilebilirlik (Anonim, 2017).

3.5.2. WCF mimarisi



Şekil 3.2 WCF Mimarisi.

Şekil 3.2'de WCF genel yapısı belirtilmektedir. Data Contract; Bir servisin her mesaj için oluşturabileceği veya tüketebileceği parametreleri tanımlar. Bu mesaj parametreleri XSD kullanarak tanımlanmakta ve XML formatını anlayabilen ve bu mesajları işleyebilen herhangi bir sistem ile çalışabilmektedir. Message Contract; SOAP protokolleri kullanarak belirli ileti parçalarını tanımlar ve mesaj parçaları üzerinde detaylı bir kontrol sağlar. Service Contract; contract servisin adını, namespace'ini ve diğer global niteliklerinin tanımlandığı bileşendir. Endpoint: istemciler (clients) ve host'lar arasındaki iletişimi şağlamak için kullanılan bir arayüzdür. Policy and Binding; endpoint'lerin

dış dünya ile nasıl haberleşileceğinin tanımlandığı bir takım özellikler bütünüdür. (Cesur, 2013).

Service Runtime katmanı; yalnızca herhangi bir servisin asıl operasyon sırasında oluşan davranışları içerir. Örneğin throttling control mekanizması sayesinde kaç tane mesajın işlenebileceği belirlenebilir, Error behaviour tanımlanarak bir hata durumunda nasıl bir davranış sergileneceği belirlenebilir yada Metadata behaviour tanımlanarak metadata'yı dış dünyaya kullanıma nasıl sunulacağı belirlenilebilir. (Cesur, 2013).

Messaging katmanı; verinin olası formatı ve message pattern'leri ile ilgili işlemleri gerçekleştirmektedir. Örneğin, WS Security Channel messaging katmanında güvenliğini sağlayan WS-Security tanımlarına uyan güvenlik standartlarını uygulamaktadır. WS Reliable Messaging Channel mesajın teslimat garantisini vermektedir. Encoder'lar mesajın metin, XML veya diğer desteklenen format'lardan herhangi biri olarak gönderilmesi gerektiğini belirlemek için kullanılır (Cesur, 2013).

Activation & Hosting; Bir servis kendi kendine barındırılabilir (self-hosted) yada başka bir uygulamaya bağlı olarakta barındırılabilir. Ayrıca bir servis yine bir EXE, Windows Service, COM+ yada IIS altında da çalışabilmektedir (Cesur, 2013).

3.6 C#

C# basit, modern, nesne yönelimli, tip-korumalı ve C ile C++ dillerinden türetilmiş bir dildir. C#, .NET için Microsoft tarafından geliştirilen ve programlama alanında C/C++ ve Java'nın güzel özelliklerini alıp, bu dillerin tehlikeli olabilecek özelliklerini dışarıda bırakan bir dildir. C#, C/C++ dilinden farklı olarak tamamıyla nesneye yöneliktir. "int", "double" gibi temel veri türleri dahi birer nesne olarak tanımlanmıştır. C# ile Windows için gelişmiş, güçlü, hızlı ve güvenli programlar yazılabilir. Fakat bunun için programın çalıştığı sistemde NET platformunun yüklü olması gerekir (Karabulut, 2015).

3.7 ASP.NET

Microsoft tarafından geliştirilen bir uygulamadır. Web sitesi yapma ve üzerinde gerekli olan tüm işlemleri gerçekleştirmeye olanak sağlayan bir teknolojidir. "ASP.Net nedir?" sorusuna cevap verebilmek için ASP'nin ne olduğunu, çalışma prensibini bilmek de fayda vardır. "Aktif sunucu sayfaları" anlamına gelen ASP, IIS ile çalıştırılmaktadır. ASP programlama diliyle oluşturulan web sitenizin herhangi bir işlemci tarafından görüntülenebilmesi aşamasında tarayıcınız HTML olarak yorumlayacaktır. ASP ile oluşturulan web sitelerinin diğer adı, dinamik web siteleridir. Web sitenize ziyaretçi geldiğinde, asp kodlarınızı ziyaretçi görüntüleyemez ve güvenliğiniz için oldukça güvenilir bir sistemdir. ASP kodları görüntülenemese de istemci tarayıcı tarafından yorumlanan sonuçlarla karşılaşacaktır (Varol, 2018).

3.8. JavaScript

JavaScript, eskiden Netscape'te ve şimdi de Mozilla'da çalışan Brendan Eich tarfından 1995'te geliştirilmiştir. Orjinal adı Netscape'in kurcusu Marc Andreessen tarafından seçilen Mocha olan Javascriptin adı Eylül 1995'te adı LiveScript olarak değiştirilmiş, daha sonra, Sun'dan marka lisansı alındıktan sonra, ismi JavaScript olarak tekrar değiştirilmiştir (Aslan, 2016).

Farklı bir dil kullandığınızda geliştiricilerin birbirinin işini yapması pek kolay olmayabilir. NodeJS ile bütün ürününüzü tek dil ile çıkaracağınız için geliştiriciler arası uyum, anlayış ve işbirliği en üst seviyede tutulur.

Tek bir iş parçacığı ile bloklanmadan çalışabilme imkanı sunması, onun getirdiği en büyük avantajlardan bir tanesidir çünkü ne kadar çok iş parçacığı (thread) o kadar risk ve yerine göre performans kaybına neden olabilir.

Asenkron mimari günümüz uygulamaları için oldukça uygundur. JavaScript'in olay güdümlü, asenkron yapısı ise bunu oldukça kolaylaşmıştır. Veritabanı, başka bir web servise erişim vb. buralardan cevap gelene kadar beklemek yerine yeni istekleri hazırlanabilir.

Npm Package Manager sayesinde, node ve npm yükledikten sonra verimli, stabil web servisi geliştirme ortamını bizlere sunmaktadır (Anonim, 2016b).

3.9. HTML (Hyper Text Markup Language)

Hyper Text Markup Language ifadesinden oluşturulmuş bir İngilizce kısaltmadır ve Türkçe'ye "Hiper Metin İşaretleme Dili" şeklinde çevrilebilir. Bir işaretleme dilinin temel işlevi, bir belge içindeki metnin yapısını, biçimini ve genel görünümünü tanımlayacak kuralları içermesi ve bunun uygulanmasına olanak vermesidir (Anonim, 2015b).

HTML Web sayfası hazırlama ve formatlama dilidir. Bu dil, daha çok, yazılı bir dokümanı formatlamak ve bir objeden başka bir objeye linkler sağlamak ile ilgili komutlar içerir. HTML dosyası ses, yazı, video, animasyon, script içerir. HTML, HTTP ve ilgili diğer protokolleri kullanabilmek için renkli ve görsel kullanıcı arayüzleri hazırlanmasını olanaklı kılar. Kodlar metin olarak yazılır, gönderilir ve tarayıcı tarafından yorumlanır. Sayfaya görsel öğelerin yanında tablolar ve başka sayfalara bağlar (link) yerleştirilebilir (Göker, 2009).

HTML statik bir dildir. Dinamik hale getirmek için JavaScript programlama dili kullanılır. Tarayıcıdan sunucuya veri gönderilecek ise PHP, ASP, JSP veya CGI programlama dillerinden biri kullanılabilir (Göker, 2009).

3.9.1. HTML belgesinin yapısı

Bir HTML belgesi etiket dediğimiz bileşenlerden oluşur. HTML yapısındaki temel bileşen bir elemandır. Gördüğünüz başlıklar, paragraflar, listeler birer elemandır. Etiketler kullanılarak bu elemanlar birbirinden ayrılarak mantıklı hale gelirler. HTML etiketleri yapısı ise çok basittir. < (küçüktür işareti) etiket ve > (büyüktür işareti) işaretiyle belirtilir. Örneğin, paragraf belirtmek için <p> etiketini kullanılır. Paragraf etiketini kapatmak istediğinde ise < işareti sonrasında / işareti koymak yeterlidir (Geleceği Yazanlar Ekibi, 2015).

HTML dili büyük küçük ayrımı yapan bir dil değildir. Bir etiket girilirken büyük, küçük ya da bir kısmı büyük bir kısmı küçük girebilirsiniz (XHTML için durum farklıdır. Tüm etiketlerin küçük harflerle yazılması gerekir) ve bu herhangi bir değişime sebep olmaz. Ancak; HTML sayfası içindeki tüm kodların küçük harfle yazılması, geliş-

tiriciler arasında kabul görmüş bir kuraldır. Dolayısıyla etiketleri küçük harfle yazmanız tavsiye edilir (Geleceği Yazanlar Ekibi, 2015).

Bir HTML belgesi başlıca iki kısımdan oluşur:

- Baş (HEAD) metni
- Gövde (BODY) metni
- Baş metni, sayfaya ait başlıkla ilişkili ifadeleri içerir.

