T.C.

TRAKYA ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

MICROSOFT DYNAMICS AX ARACINI KULLANARAK İŞLETME İHTİYAÇLARINA YÖNELİK UYGULAMALARIN GELİŞTİRİLMESİ

ECEHAN STRİKO

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

Tez Danışmanı: Dr. Öğr. Üyesi Özlem UÇAR

EDİRNE-2021

Ecehan STRİKO'nun hazırladığı "**MICROSOFT DYNAMICS AX ARACINI KULLANARAK İŞLETME İHTİYAÇLARINA YÖNELİK UYGULAMALARIN GELİŞTİRİLMESİ**" başlıklı bu tez, tarafımızca okunmuş, kapsam ve niteliği açısından **Bilgisayar Mühendisliği** Anabilim Dalında bir **Yüksek lisans** olarak kabul edilmiştir.

Jüri Üyeleri (Ünvan, Ad, Soyad):	İmza
Doç. Dr. İlhan UMUT	……
Dr. Öğr. Üyesi Özlem UÇAR	……
Dr. Öğr. Üyesi Edip Serdar GÜNER	……

Tez Savunma Tarihi: 10 / 06 / 2021

Bu tezin Yüksek Lisans olarak gerekli şartları sağladığını onaylarım.

İmza

Dr. Öğr. Üyesi Özlem UÇAR

Tez Danışmanı ……

Trakya Üniversitesi Fen Bilimleri Enstitüsü onayı

……

Prof. Dr. Hüseyin Rıza Ferhat KARABULUT

Fen Bilimleri Enstitüsü Müdürü

T.Ü.FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI YÜKSEK LİSANS PROGRAMI

DOĞRULUK BEYANI

Trakya Üniversitesi Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında, tüm verilerin bilimsel ve akademik kurallar çerçevesinde elde edildiğini, kullanılan verilerde tahrifat yapılmadığını, tezin akademik ve etik kurallara uygun olarak yazıldığını, kullanılan tüm literatür bilgilerinin bilimsel normlara uygun bir şekilde kaynak gösterilerek ilgili tezde yer aldığını ve bu tezin tamamı ya da herhangi bir bölümünün daha önceden Trakya Üniversitesi ya da farklı bir üniversitede tez çalışması olarak sunulmadığını beyan ederim.

10 / 02 / 2021

Yüksek Lisans Tezi

Microsoft Dynamics Ax aracını kullanarak işletme ihtiyaçlarına yönelik uygulamaların Geliştirilmesi

T.Ü. Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

ÖZET

Günümüz teknolojisi ve bilgisi dahilinde bir şirketin ayakta kalmasını sağlayan en önemli etkenlerden birisi rekabet unsurudur. Şirketlerin büyümesi ve yaptığı işlemlerin karmaşıklaşması ile beraber rekabet unsurunu sağlayabilecek ve yönetimsel anlamda kararlar alınabilecek sistemlerin ve programların kullanılması artık kaçınılmaz bir hale gelmiştir.

Şirketlerin ERP programlarına duyduğu ihtiyaç ile beraber ERP pazarında önemli yazılım programları geliştirilmiş ve geliştirilmeye devam etmektedir. ERP programları arasında gösterebileceğimiz Dynamics Ax bu rekabet ortamında şirketlerin tercih edip kullandıkları sistemlerden birisi olmaktadır. Bu yazımızda; Analiz, Tasarım, Geliştirme ve Test aşamalarından geçen bir projede yazılımcının gözünden geliştirmede dikkat edilmesi gereken unsurlar, yapılan işlerin insan hayatını nasıl kolaylaştırdığı ve rekabet ortamlarında hızlı aksiyon alınmasının şirketler açısından önemine vurgu yapılacaktır. Stok yönetimi, tedarik zinciri, mali işler ve finans gibi şirketlerin yoğun olarak kullandığı ve işlem yoğunluklarına göre takip etmesinin zorlanıldığı kısımlarda yapılan geliştirmelerin maliyet ve zaman kavramlarını nasıl düşürdüğü gözlenecektir. Bir projenin geliştirilmesi aşamasında kalite ve performans konularında dikkat edilmesi gereken kısımlar ayrıca incelenecek ve bu geliştirmelerin ürettikleri çıktıların insanların çalışma hayatında nasıl bir fayda sağlayacağı detaylı bir şekilde aktarılacaktır.

Yıl : 2021

Sayfa Sayısı : 106

Anahtar Kelimeler : KKP, Dynamics AX, Axapta, Kurumsal Kaynak Planlaması,

Master Thesis

Development of applications for companies requirements using Microsoft Dynamics Ax

Trakya University Institute of Natural Sciences
Computer Engineering Department

ABSTRACT

One of the most important factors that ensure the survival of a company within today's technology and knowledge is possible by providing the competitive element with other companies at an adequate level. With the growth of companies and the complexity of their transactions, it is now inevitable to use systems and programs that can provide the competitive element and make managerial decisions.

With the need of companies for ERP programs, important software programs are developing in the ERP market. Dynamics Ax, which can be classify among the ERP programs, is one of the systems that companies prefer and use in this competitive environment. In this article; In a project that passes through the Analysis, Design, Development and Test phases, it will be emphasized that the factors that should be considered in a development from the eyes of the software developer, how the work done facilitates human life and the importance of taking quick action for companies. In this article we will discover how companies improve their some process about stock management, supply chain, financial operation. And how improvments effects companies processes. In a development phase, attention to quality and performance issues will be examined and explain benefits of the outputs of these improvements.

Year : 2021

Number of Pages : 106

Keywords : ERP, Dynamics AX, Axapta, Enterprise Resource Planning

ÖNSÖZ

Hazırlanan bu tez çalışmasında Analiz, Tasarım, Geliştirme ve Test aşamalarından geçen bir projenin geliştirilmesinde yazılımcının gözünden dikkat edilmesi gereken unsurlar, yapılan işlerin insan hayatını nasıl kolaylaştırdığı ve rekabet ortamlarında hızlı aksiyon alınmasının şirketler açısından önemine vurgu yapılacaktır. Stok yönetimi, tedarik zinciri, mali işler ve finans gibi şirketlerin yoğun olarak kullandığı ve işlem yoğunluklarına göre takip etmesinin zorlanıldığı kısımlarda yapılan geliştirmelerin maliyet ve zaman kavramlarını nasıl düşürdüğü gözlenecektir. Bir projenin geliştirilme aşamasında kalite ve performans konularında dikkat edilmesi kısımlar ayrıca incelenecek ve bu geliştirmelerin ürettikleri çıktıların insanların çalışma hayatında nasıl bir fayda üreteceği detaylı bir şekilde aktarılacaktır.

Çalışmanın hazırlanmasında büyük emeği geçen, çalışmanın her aşamasında bana yol gösteren, desteğini ve yardımını esirgemeyen Sayın Dr. Öğr. Üyesi Özlem UÇAR ‘a teşekkür ederim. Tez yazım aşamasında bana sabreden ve çalışmamın her adımında yardımcı olan Stok ve Tedarik Zinciri konularında bana desteği olan danışmanım Endüstri Mühendisi Davut DAL’a ve Mali İşler ve Finans konularında danışmanım Endüstri Mühendisi Kadir KARAMAN’a sonsuz teşekkür ederim.

Bu süreçte her daim yanımda olan ve manevi desteklerini hep hissettiren sevgili eşime ve aileme desteklerinden ötürü ayrıca teşekkürü borç bilirim.

İÇİNDEKİLER

ÖZET iv
ABSTRACT vi
ÖNSÖZ viii
İÇİNDEKİLER ix
SİMGELER VE KISALTMALAR DİZİNİ xii
ŞEKİLLER DİZİNİ xiii
BÖLÜM 1 1
GİRİŞ 1
BÖLÜM 2 3
KURUMSAL KAYNAK PLANLAMASI 3
2.1. Kurumsal Kaynak Planlamasının Tanımı 3
2.2. Kurumsal Kaynak Planlamasının Temel Özellikleri 4
2.2.1. Esneklik 4
2.2.2. Fonksiyonellik 4
2.2.3. Modülerlik 4
2.2.4. Bilgiye Hızlı Erişim 4
2.2.5. Evrensellik 5
2.3. Kurumsal Kaynak Planlaması Kavramının Ortaya Çıkışı 5
2.4. Kurumsal Kaynak Planlamanın Ortaya Çıkış Nedenleri 5
2.5. Kurumsal Kaynak Planlamanın İhtiyaç Nedenleri 6
2.6. Kurumsal Kaynak Planlaması Alanındaki Markalar Nelerdir? 7
2.7. Kurumsal Kaynak Planlamasına Karar Verme 9
2.8. ERP Yazılım Firmasında Yazılımcılar Ne Yapıyorlar? ERP Sorumlusu Olarak Şirketlerde Nasıl Sorumluluklara Sahipler? 10
BÖLÜM 3 12
KURUMSAL KAYNAK PLANLAMASI (ERP) PROGRAMI: DYNAMICS AX 12
3.1. Dynamicx Ax Hakkında 12
3.2. Dynamics Ax'ın Özelikleri 13
3.3. Dynamics Ax'ın Avantajları 13
3.4. Microsoft Dynamics Ax'ın Şirketlere Kazandırdıkları 14