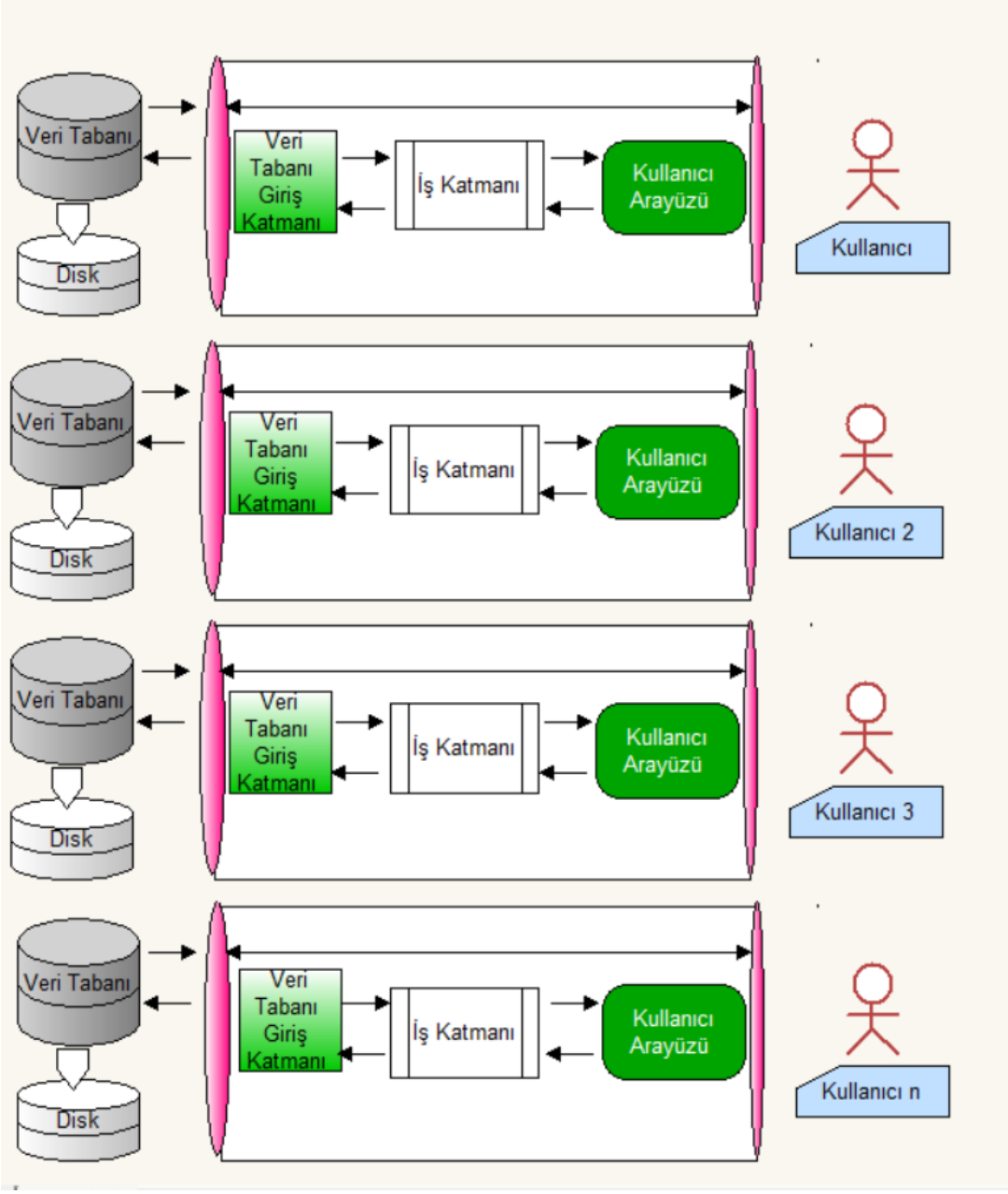
Gövde metni ise, HTML koduna ait asıl metni içerir. Bu metin,

- Paragraflar,
- Listeler ve
- Diğer elemanlardan oluşur (Geleceği Yazanlar Ekibi, 2015).

4. BULGULAR

Bu tez çalışmasında SOM ile Taşınır Kayıt Takip Uygulaması modellenmiştir. Geliştirilmesi planlanan bu yapı; servis odaklı, esnek ve çevik bir sistemdir.

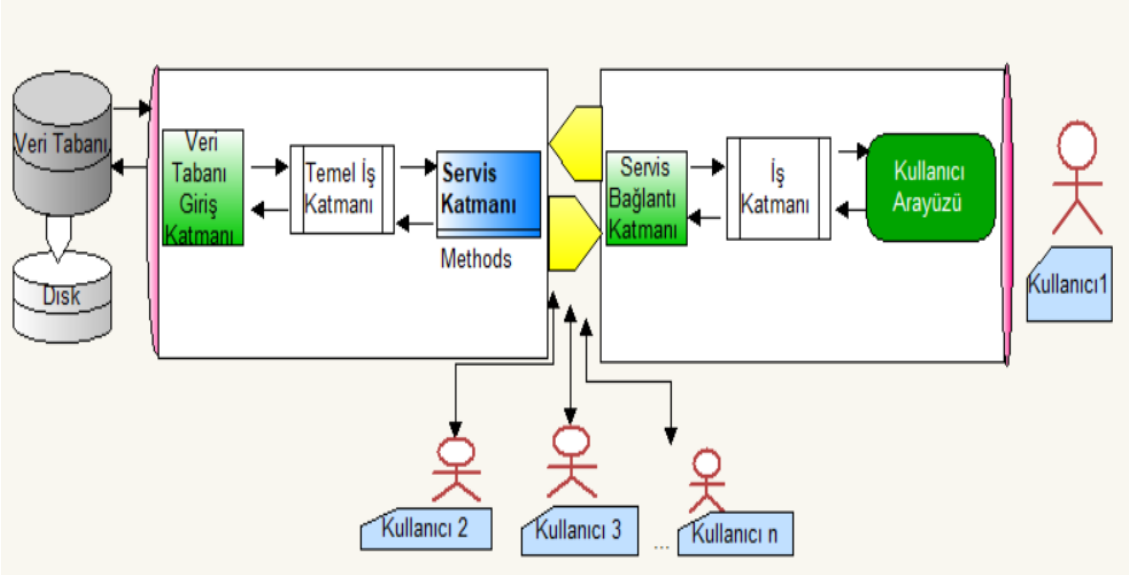
4.1. Noktadan Noktaya (Point to Point) Yapı



Şekil 4.1 SOM Öncesi Yapı Modeli.

Şekil 4.1‘de SOM tasarım desenini kullanmadan önce noktadan noktaya (point to point) mimarisinin bir örneği vardır. Bu yapıda kullanıcı, önce kullanıcı arayüzü üzerinden veri tabanı giriş katmanına (Data Access Layer) sonrada veritabanına (database) bağlanmaktadır. Geliştirilen/geliştirilecek her uygulamada veri tabanı üzerinden herhangi bir CRUD işlemi için eğer aynı veri tabanını kullanıyorlarsa (hiçbir kurumsal işletme kontrolsüz bağlantıya izin vermez) veri tabanına ayrı ayrı bağlanmak zorundadır ya da her uygulama için farklı birbirinden bağımsız ve uyumsuz veri tabanları oluşturmak zorundadır.

Uygulamalar içerisinde aynı ya da benzer işi gören işlemler(proccessler), metodlar ve fonksiyonlar her uygulama için ayrı ayrı yazılmak zorundadır.



Şekil 4.2 SOM Yapı Modeli.

Şekil 4.2‘de SOM ile her kullanıcı farklı arayüzlerden aynı veri tabanına bağlanarak birbirinden bağımsız ve uyumsuz veri silolarından kullanıcıyı kurtarmaktadır. Yazılmış olan her işlem sınırsız kullanımıyla nesnel programlamanın temel prensiplerinden olan tekrar kullanılabilirlilik prensibi sadece yazılım sınırları içerisinde değil yazılımın mimarisi içerisinde de uygulanmış olacaktır. Dış ortama servisler aracılığıyla

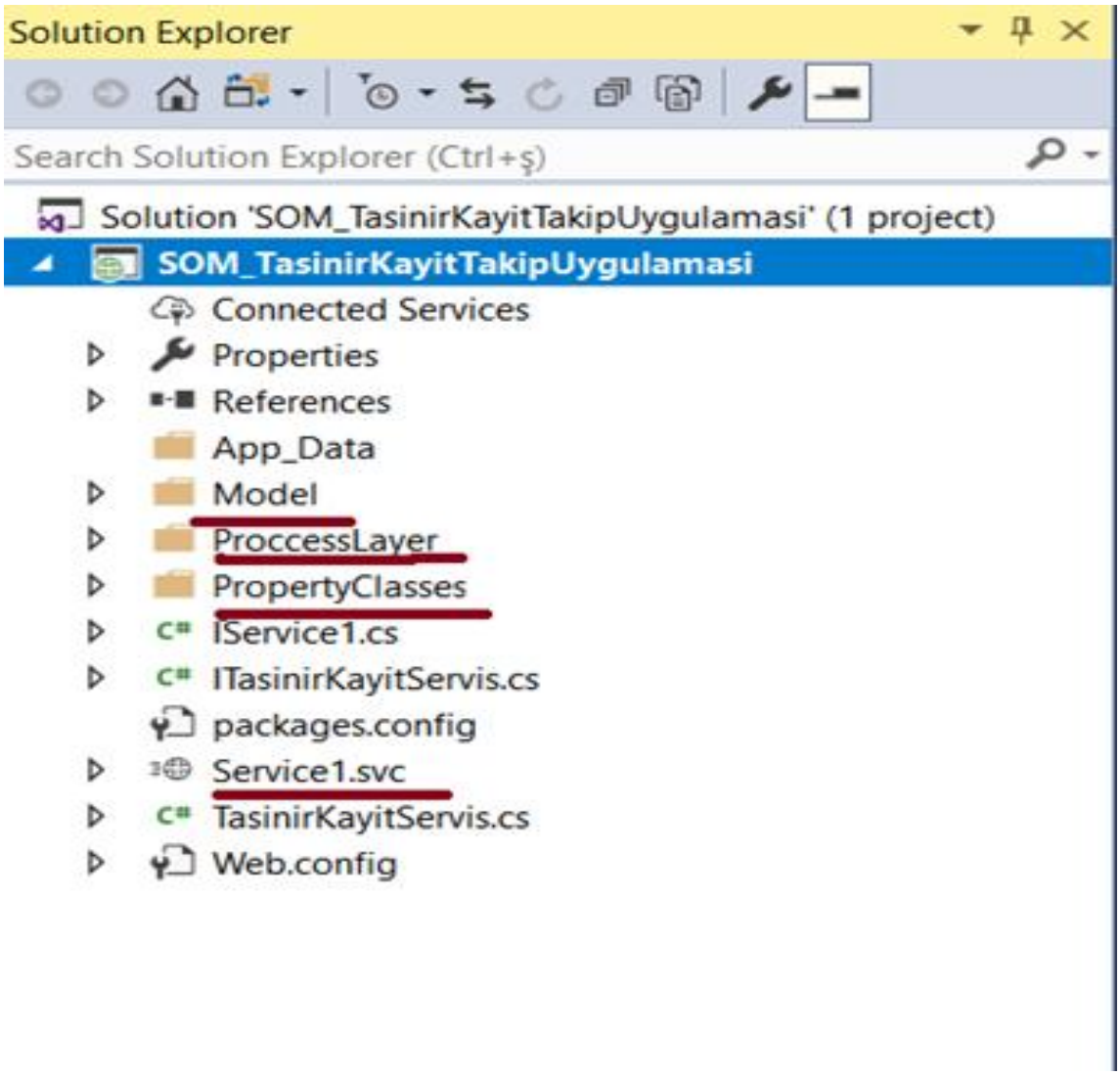
ulaşan ana uygulama servis yapısı gereği platform bağımsız, her uygulama ile temel uygulamanın entegrasyonu sağlanmış olacaktır.