3.4.1. Uzun Vadede Güven Desteği 14
3.4.2. Kolay Entegre 14
3.4.3. Süreçlerin Optimizasyonu 14
3.4.4. Güçlü Yapısı 15
3.4.5. Zaman Ve Maliyet Tasarrufu 15
3.4.6. Sürekli Destek 15
BÖLÜM 4 16
DYNAMICS AX ÜZERİNDE YAPILAN ÖZELLEŞTİRME ÖRNEKLERİ 16
4.1. Tahsilat Modül Geliştirmesi 16
4.2. Banka Bazında Müşteri Tahsilat Raporu Geliştirmesi 29
4.3. Dönem Sonu KDV (Katma Değer Vergisi) Tahakkuk İşlemleri 34
BÖLÜM 5 44
MICROSOFT DYNAMICS AX RAPORLAMA HİZMETİ: SSRS RAPORLARI 44
5.1. Microsoft Sql Server Raporlama Servisleri (SSRS) 44
5.2. SSRS Raporlarının Yapısı Ve Tasarımı 45
5.2.1. Rapor Veri Sağlayıcı (RDP) Sınıfı Tanımı 45
5.2.1.1. Srsreportdataproviderbase, Srsreportdataproviderpreprocess Farkları…. 45
5.2.2. Contract Class (Veri Sözleşmesi Sınıfı) 46
5.2.3. SSRS Rapor Tasarımı İçin Tablo Yapısı 46
5.2.4. SSRS Raporunun Visual Studio Ortamında Oluşturulması Ve Tasarımı. 50
BÖLÜM 6 56
DYNAMICS AX ÜZERİNDE ENTEGRASYON GELİŞTİRMESİ: WEB SERVİS 56
6.1. Web Servis İle Siparişlerin Dynamics Ax Üzerinde Oluşturulma Geliştirmesi .. 56
BÖLÜM 7 65
MICROSOFT DYNAMICS AX ÜZERİNDE DEPO GELİŞTİRMESİ: EL TERMİNALİ 65
7.1. Mutabakat Modülü Geliştirmesi Ve El Terminalinin Oluşturulması 65
BÖLÜM 8 81
MICROSOFT DYNAMICS AX ÜZERİNDE SFTP KULLANIMI 81
8.1. SFTP İle Eldeki Stok Bilgisinin Gönderme Geliştirmesi 81
8.2. Toplu İş Yapısı 81
8.3. WINSCP Kullanarak SFTP İşlemi 84
BÖLÜM 9 89
SONUÇ 89

KAYNAKLAR 91
ÖZGEÇMİŞ..... 92

SİMGELER VE KISALTMALAR DİZİNİ

AOT	Application Object Tree
ERP	Enterprise Resource Planing / Kurumsal Kaynak Planlaması
HTML	Hypertext Markup Language
IAS	Industrial Application Software
IT	Information Technologies
İMKB	İstanbul Menkul Kıymetler Borsası
KDV	Katma Değer Vergisi
MRP	Material Requirements Planning
RDP	Rapor Veri Sağlayıcı
SAP	Systems Application Product
SFTP	Secure File Transfer Protocol
SSRS	SQL Server Raporlama Servisleri
URL	Uniform Resource Loader
VS	Visual Studio
WSDL	Web Services Description Language

ŞEKİLLER DİZİNİ

Şekil 4.1. Tahsilat dönem ekranı. 17
Şekil 4.2. Tahsilat satırlarının hesaplanması. 17
Şekil 4.3. Message box yapısı. 18
Şekil 4.4. Delete_from fonksiyonu kullanımı. 18
Şekil 4.5. Veri setinin dolaşımında query nesnesi örneği. 19
Şekil 4.6. Veri setinin dolaşımında query nesnesiz sorgu örneği. 19
Şekil 4.7. Query nesneli sorgu ile query nesnesiz sorgu performans karşılaştırması. 20
Şekil 4.8. Tahsilat satırları tablosunun doldurulması. 21
Şekil 4.9. Tahsilat satırlarının görüntülenmesi. 21
Şekil 4.10. Mail gönderimi için dönemin belirlenmesi. 22
Şekil 4.11. Mail gönderme fonksiyonu için referans kabul edilecek dönemin tekilliği. 22
Şekil 4.12. Mail gönderim butonları. 23
Şekil 4.13. Mail içeriğinin kontrolünün sağlanması, oluşturulması ve gönderilmesi. 24
Şekil 4.14. Mail mesajının oluşturulması 25
Şekil 4.15. Mailin gönderimi için oluşturulan metot. 26
Şekil 4.16. Mailin içeriği. 27
Şekil 4.17. Ödeme sözü sorgu raporları. 28
Şekil 4.18. Ödeme sözünü kısmen yerine getiren müşteriler raporunun kod bloğu. 28
Şekil 4.19. Ödeme sözünü kısmen yerine getiren müşteriler formu. 29
Şekil 4.20. Tablo tipleri. 30
Şekil 4.21. Insert_recordSet ve update_recordSet fonksiyonları ile veri oluşturulması. 32
Şekil 4.22. While select fonksiyonu ile veri oluşturulması. 33
Şekil 4.23. Performans karşılaştırması. 33
Şekil 4.24. Banka bazında müşteri tahsilat formu. 34
Şekil 4.25. Ana hesap kurulumu için tablo ve formlarının oluşturulması. 35
Şekil 4.26. VatToBeReduce Enum değeri. 36
Şekil 4.27. CalculatedVat Enum değeri. 36
Şekil 4.28. DeferredVat Enum değeri. 37
Şekil 4.29. TaxesAndFundsPayable Enum değeri. 37
Şekil 4.30. KDV tahakkkuk kurulumu formu. 38
Şekil 4.31. KDV tahakkuk dönemleri formu. 38
Şekil 4.32. Tahakkuk hesapla – ana hesap bazlı fonksiyonu kod bloğu. 40
Şekil 4.33. KDV tahakkuk dönem satırları formu. 41
Şekil 4.34. KDV muhasebe günlükleri formu. 42
Şekil 4.35. Muhasebe kaydı oluşturma kod bloğu. 43
Şekil 5.1. SSRS rapor tasarımı için tablo özellikleri. 47
Şekil 5.2. Contract class tanımlaması. 47
Şekil 5.3. Contract class metotu. 48
Şekil 5.4. RDP sınıfı içerisinde tanımlanan classdeclaration metotu. 48
Şekil 5.5. RDP sınıfı içerisinde tanımlanan getDmrTransferJournalReport metotu. 49
Şekil 5.6. RDP sınıfı içerisinde tanımlanan processReport metotu. 49