Proje, Visual Studio 2017 geliştirme ortamında ASP.NET framework 4.5 versiyonu kullanılarak geliştirilmiştir. Veri Erişim Katmanı (VEK) ve Kullanıcı Arayüzü Katmanı (KAK) olmak üzere iki katmandan oluşturulmuştur. Yapılan analizler doğrultusunda veri tabanı tarafında da MSSQL üzerinden tablolar, tablolar arası ilişkiler ve ihtiyaç duyulan prosedürler yaratılmıştır. Kullanıcı arayüzü katmanında ise ekranlar geliştirilmiştir. CRUD işlemleri için veri tabanına entitiy data model üzerinden erişilmiştir.

4.2. Geliştirilen Servisler

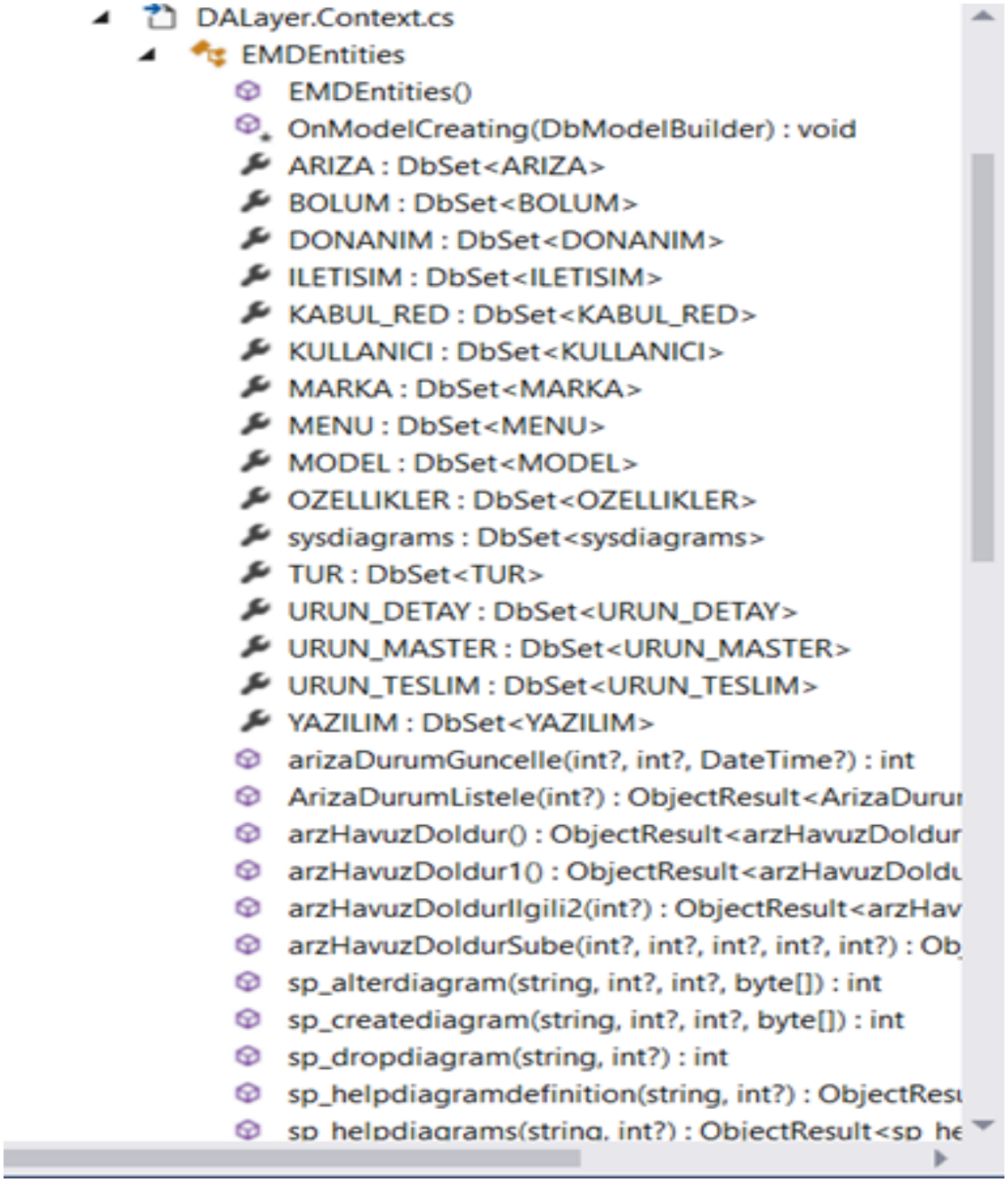
Bu çalışmada hazırlanan proje de SOM için Visual Studio geliştirme ortamında C# yazılım geliştirme dili kullanılarak ana proje ve entegre proje adı altında iki ayrı yazılım geliştirimiştir.

Projenin isterlerinin tespiti için kurum içindeki personeller ile analiz çalışması yapılalarak bu analiz çalışması için balzamique mockup kullanılmıştır. Analiz çalışması doğrultasında veri tabanı modellenmiştir. Oluşturduğumuz bu veri tabanına CRUD işlemleri için uygulama tarafında veri tabanı katmanı yaratılmıştır. Analizler doğrultusunda ihtiyaç duyulan iş parçacıkları yeniden kullanılabilirlilik ilkesi gereği maksimum seviyede küçültülerek iş katmanı oluşturulmuştur. Her iş parçacığı servis katmanı üzerinden web servisinin bir metodu olarak sunulmuştur. Kullanıcı arayüzü katmanı haricen oluşturulup servislerle etegrasyon sağlanmıştır. SOM teknolojisi ile kurulan otomasyonda her iş parçacığı bir servisin metotları olarak tasarlanmıştır ve kurumların diğer projelerinde de entegre olabilecek bir yapı tasarlanmıştır.



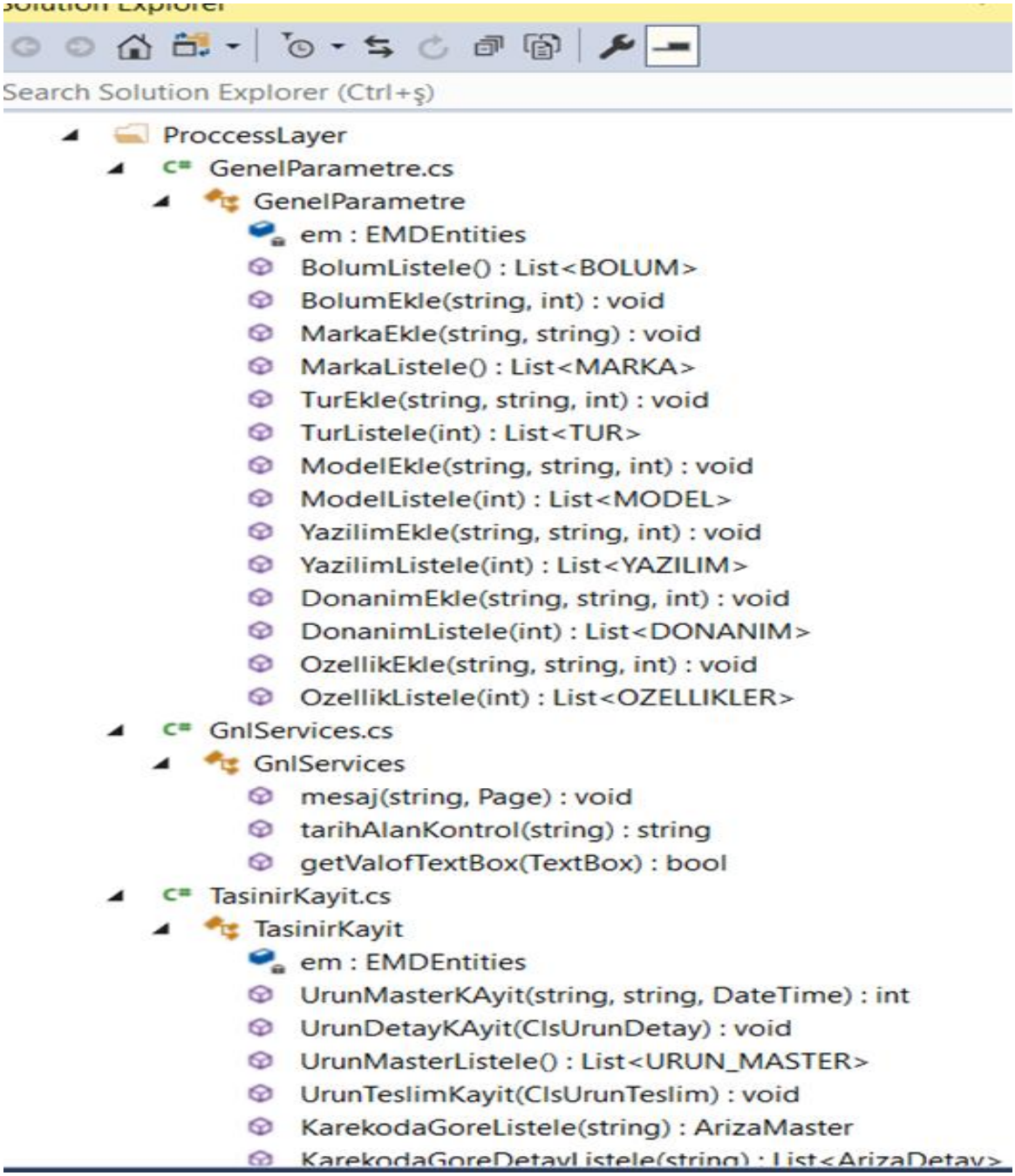
Şekil 4.3 Ana Proje Katmanları.

Şekil 4.3‘te gösterildiği gibi ana projed VEK, İşlem Katmanı (Proccess Layer) ve Servis Katmanı (Servis Layer) olmak üzere üç katmandan oluşturulmuştur. Güvenlik amaçlı Özellik Sınıf (Property Class) katmanı içerisinde her tablonun eşleştirme (maping) işlemi yapılmıştır. VEK için ADO.Net Entity Data Model kullanılarak veri erişimi yapılmıştır. İşlem (Proccess) katmanında her iş parçacığı minimum bağımlılık (loose coupling) prensibi gereği sorumlulukları minimize edilerek oluşturulmuştur.. WCF servisleri ile servis katmanı oluşturulmuştur.



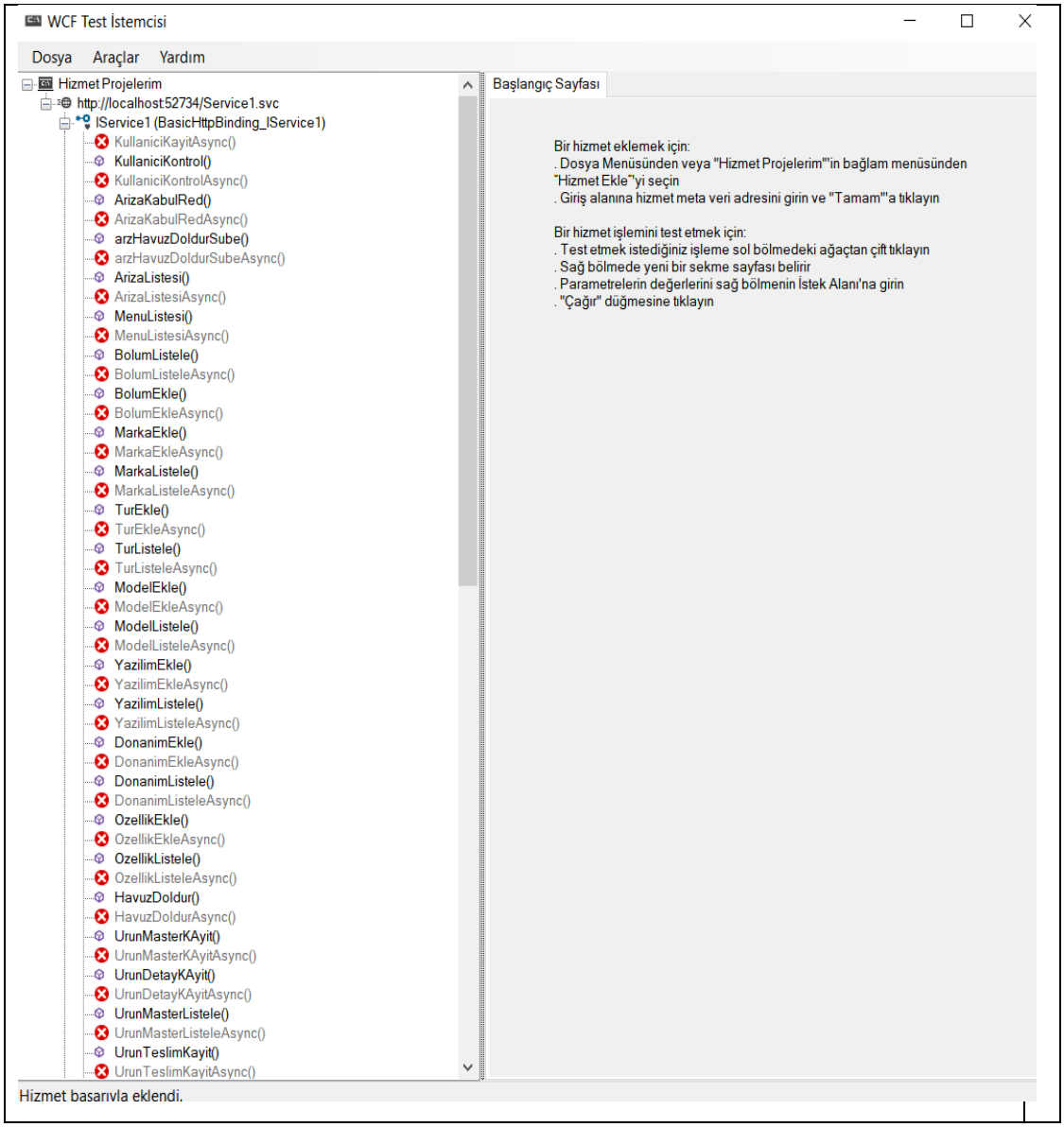
Şekil 4.4. VEK.

Şekil 4.4'de görüldüğü gibi, Veritabanı üzerinde tablolar ve prosedürler oluşturulmuştur.



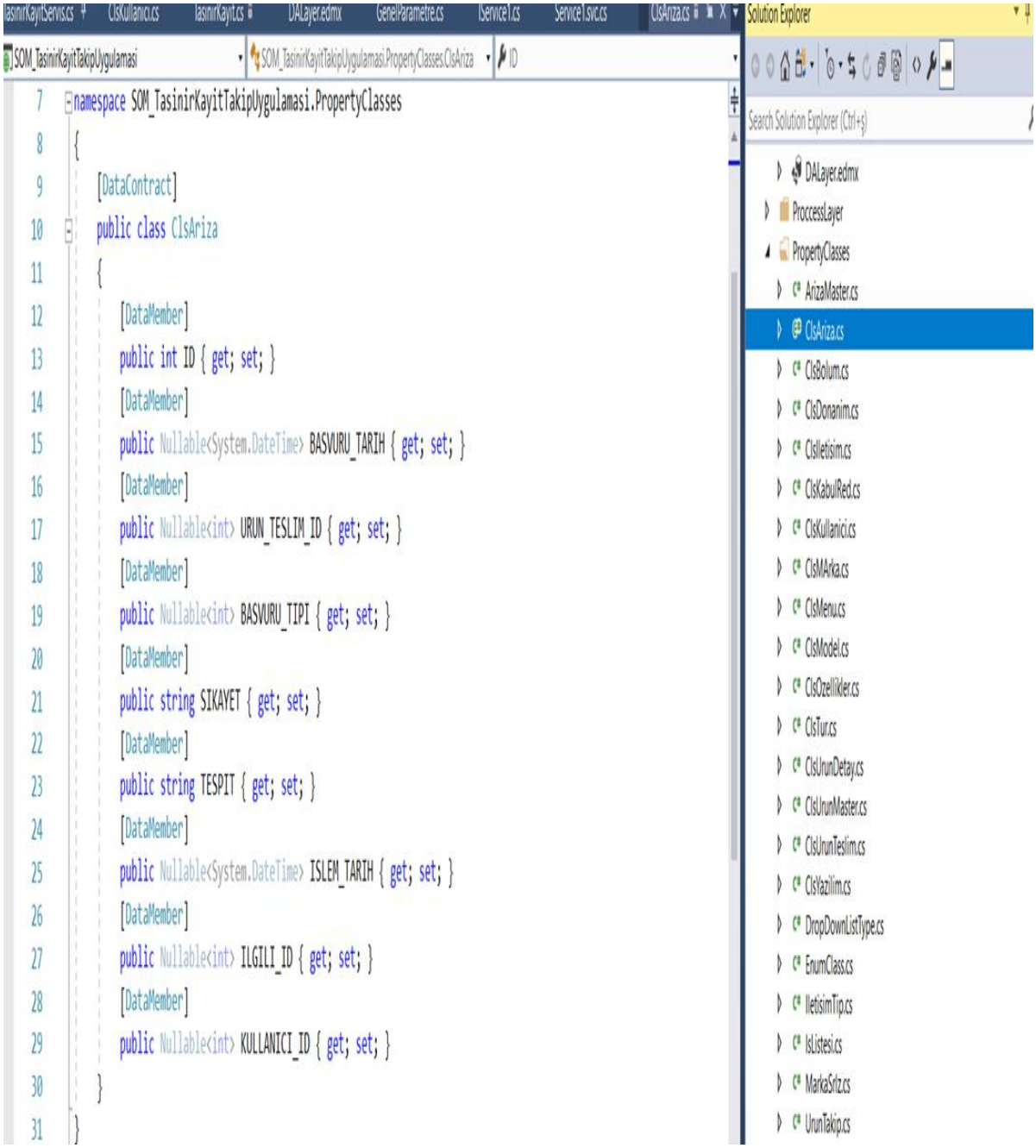
Şekil 4.5. İKK.

Şekil 4.5'de görüldüğü üzere işlem katmanında analizler doğrultusunda ihtiyaç duyulan iş parçacıkları yeniden kullanılabilirlilik ilkesi gereği maksimum seviyede küçültülerek, minimum bağımlılıkla oluşturulmuştur. İKK aynı zamanda veri tabanı üzerinde yapılacak kayıtlar ve sorgu sonucu geri dönmesi gereken alanların belirlenmesi işlemi gerçekleştirilmiştir.