Şekil 5.7. Controller sınıfı içerisinde tanımlanan main metotu. 50
Şekil 5.8. Visual studio ssrs raporu oluşturma. 50
Şekil 5.9. SSRS rapor tasarımına dataset eklenmesi. 51
Şekil 5.10. SSRS raporuna dataset özelliği. 51
Şekil 5.11. SSRS rapor tasarımında kullanılacak alanların eklenmesi. 52
Şekil 5.12. SSRS rapor tasarımında dataset alanlarının oluşması. 52
Şekil 5.13. Rapor tasarımının oluşturulması. 53
Şekil 5.14. Rapor tasarımının oluşturulması. 53
Şekil 5.15. SSRS rapor çıktısının kullanıcı tarafından görüntülenmesi. 54
Şekil 5.16. İndirimli ürün fiyat etiketi. 55
Şekil 6.1. Service için contract class örneği. 56
Şekil 6.2. Web servisten veri alma. 57
Şekil 6.3. Satış siparişinin başlığını oluşturma. 59
Şekil 6.4. Sipariş satırlarının oluşturulması. 59
Şekil 6.5. Servis oluşturma ve gerekli tanımlamaların yapılması. 60
Şekil 6.6. Oluşturulan metotlarının servise eklenmesi. 60
Şekil 6.7. Service groups oluşturma ve özellikleri. 61
Şekil 6.8. Service groups ile service ilişkisinin kurulması. 62
Şekil 6.9. Servislerin deploy edilmesi. 62
Şekil 6.10. Servisi etkinleştirme. 63
Şekil 6.11. Sipariş oluşturma web arayüzü. 64
Şekil 7.1. Sürecin yönetileceği ekran. 66
Şekil 7.2. Alıcı ve göndericiye göre filtreleme fonksiyonları. 66
Şekil 7.3. Alıcıda bekleyenleri filtrele fonksiyonu. 67
Şekil 7.4. Tüm mutabakatları işle fonksiyonu. 67
Şekil 7.5. Tüm mutabakatları işle fonksiyonu kod bloğu. 68
Şekil 7.6. Eksik beyanı geri çek fonksiyonu. 69
Şekil 7.7. Eksik beyanı geri çek butonu ile açılan form. 70
Şekil 7.8. Eksik beyanı geri çek buton fonksiyonu kod bloğu. 71
Şekil 7.9. Eksik beyanı kabul et fonksiyonu. 72
Şekil 7.10. Eksik beyanı kabul et butonu ile açılan form. 72
Şekil 7.11. Eksik beyanı kabul et butonu fonksiyonu kod bloğu. 73
Şekil 7.12. Fazla gönderim fonksiyonu. 74
Şekil 7.13. Fazla gönderim fonksiyonu kod bloğu. 75
Şekil 7.14. Girişte kolisi bekleyen ürünler fonksiyonu. 76
Şekil 7.15. Mutabakattakileri görüntüle fonksiyonu. 76
Şekil 7.16. Transfer mutabakatı el terminali ekranı. 77
Şekil 7.17. Displayform() metot el terminali işlemleri. 78
Şekil 7.18. BuildFirstPage() metotu el teminali veri giriş ekranı alanları. 79
Şekil 7.19. El terminali ekranında girilen değerlerin validasyonu. 80
Şekil 8.1. Toplu iş sınıfı görünümü. 81
Şekil 8.2. DmrExportTxtBPInventOnHandFTPBatch Classı ClassDeclaration örneği. 82
Şekil 8.3. DmrExportTxtBPInventOnHandFTPBatch Classı CanGoBatchJournal() metot. 82
Şekil 8.4. DmrExportTxtBPInventOnHandFTPBatch Classı Pack() ve unpack() metot. 83
Şekil 8.5: DmrExportTxtBPInventOnHandFTPBatch Classı Run() metot. 83
Şekil 8.6. DmrExportTxtBPInventOnHandFTP Class run() metot. 84
Şekil 8.7. DmrExportTxtBPInventOnHandFTP Class create() metotu. 86
Şekil 8.8. .txt dosyasının verilerinin yazdırılması. 86
Şekil 8.9. Winscp üzerinden oluşan .txt dosyasının gönderimi. 87