Şekil 4.6. Hizmet Servis Metodları.

Şekil 4.6'da SOM'un servis katmanı görüntüsü verilmiştir. Ana uygulamada yazılan her iş parçası servis katmanı üzerinden servisin bir metodu olarak yayınlamıştır. Böylelikle platform bağımsız ana uygulamanın her iş parçacığına erişebilir olması niteliği kazandırmıştır. Aynı zamanda bu iş parçacıkları bir araya gelerek işletmenin isteği doğrultusunda yeniden kurallar ve veri tabanı katmanı yaratılmadan yeni ekranların oluşturulabilme olanağı sağlandı.



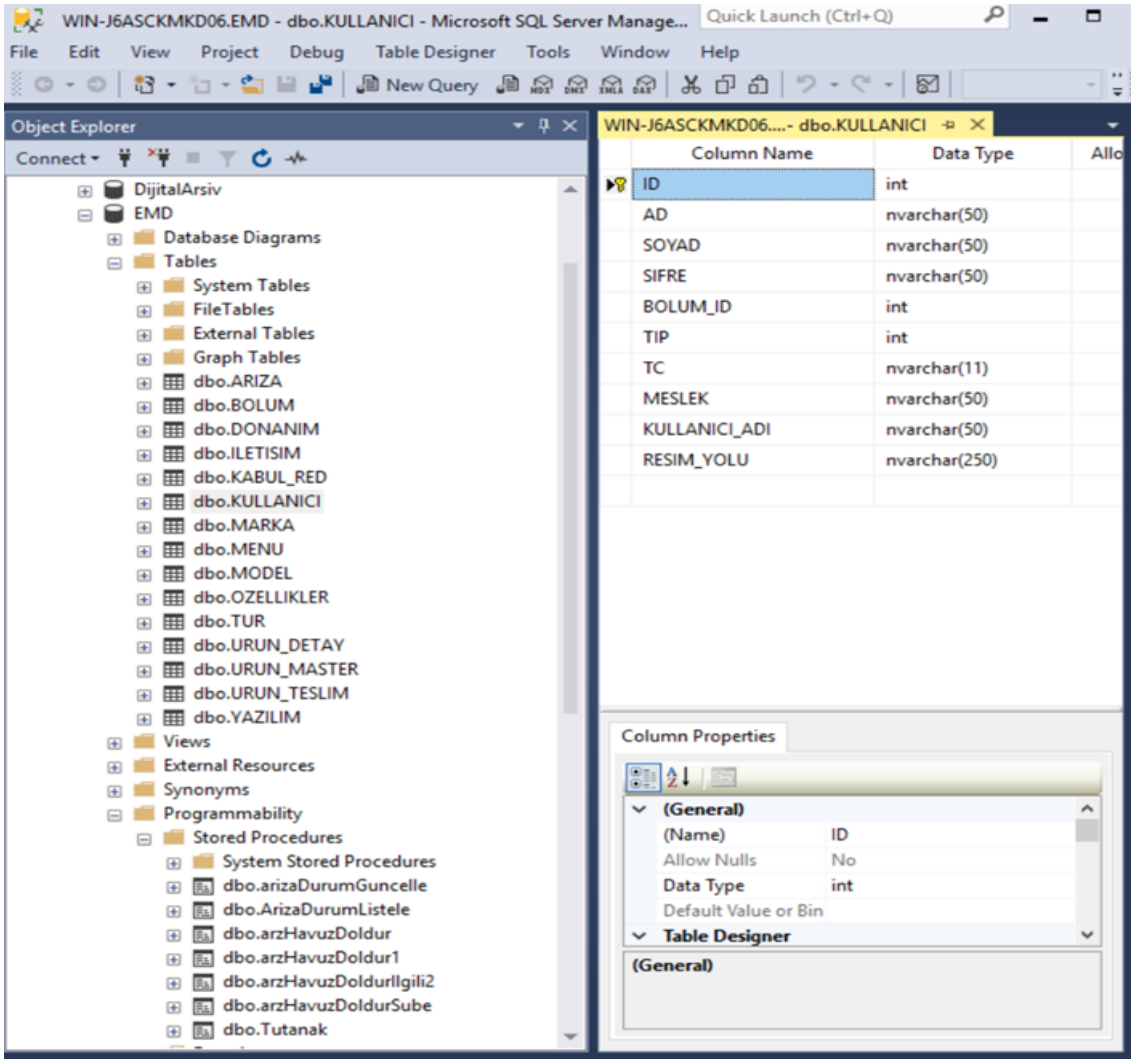
Şekil 4.7 Özellik Sınıfı.

Şekil 4.7'de görüldüğü gibi Özellik Sınıfı katmanı içerisinde en yüksek seviyede kontrol ve güvenlik amaçlanmış ve her tablonun eşleştirme (maping) işlemi gerçekleştirilmiştir.

4.3. Proje Veri Tabanı

Analiz çalışması doğrultusunda tabloların alanları, alanların tipleri, alanların boyutları, tablolar arası ilişkiler kurularak veri tabanı modellenmiştir.

Şekil 4.8'de veri tabanı tarafında da yapılan analizler doğrultusunda MSSQL üzerinden tablolar, tablolar arası ilişkiler ve ihtiyaç duyulan prosedürler yaratılmıştır.

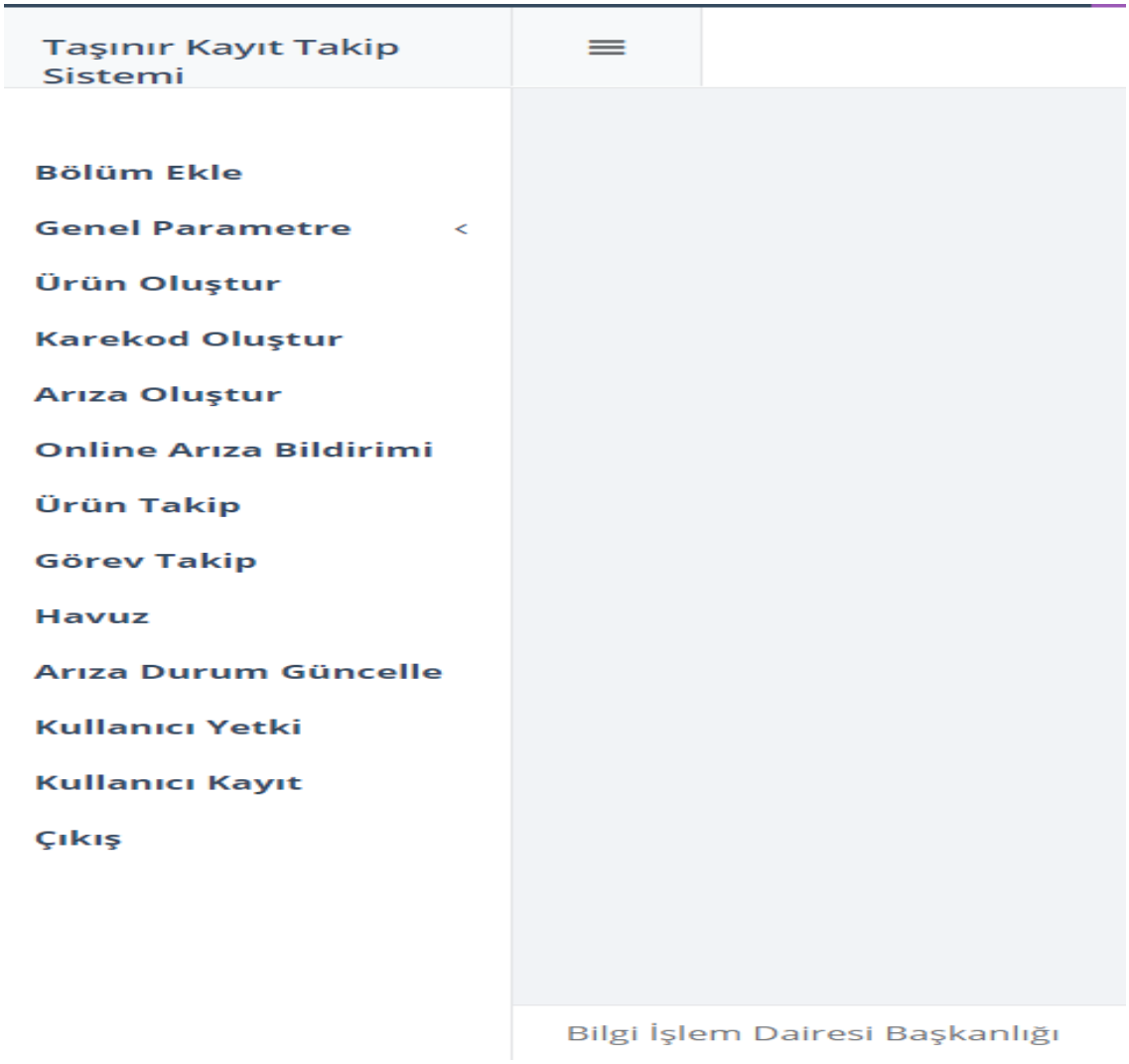


Şekil 4.8 Hazırlanan Veri Tabanı.

4.4. Servislere Entegre Edilen Kullanıcı Arayüzü Birimi

Proje, Visual Studio 2017 geliştirme ortamında ASP.NET framework 4.5 versiyonu kullanılarak geliştirilmiştir. Proje VAK ve KAK olmak üzere iki katmandan oluşturulmuştur.

Ana uygulamada geliştirilen servisler TKTUServis adıyla web referans olarak entegre projeye referans edilmiştir. Yapılan analizler doğrultusunda kullanıcı arayüzü katmanının ekranları oluşturulmuştur. Uygulamadaki CRUD işlemleri veri tabanına bağlanmadan ana uygulamadaki servisler çağırılarak gerçekleştirilmiştir.