BÖLÜM 1

GİRİŞ

Globalleşen dünyanın getirdiği büyüklük ve karmaşıklık karşısında şirketlerin bu rekabet ortamına ayak uydurması zorunlu hale gelmiştir. Ülkemizde de bu rekabet ortamı fazlası ile hissedilmekte ve farkındalığı yönetimsel anlamda belli olmaktadır. Şirket yapılarının da bu büyük rekabet ortamında kendi sistemlerinin verimliliği sorgulanmaktadır. Yönetimsel anlamda merkezi tek bir sisteme dayalı ve iç içe geçmiş entegre yapılar basit ve kullanışlı birçok metodu beraberinde getirmiştir. Bu sistemlerin başında da şirketlerin kullandığı ERP (Kurumsal Kaynak Planlaması) programı olmaktadır. ERP Programlarının stabil ve kullanıcı dostu olması, yapılarının şirket bazında kolayca özelleştirilebilir bulunması kullanım açısından şirket bazında büyük fayda sağlamaktadır. Bu kısımda devreye giren yazılım ve sistem ekipleri gerektiği kısımlarda kalite, maliyet ve zaman unsurlarını da optimize edecek yapılar ile şirketlerin daha yönetilebilir ve hızlı karar alınmasına destek vermektedir.

Bu yazımızda şirketlerin kullandıkları ERP Programlarında müşteri tarafındaki taleplere olan yaklaşım ve bir proje aşamasında analiz, tasarım, test ve geliştirme kısımlarındaki yazılımcının gözünden iş süreçlerinin iyileştirilmesi amaçlanmıştır. Geliştirmenin analiz aşamasından son kullanıcı testine kadar performans ve “best practice” adımları gözetilerek uygulanması sağlanmıştır. Çalışma içerisinde bahsedilen örnek ve metotlar ile beraber son kullanıcısının isteklerine uygunluğu kontrol edilmiş ve faydaları sonuç olarak ayrıntılı bir şekilde belirtilmiştir.

Çalışmamız 9 ana bölümden oluşmaktadır. Birinci bölümde tezle ilgili genel bilgi sunulmaktadır. İkinci bölümde ERP sistemlerinin tanımı, genel özellikleri, ortaya çıkışı, ihtiyaç nedenleri gibi konular anlatılmıştır. Üçüncü bölümde Microsoft Dynamics AX ERP yazılımı özellikleri, avantajları, şirketlere kazandırdığı gibi konular anlatılmıştır. Dördüncü bölümde ise Dynamics AX yazılımının kullanıcı isteklerine göre uyarlanma örneklerinin yanında performanslı kod için dikkate alınması gereken konular kıyaslanarak açık bir şekilde belirtilmiştir. Beşinci bölümde Microsoft Dynamics AX üzerinde nasıl raporlar oluşturabileceğimizi ayrıntılı bir şekilde incelemiş olup seçilen fonksiyonlar ile bilgiler verilmiştir. Altıncı bölümde Microsoft Dynamics AX ile özelleştirilmiş bir web servis üzerinden veri alışverişine ve kullanılan metotlara değinilmiştir. Yedinci bölümde stok takibinin yapılmasına ve bunu el terminali ile olan bir uygulamasına yer verilmiştir. Sekizinci bölümde ise oluşturulan bir .txt dosyasını SFTP (Secure File Transfer Protocol) protokolünü kullanarak ortak bir dosyaya bırakma çalışması gerçekleştirilmiştir. Dokuzuncu bölümde sonuç yer almaktadır.

BÖLÜM 2

KURUMSAL KAYNAK PLANLAMASI

2.1. Kurumsal Kaynak Planlamasının Tanımı

ERP en genel anlamı ile şirketin işinin önemli ve kritik bileşenlerini bir arada ve bütün bir şekilde yönetilmesi sürecini ifade eder. ERP yazılım kavramı ise üretim planlama, satın alma, stok yönetimi, satış, pazarlama stratejisi, finans, mali işler, bütçe ve insan kaynakları gibi farklı iş süreçlerini birbirine bütün olarak bağlayan yazılımları ifade eder. ERP'nin tanımına farklı bir bakış açısı ile; "Bir kurumun bütününde ve ekosisteminde akan tüm bilginin tamamen entegre ve birbiriyle %100 bütünlüklü, gerçek zamanlı kayıt ve kontrol altına alınmasıdır" diye tanımlayabiliriz.

ERP sistemleri üretim ve hizmet sektörlerinde maliyetlerine göre farklılaşmak ile beraber her şirkete uyarlanabilir ve özelleştirilebilir yapıdadır. Yapılan ERP tanımı araştırmalarımda karşıma çıkan ve ERP'yi gerçek anlamda net bir şekilde açıklayan örnek tanımı şu şekildedir: ERP yazılımı bir işletmenin merkezi sinir sistemi görevini görür. Nasıl ki insan vücudundaki duyu organları algıladıkları duyuları beyne gönderir ve beyin bu verilerle belli kararlar alır ve harekete geçer, işte işletmenin tüm birimlerini ve ekosistemini kapsayan bir ERP programı da farklı birimlerdeki çalışanların her birinin kendi işine ait operasyonları yönetebildiği, ancak verinin baştan sona organizasyon bütününde aktığı ve bu sayede bir birimde oluşan bilginin diğer birimlerce verimli şekilde erişilip kullanılabildiği ve yöneticilerin geleceğe ilişkin tahminler yapıp karar vermesini kolaylaştıran raporlara dönüştüğü bilgi sistemidir. (workcube,2018)

2.2. Kurumsal Kaynak Planlamasının Temel Özellikleri

ERP sistemleri ülkemizde özellikle orta ve büyük işletmelerde yaygınlaşmakta ve buna paralel olarak küçük diyebileceğimiz birçok sektörde de talep edilir hale gelmektedir. ERP sistemleri firmaların farklı ihtiyaçlarına çözüm getirebilecek şekilde özelleştirilebilirler. Ama genel yapı itibarı ile sistemin temel özellikleri şu şekilde sıralanabilir:

2.2.1. Esneklik

İşletmeler sistemleri kendi çıkarları ve kararları doğrultusunda kullanabilme olanağına sahiptirler. Sistemin geniş bir fonksiyon ve alternatif iş yapısı yönetiminden, kazançları ve gelirleri doğrultusunda ihtiyaç duydukları modülleri uygulayabilmektedirler. Entegrasyon ERP sistemleri; modülleri sayesinde kesintisiz ve gerçek zamanlı veri akışı sağlayabilmektedir. Bununla birlikte kullanışlı ara yüzü sayesinde şirketin operasyonel ve yönetimsel kontrol edilebilirliğini arttırır.

2.2.2. Fonksiyonellik

ERP sistemlerinin şirketin gerek duyduğu ihtiyaçları için olmazsa olmaz fonksiyonellikleri vardır. Bunlar sırasıyla Finans ve Mali işler Yönetimi, Stok ve Tedarik Zinciri Yönetimi, Üretim ve Depolama Yönetimi, Bütçe Yönetimi, şeklinde sıralanabilir. Fonksiyonelliğin iş süreç verimliliği ve bilgi bütünlüğü açısından önemi büyüktür.

2.2.3. Modülerlik

ERP sistemleri modüler yapıya sahip olup içeriğinde bir çok işlev bulundurmaktadır. Modüller tek başlarına kullanılabileceği gibi şirket ihtiyaçlarına göre özelleştirilebilen yazılım çözümleridir.

2.2.4. Bilgiye Hızlı Erişim

Çeşitli yapılar ile entegre olabilmesi veri akışına olanak sağlar ve bu durum gerçek zamanlı veri girişini kolaylaştırmaktadır. Bu özellik sayesinde kullanıcılar tutarlı ve doğru veriye istenilen zamanda hızlı bir şekilde ulaşabilmektedir.

2.2.5. Evrensellik

ERP danışmanlık şirketlerinin teknolojik gelişmeleri yakından takip ederek oluşabilecek her türlü değişimi ERP yazılımlarına uygulama ve entegre etme gibi bir amaçları bulunmaktadır.

Günümüzde ERP sistemleri, gerçek zamanlı, açık kaynak kod mimarisine sahip sistemlerdir. İstemciler veriyi uzaktan işleyebilme özelliğine sahiptir. Bu tür sistemler şirketlerdeki tüm kullanıcıların doğru ve gerçek zamanlı veriye ulaşabilmesine olanak sağlamaktadır. (Arslan, 2015, s. 6)