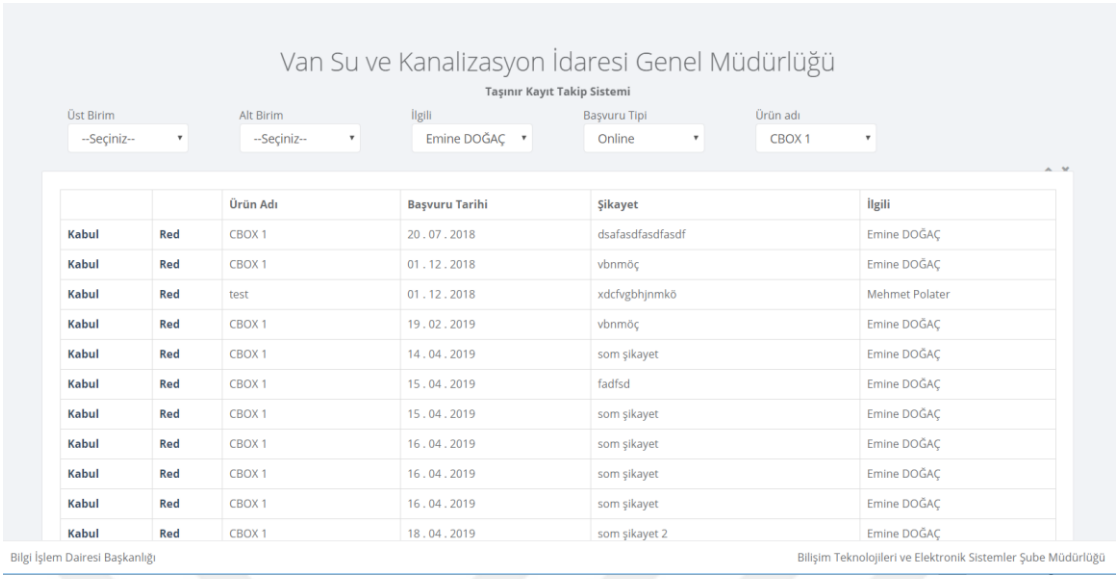


Şekil 4.9 Anasayfa Ekranı Arayüzü.

Şekil 4.9'da anasayfada bulunan parametreler yer almaktadır. Bu alanda bölüm eklenebilmekte, genel parametreler eklenebilmekte, ürün oluşturulabilmekte, karekod ile ürün teslimi yapılabilmekte, arıza talebi oluşturulabilmekte, online arıza bildirimi yapılabilmekte, ürün ve görev takibi yapılabilmekte, arıza durum güncellenebilmekte, kullanıcı kaydı ve kullanıcıya yetki verilebilmektedir.

Şekil 4.10 Ürün Oluştur Ekranı Arayüzü.

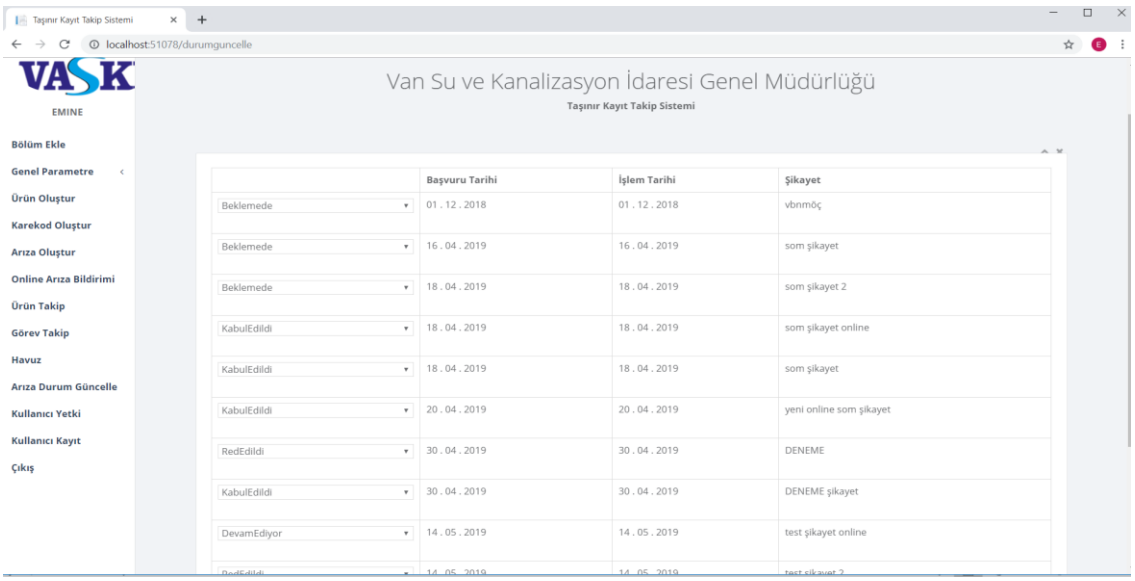
Şekil 4.10'da kuruma alınan taşınırların kaydının tutulduğu, son kullanıcıya kadar kontrolünün sağlandığı, taşınırlarda oluşan sorunların kurum içi birimlere iletildiği, bu sorunların giderilmesi için iş emirlerinin çıkarıldığı ve iş emirlerimin takibinin yapıldığı sistemdir. Karekod yazdırma ve okuma işlevi üzerine kayıtlar tutulmaktadır.



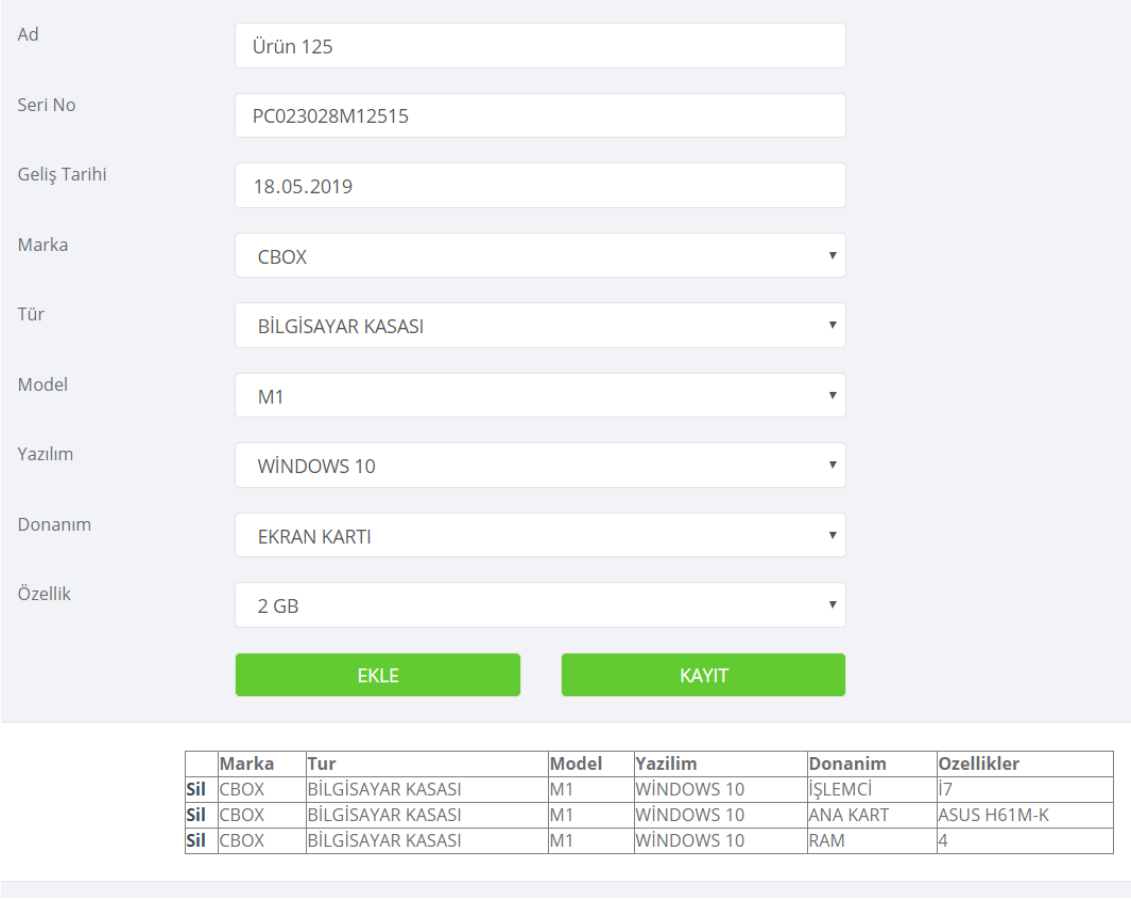
Şekil 4.11 Havuz Ekranı Arayüzü.

Şekil 4.11'de kurum amirlerinin ekranlarına kurum içinde yapılan tüm talepler kabul edilerek seçilen ilgili kişilere iş emri olarak atanır veya red edilerek havuzdan düşürülür.

Şekil 4.12'de kurum içerisindeki çalışanlara iş emir olarak atanan görevlerin arıza durumlarının "Kabul edildi, Red Edildi, Bitti, Başlanmadı, Devam Ediyor, Beklemede" seçenekleriyle güncellendiği ekrandır.



Şekil 4.12 Arıza Durum Güncelle Ekranı Arayüzü.



Şekil 4.13 Taşınır İşlem Fişi Kaydı Ekranı Arayüzü.

Şekil 4.13'te görüldüğü gibi kurum içerisinde kullanılacak her ürünün kaydedildiği ekrandır. Temelde bu ürün/ürünler kendi ekranımıza kaydedildiği gibi destek hizmetleri biriminden kurumda alımı yapılan ürünlerin tarafımıza servisler aracılığıyla kayıt işlemleri yapılabilmektedir.

Çizelge 4.1'de geleneksel yaklaşım ile SOM arasındaki farklılıklar karşılaştırmalı olarak verilmiştir.