2.3. Kurumsal Kaynak Planlaması Kavramının Ortaya Çıkışı

1960'lı yıllarda üretim sistemleri envanter kontrolüne odaklanırken, genelde kuruma özgü geliştirilen çoğu yazılım paketi, geleneksel envanter kavramlarına dayanarak envanter kayıtlarının tutulmasını sağlamıştır. 1970'li yılların başında malzeme ihtiyaçlarının hesaplanması için geliştirilen MRP (Material Requirements Planning) sistemleri, üretimde planlama ve kapasiteyi de göz önünde bulundurarak dinamik bir sistem halini almış ve zaman içerisinde ek olarak sisteme bağlı malzeme değişimlerini güncelleyen teknik geliştirilmiştir. 1980'li yıllarda MRP'e finans, satınalma ve üretim planlama gibi fonksiyonların eklenmesi ile Üretim Kaynakları Planlaması (Manufacturing Resource Planning, MRP II) yaklaşımına geçilmiştir. 90'lı yılların başından günümüze, bu kavramların tamamını kapsayan bütünleşik bir kurumsal çözüm olarak, ERP yazılımları gündeme gelmiştir. ERP, yalnız üretim değil hizmet dahil tüm sektörlere hizmet vermektedir. (Arslan, 2015, s. 7) (Bayraktar, 2006, s. 692)

2.4. Kurumsal Kaynak Planlamanın Ortaya Çıkış Nedenleri

ERP yapılarının ortaya çıkış nedenleri aşağıdaki maddeler üzerinden açıklanabilir;

- Globalleşen ve uluslararası rekabet.
- Bilgi teknolojisinin sağladığı yeni olanaklar.
- Uluslararası tedarik ve dağıtım zincirlerinin yaygın ve daha verimli kullanılabilir duruma gelmesi.

• Çok tesisli organizasyonların iyi idare ve kontrol edilmesi ihtiyacı.
• Ürün ve üretim politikalarındaki rekabete bağlı değişmeler.
• Kesin verileri elde etmedeki zorluk.
• Zamanında oluşturulamayan bilgi.
• Karmaşık işletme fonksiyonlarının uygun olmayan kullanıcı arayüzü.
• Fiziki olarak dağınık imalat operasyonları.
• Uluslararası pazarlara açılma gereksinimi (yerel pazarların doyması sonucu uluslararası pazarlarda söz sahibi olarak iç pazar gücünü artırma stratejisinin sonucu olarak) (Sönmeztürk, 2008, s. 5)

2.5. Kurumsal Kaynak Planlamanın İhtiyaç Nedenleri

Özellikle belirli bir standart hedefe ulaşmayı, iş akışlarını düzenlemeyi, kontrollü bir şekilde büyümeyi ya da rekabete karşı ayakta kalmayı hedefleyen kuruluşlar, ERP programları ile daha az hata yapar, daha verimli çalışır, gerçek zamanlı, doğru ve tutarlı verilere rahatlıkla ulaşabilirler. Doğru ve güncel verilere ulaşabilmek de tüm çalışanların firmadaki iş süreçlerine kolayca hakim olabilmelerine olanak sağlar ve üst yönetime de stratejik kararlar almasında önemli ölçüde katkı sağlar. Örneğin, finans departmanındaki bir kişi ERP sisteminde, azalan ya da artan giderlerini rahatça görebilir ve bütçe planlamasını buna göre düzenleyebilir. Öte yandan girilen fatura işlemi de o anda kuruluşun muhasebe kayıtlarında görülebilir. Bu hem finans departmanının işlerini kolaylaştırır hem de yönetimin gelir ve gideri güncel olarak izlemesine olanak verir.

ERP sistemlerini tercih etmeyen veya kullanmayan firmalar, farklı departmanlarda, birbirleri ile entegre olmayan farklı yazılımlar kullanmak zorunda kalmaktadırlar. Bu işletmelerde, her departman farklı yapı ve sistemleri kullanarak şirket kayıtlarını tutmaya çalışmaktadır. Böyle bir yapıda, birçok departman benzer veya aynı kayıtları farklı sistemlere girmekte ve bu girilen veriler gerçek zamanlı olmamaktadır. Örneğin birçok işletmede, tedarikçi bilgilerine ulaşılmak istendiğinde, tedarikçi bilgilerine bir excel dosyasından, tedarikçiye yapılan ödemelere bağımsız bir muhasebe programından, gönderilen faturalara ise fiziksel arşivlerden ulaşılmaktadır. Böyle bir durumda, işletmedeki çalışan tedarikçi ile iletişim kurarken, ilgili bilgilere ulaşmak için farklı birimlerle görüşmek durumunda kalmakta ve böyle bir yapıda bu bilgilerin gerçek zamanlı ve tutarlı olduğundan emin olmak çoğu zaman takip edilemez.

Buna karşılık ERP sistemi kullanan bir işletmede çalışan, tedarikçiye ait her türlü bilgiye aynı sistem üzerinden rahatça ulaşabilmekte ve en önemlisi de bu bilgilerin gerçek zamanlı, tutarlı ve güvenilir olduğundan emin olmaktadır. İşletmeleri ERP sistemi almaya yönelten nedenler şunlardır:

• İş süreçlerinin verimliliğini arttırma beklentisi (eldeki stok takibi gibi).