Çizelge 4.1 Yöntemlerin Karşılaştırılması

Ölçütler	**Geleneksel Mimari**	**SOM**
İş Parçacıklarının Özelliği	Aynı işlevi gören iş parçacıklarının farklı projelerde tekrar tekrar yazıldığı görülmüştür	İş Parçacıkları servisin metodları olarak yazıldığından, aynı iş parçacığını yeniden geliştirme ihtiyacı duymadan farklı projelerde de kullanılabilir olduğu görülmüştür
Kod Maliyeti	Tekrar eden kod bloklarından dolayı kod geliştirme maliyetinin artabileceği görülmüştür.	Tekrar kullanılabilir servis metodlarından kaynaklı kod geliştirme maliyetinin düşebileceği görülmüştür.
Entegrasyon Özelliği	Noktadan Noktaya mimari ile geliştirilen uygulamaların yapısı gereği başka projelerle entegrasyon yapılamadığı görülmüştür.	SOM ile geliştirilen uygulamalar yapısı gereği son kullanıcının her isteğine yanıtını SOAP, json vb notasyon ile verdiği için her projeyle entegrasyon yapılabildiği görülmüştür.
Platform Bağımsızlığı	Platform bağımlı olduğu görülmüştür.	SOM ile geliştirilen uygulamalar yapısı gereği son kullanıcının her isteğine, yanıtını SOAP, json vb notasyon ile verdiği için her platforma entegre edilerek platform bağımsız geliştirme olanağı sağladığı görülmüştür.

Çizelge 4.1 Yöntemlerin Karşılaştırılması (devam)

Ölçütler	Geleneksel Mimari	SOM
Güvenlik	Her uygulama ve veri tabanına uygulanmak zorunda olduğu ve bunun güvenlik maliyetini artırdığı görülmüştür.	Sadece servisler ve bir veri tabanına uygulandığı ve güvenlik maliyetinin düştüğü görülmüştür.
Değişim ve yeniden geliştirme	İş gücü ve zaman maliyetinin arttığı görülmüştür.	Servislerin metodları bir araya getirilerek iş gücü ve zaman maliyetleri indirgenerek gerçekleştirildiği görülmüştür.
Veri uyumluluğu	Birbirinden bağımsız ve uyumsuz veri siloları oluşturulabildiği görülmüştür.	Tek veri tabanı kullanılarak anlamlı ve tutarlı veri kaynağı oluşturulabildiği görülmüştür.
Uygulamanın iş süreçleri üzerine hâkimiyet	Tek ve uzun geliştirme sürecine sahip olduğu görülmüştür.	Artırımlı kod geliştirilerek uygulama üzerinde daha fazla hâkimiyet sağladığı görülmüştür.
Anlaşılır sade nesnel yapı	Uygulama bloklarından oluştuğu görülmüştür.	Birlikte çalışan bağımsız bileşenler, nesneler ve servisler oluştuğu görülmüştür.
Bilişim teknolojileri personelinin uygulama üzerindeki hâkimiyeti	Kapalı kutu yapısından dolayı kurum içerisindeki tüm bilişim teknolojileri çalışanlarının hâkimiyeti dışında olduğu görülmüştür.	SOM yapısından kaynaklı servisler lego misali bilişim teknolojileri çalışanları tarafından biraraya getirilerek yeni ekranlar, yeni uygulamalar geliştirildiği görülmüştür.

5. TARTIŞMA VE SONUÇ

Çalışmada birden fazla proje hazırlanarak GYM ve SOM karşılaştırması yapılmıştır. GYM ile oluşturduğumuz projede büyük dezavantajlar tespit edilmiştir. GYM ile yazılan her proje için aynı işlevi gören iş parçacıklarının farklı projelerde tekrar tekrar yazıldığı görülmüş, bundan dolayı da tekrar tekrar yazılan kodlar ile birlikte daha önce yazılmış kodlara bakım destek maliyetlerinin artışı tespit edilmiştir. Bilindiği üzere nesnel programlamanın temel özelliklerinden biri de tekrar kullanılabilirlilik ilkesidir. Kang ve ark (2010) çalışmalarında SOM kullanılan bir uygulamada yeniden kullanılabilirlik üst seviyede olduğunu, oluşturulan her servisin farklı projelerde dahi olsa yeniden kullanılabildiği fikrini ortaya çıkarmışlardır. Tekrar kullanılabilirlilik ilkesinin sadece dahili ortamında değil de aynı zamanda genel, farklı platformlarda yada entegre edilebilecek farklı projelerde de uygulanılıyor olabilmesi kod yazma ve bakım için maliyetleri ciddi oranda minimize etmektedir.

GYM ile geliştirilen her uygulama için veritabanı oluşturmak zorunda kalınmış olup birbirinden bağımsız ve uyumsuz veri tabanları oluştuğu tespit edilmiştir. Tablolar, tabloarın alanları, prosedürler, trigger veya indexlerin tekrar tekrar yazılma ihtiyacı oluşmuştur. Ve hatta bazı projelerde aynı amaçla kullanılan tabloların alanlarının biribirilerinden farklı olduğu görülmüştür.

SOM ile oluşturduğumuz yapıda aynı veritabanı birden fazla proje için kontrollü kullanılabilir özelliğine sahip olması ile her proje için tablolar, tabloların alanları, prosedürler, trigger veya indexlerin tekrar tekrar yazılma gereksinimi ortadan kaldırılmış olup uyumsuz veri siloları sorunu giderilmiştir. Noktadan noktaya olan GYM ile geliştirilen bir yazılımda, sonradan ihtiyaç duyulan yeni ekranların geliştirilmesi için yazımların her katmanında (VEK, İKK, KAK) yeniden kod geliştirmeye gereksinimi duyulmuştur. SOM ile geliştirilen projede serivislerdeki metodlar bir araya getirilerek lego misali kurumların ihtiyaç duydukları yeni ekranları sadece KAK'da yapacakları geliştirme ile oluşturabildikleri görülmüştür. GYM ile geliştirilen proje veritabanından ayrıştırılmadan KAK üzerinden erişiliyor olabilmesinin güvenlik zaafiyetine yol açabiliceği tespit edilmiştir. SOM ile geliştirilen projede veri tabanı, kullanıcı arayüzlerinden ayrıştırıla-

rak özellikle CRUD işlemleri esnasında maksimum seviyede veri güvenliği sağlama olanağı vermektedir.

Projemiz SOM ile işlev ve süreç odaklı olmuştur. Performans ve değişim için tasarlanmıştır. Platform kararlaştırılmamış ve platform bağımsız entegrasyon imkanı sağlamıştır. Uygulama blokları değil birlikte çalışan bağımsız bileşenler, nesneler ve servisler oluşturulmuştur. Önemli bir avantajı ise tek ve uzun geliştirme sürecinin yerini arttırımlı geliştirme ve yaşam döngüsünün almasıdır.

Ayrıca kurumsal düzeyde, SOM projelerinde müşterilerin önceliği maliyetleri düşürmektir. . Çopur (2011) çalışmasında, SOM kullanılarak işletmeler; geliştirme maliyetlerinde önemli tasarruflar elde edebilir, yeni uygulamaların geliştirilmesinde mevcut hizmetlerin yeniden kullanıldığı ve yeniden yapılandırarak değişen iş koşullarına hızla adapte olabildiği fikrini ortaya çıkarmıştır. Bu doğrultuda SOM kullanılarak BT departmanına daha düşük maliyetlerle daha büyük yatırımlar yapılabilir hale gelir. Hazır servislerin kullanılması ile BT departmanının kurum içi iş süreçlerine katılımında artış sağlanır. İş süreçlerinde esneklik gözlemlenir. Servislerden oluşan esnek yapıyla kurumun ihtiyaç duyduğu/duyabileceği ekranlar geliştirilebilir hale gelir. Güvenlik, yazılım ve veritabanı lisans maliyetleri minimize edilir. Birden fazla veritabanı kullanılmayarak birbirinden bağımsız ve tutarsız veri siloları yerine oluşturulan bir veritabanı servisinin üzerinde izin verilen herkesin erişimi sayesinde tutarlı veriler elde edilebilir. SOM kullanarak katma değer getirilerine sahip olunur. Bunlar özellikle servisler ve aktörlerin iyileştirilmiş iş süreçleri ve iş birliğini sağlaması, modüler parçalarla gevşek bağlı kolay entegrasyon imkanı sağlaması, varolandan yararlanma ve tasarruf ile yeniden kullanım imkanı, değişen iş ihtiyaçlarına kolay yanıt verilerek iş esnekliğinin sağlaması, arttırımlı geliştirme ve hızlı geri dönüş ile azaltılmış risk sağlamasıdır.

KAYNAKLAR

A. Arsanjani., 2002. Developing and integrating enterprise components and services: ***Introduction. Communications of the ACM***, **45**(10): 30-34.

Altınbaş S., 2008. ***Servis Odaklı Mimari Yaklaşımı Kullanılarak Ontoloji Temelli İlişkisel Veritabanlarına Erişim*** (yüksek lisans tezi) Ege Üniversitesi, Fen Bilimler Enstitüsü,121.

Anonim. 2014a. Web Yazılım Dilleri ve Önemi, ***https://www.nsreklam.com/web-yazilimi-dilleri-ve-onemi***. Erişim tarihi:15.05.2019.

Anonim. 2014b. Yazılım Nedir? Niye Önemlidir? Yazılım Çeşitleri Nelerdir?. ***https://www.brandingturkiye.com/yazilim-nedir-niye-onemlidir-yazilim-cesitleri-nelerdir***. Erişim tarihi:15.05.2019.

Anonim. 2015a. Yazılım Çeşitleri Nelerdir?. ***https://destek.kmk.net.tr/yazilim-cesitleri-nelerdir.html***. Erişim tarihi:15.05.2019.

Anonim, 2015b. HTML'e Giriş. ***https://geleceqiyazanlar.turkcell.com.tr/konu/web-programlama/egitim/101-html/htmle-giris.*** Erişim tarihi:16.07.2018.

Anonim. 2016a. JavaScript Nedir?. ***https://geleceqiyazanlar.turkcell.com.tr/konu/web-programlama/egitim/301-javascript/javascript-nedir*** . Erişim tarihi:16.07.2018.

Anonim. 2016b. JavaScript ve Avantajları?. ***https://www.atasevermedia.com/javascript-ve-avantajlari.html***. Erişim tarihi:01.05.2019.

Anonim, 2017. Windows Communication Foundation nedir? ***https://docs.microsoft.com/tr-tr/dotnet/framework/wcf/whats-wcf/***. Erişim tarihi: 15.07.2018.

Anonim. 2018. Yazılım Nedir? Niye Önemlidir? Yazılım Çeşitleri Nelerdir?. ***https://www.brandingturkiye.com/yazilim-nedir-niye-onemlidir-yazilim-cesitleri-nelerdir***. Erişim tarihi:15.05.2019.

Oğuzhan, A., 2016. ES5, ES6, ES2016, ES.Next: JavaScript sürümleri nasıl ilerliyor?. ***http://oguzhan.in/es5-es6-es2016-es-next-javascript-surumleri-nasil-ilerliyor***. Erişim tarihi:01.05.2019.

Avcı, K., 2016, SOA mimarisi nedir? ***https://www.kadir.xyz/yazi/60/soa-mimarisi-nedir,*** Erişim tarihi: 13.07.2018.

Beklen A., 2009. ***Kurumsal Servis Odaklı Mimari Kavramı, Teknolojisi Ve Tasarımı*** (yüksek lisans tezi). Maltepe Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul.

C. Bussler. 2013. ***B2B integration: Concepts and Architecture***. Springer Science & Business Media.

Cesur, Kazım, 2013. WCF Nedir?, ***http://www.kazimcesur.com/wcf/*** Erişim tarihi: 15.04.2019.

Channabasavaiah K., Holley K., Edward M. Tuggle, Jr. 2004. Migrating to a service-oriented architecture, ***IBM Journal of Research Development***, 3-22.

Çopur, D., 2011, ***An Application Of Service Oriented Architecture (Soa) Approach*** (yüksek lisans tezi) .Başkent Üniversitesi, Fen Bilimler Enstitüsü, Ankara.

Dirik, S. 2012 Yazılım Mimarisi (Software Arhitecture). ***http://serhatdirik.blogspot.com/2011/04/yazlm-mimarisi-software arhitecture.html.*** Erişim tarihi: 05.05.2019,

Dongsu, K. and Doo-Kweon, B., 2010, Bridging software product lines and service-oriented architectures for service identification using BPM and FM, ***9th IEEE/ACIS International Conference on Computer and Information Science.*** Yamagata, Japan.

Erl T., 2005. ***Service-Oriented Architecture: Concepts, Technology, and Design***, Prentice Hall PTR, ISBN: 0-13-185858-0, New Jersey.

Francis T., Herness E., High Jr R., Knutson J., Rochat K., Vignola C. 2004. ***Professional IBM WebSphere 5.0 Application Server***.

Geleceği Yazanlar Ekibi, 2015. Bir HTML Belgesinin Yapısı. ***https://gelecegiyazanlar.turkcell.com.tr/konu/web-programlama/egitim/101-html/bir-html-belgesinin-yapisi***. Erişim tarihi:01.05.2019.

Göker Ö., 2009 ***İnternet Üzerinden Havuz Denetimi*** (yüksek lisans tezi) Muğla Üniversitesi, Fen Bilimleri Enstitüsü,98, Muğla.

Gündüz, D., (b), 2002. ***Veritabanına Giriş, Linux Kullanıcıları Derneği Seminerleri***, İstanbul.

H. Kreger. 2003. Fulfilling the Web services promise**.** ***Communications of the ACM***, **46**(6), 29-ff.

Herand, D., 2013, ***Servis Odaklı Mimarinin (Som) İş Süreçleri Üzerine Uygulanabilmesi İçin Bir Metolojinin Geliştirilmesi*** (yüksek lisans tezi). Sakarya Üniversitesi, Fen Bilimler Enstitüsü, Sakarya.

Hudayioglu, F. 2006, SOA ve ESB nedir? ***http://turk-internet.net/portal/yaziqoster.php?yaziid=14659,*** Erişim tarihi: 14.07.2018.

Jeng,JJ. and An, L., 2007, System Dynamics Modeling for SOA Project Management, ***IEEE International Conference on Service-Oriented Computing and Applications (SOCA'07)***. California, USA.

Kaya M. 2009 ***Service Oriented Architecture*** (Yüksek lisans tezi) Doğuş Üniversitesi, Fen Bilimler Enstitüsü İstanbul

Karabulut, C., 2015. ***Faunistik Database Programının Geliştirilmesi***. (yüksek lisans tezi) Hitit Üniversitesi, Fen Bilimleri Enstitüsü), 53.

Lee, Y., Ma, C., Chou, S., 2005. ”A Service-oriented architecture for design and development of middleware”, ***Proceedings of 12th Asia-Pasific Software Engineering Conference(ASPEC’05)***.

M. P. Papazoglou., W.-J. Heuvel. 2007. Service oriented architectures: approaches, technologies and research issues. ***The VLDB Journal—The International Journal on Very Large Data Bases,*** **16**(3), 389-415.

O'Brien, L., 2009, A framework for scope, cost and effort estimation for service oriented architecture (SOA) projects, ***Australian Software Engineering Conference***. Sydney, Australia.

Obalı M., 2006. ***Oracle 10 G***. İstanbul: Pusula Yayıncılık.

Srinivasan, L. Treadwell, J., 2005. ***An Overview of Service-oriented Architecture, Web Services and Grid Computing,*** HP Software Global Business Unit

Şenyurt, S. B. 2015. SOA Nedir? ***https://www.buraksenyurt.com/post/SOA-Nedir/*** Erişim tarihi: 14.07.2018.

Tekinerdoğan, B., 2014. ***Software Architecture in Volume I. Computer Science Handbook, 2nd Edition***, , CRC Press-Taylor and Francis Group.

Üstündağ S., 2006 ***Servis Odaklı Mimari*** (Yüksek lisans tezi) Ege Üniversitesi Fen Bilimleri Enstitüsü,133, İzmir.

Varol, Y., 2018. ASP.net nedir?, ***https://www.armadigital.net/web-tasarim-blog/asp-net-nedir/.*** Erişim tarihi: 15.07.2018.

Yeren, A., 2016, Yazılım nedir? ***https://www.mediaclick.com.tr/blog/yazilim-nedir,*** Erişim tarihi: 15.05.2019,

ÖZ GEÇMİŞ

Emine DOĞAÇ, 1988 yılında Van ilinde doğdu. İlk, orta ve lise eğitimini Van'da tamamladı. 2012 yılında Doğu Akdeniz Üniversitesi Bilgisayar Mühendisliği Bölümü'nü bitirdi. 2014 yılında İş Güvenliği Uzmanı ünvanını aldı. 2015 yılında Van Yüzüncü Yıl Üniversitesinde İşletme Anabilim Dalı'nda yüksek lisansını tamamladı. 2013 yılından itibaren Van Su ve Kanalizasyon İdaresi Genel Müdürlüğü'nde Bilgisayar Mühendisi kadrosunda aktif görev yapmakta olup son üç yıldır Bilişim Teknolojleri ve Elektronik Sistemler Şube Müdürlüğü ünvanında görev yapmaktadır.

T.C
VAN YÜZÜNCÜ YIL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
LİSANSÜSTÜ TEZ ORİJİNALLİK RAPORU

Tarih: 24/07/2019

Tez Başlığı / Konusu: Servis Odaklı Mimari İle Taşınır Kayıt Takip Uygulamasının Geliştirilmesi

Yukarıda başlığı/konusu belirlenen tez çalışmamın Kapak sayfası, Giriş, Ana bölümler ve Sonuç bölümlerinden oluşan toplam 23 sayfalık kısmına ilişkin, 24/07/2019 tarihinde tez danışmanım tarafından Turnitin intihal tespit programından aşağıda belirtilen filtreleme uygulanarak alınmış olan orijinallik raporuna göre, tezimin benzerlik oranı % 1 (bir) dir.

Uygulanan filtreler aşağıda verilmiştir:

- Kabul ve onay sayfası hariç,
- Teşekkür hariç,
- İçindekiler hariç,
- Simge ve kısaltmalar hariç,
- Gereç ve yöntemler hariç,
- Kaynakça hariç,
- Alıntılar hariç,
- Tezden çıkan yayınlar hariç,
- 7 kelimeden daha az örtüşme içeren metin kısımları hariç (Limit inatch size to 7 words)

Van Yüzüncü Yıl Üniversitesi Lisansüstü Tez Orijinallik Raporu Alınması ve Kullanılmasına İlişkin Yönergeyi inceledim ve bu yönergede belirtilen azami benzerlik oranlarına göre tez çalışmamın herhangi bir intihal içermediğini; aksinin tespit edileceği muhtemel durumda doğabilecek her türlü hukuki sorumluluğu kabul ettiğimi ve yukarıda vermiş olduğum bilgilerin doğru olduğunu beyan ederim.

Gereğini bilgilerinize arz ederim.

24/07/2019

Adı Soyadı: Emine DOĞAÇ

Öğrenci No: 159101132

Anabilim Dalı: Elektrik Elektronik Mühendisliği ABD

Programı: .....

Statüsü: Y. Lisans ☒ Doktora ☐

DANIŞMAN ONAYI
UYGUNDUR

Doç. Dr. Rıdvan SARAÇOĞLU

ENSTİTÜ ONAYI
UYGUNDUR

(Unvan, Ad Soyad, İmza)
Enstitü Müdürü