• İş süreçleri (departmanlar) arasında uyumun sağlanması.

• Stratejik kararlarda gerçek zamanlı ve tutarlı bilgilere kolay erişim sağlanması.

• Bilgisayar Teknolojisi alt yapısını tek sistem altında toplayarak yönetimi kolaylaştırma.

• İş süreçlerini standartlaştırmak ve basitleştirmek.

• Teknolojik yenilikleri sağlayamamış, her biri birbirinden bağımsız işleyen yapıların yenilenmesi ve tek bir merkezden yönetilen bir sistemin kurulma ihtiyacı.

• Bütün işletmelerde kullanımı basitleştirecek, tutarlı, mantıklı ve standart arayüz beklentilerinin artması.

• İşletme maliyetlerinde azalma beklentisi.

• Arka planda tekrar eden periyodik işlemlerin otomatikleştirilmesi.

• Operasyonel, yönetimsel, stratejik kararlarda iyileşme beklentisi.

• Bölgesel olarak birbirlerine yakın olmayan birimler arası uyumun sağlama.

• Müşterilerle ve tedarikçilerle olan iletişimi güçlendirme.

• Verimli bir E-ticaret yapısı kurmak veya var olan yapıyı revize etmek. (Sönmeztürk, 2008, s. 11) (Talu, 2004)

2.6. Kurumsal Kaynak Planlaması Alanındaki Markalar Nelerdir?

Küreselleşen ERP pazarında kullanılan farklı birçok sistem yapıları bulunmaktadır. Firmalar adına ERP sistemleri üreten şirketlerden bazıları şunlardır:

SAP (Systems Application Product): Almanya yazılım ürünü olarak karşımıza çıkan SAP, ERP sektörünün en eski ve köklü firmalarındandır. 1972 yılında kurulmuş

ve %25'lik pazar pay oranı ile ERP pazarında birinci sırada yer almaktadır. Türkiye pazarında da lider konumda olan SAP, %36.3 pazardaki payı ve 15.000 kullanıcı sayısı ile geniş bir kullanım alanına sahiptir. Yaklaşık olarak 230 firma tarafından tercih edilmiş durumda ve Türkiye'de lider olarak sayılan ilk 500 firmasında ERP kullanımı olarak büyük oranda SAP tercih edilmiş durumdadır.

ORACLE: Yaygın olarak veritabanı çözümleri ile bilinen ORACLE'ın "Oracle E-Business Suit" adında tanımlanan ERP sistemi de mevcuttur. Oracle, ERP pazarının dünya çapına ikinci konumundayken, üçüncü konumda bulunan Peoplesoft (JD Edwards) şirketi ve yazılımını satın alarak, SAP'a karşı rekabet oluşturacak bir adım atmıştır. Oracle ve Peoplesoft yazılımları birleştirme çalışmaları halen devam etmektedir. Türkiye'de yaklaşık 200 şirkette kullanılmaya devam etmektedir.

MICROSOFT: Yazılım devi Microsoft, şirketin büyüme stratejisi ile son dönemde ERP pazarında yer almış ve dünya çapında yaklaşık %5 pazar payı elde etmiştir. Türkiye'de aktif olan paketler Microsoft Dynamics başlığı altında Ax, Navision, Business Central ve Dynamics 365 olarak bulunmaktadır. Pazar olarak küçük, orta ve büyük işletmelere de hitap etmeye çalışmaktadır. Kobi pazarına Navision ve Business Central paketleri ile çözüm sunmaktadır. Türkiye'de ki çözümleri Partner (çözüm ortakları) şirketler üzerinden birçok firmaya göre özelleştirilebilir hale getirmeyi hedeflemiştir.

IFS: İsveç menşeili bir ERP sistemi olan IFS, sahip olduğu bileşen mimarisi ile farklı büyüklükteki şirketlere kısa zamanlarda kurularak adım adım genişleyebilen; aralarında üretim, e-ticaret, finans ve mali işler, bürçe, insan kaynakları, tedarik zinciri yönetimi, müşteri ilişkileri yönetimi, servis yönetimi ve mühendislik gibi çözümlerin de bulunduğu 60 dan fazla iş yapısını bünyesinde bulundurmaktadır. Türkiye'deki satış, pazarlama, yerelleştirme, uygulama, müşteriye ait özelleştirme, eğitim ve danışmanlık hizmetlerini partner şirketler vermektedir. IFS Uygulamaları, Türkiye pazarında, 1995 yılından itibaren, aralarında büyük ölçekli İMKB (İstanbul Menkul Kıymetler Borsası) firmalarının da bulunduğu 50'den fazla şirket üzerinde ve 2.000'den fazla personel tarafından kullanılmaktadır.

IAS (Industrial Application Software): Almanya'da Hakan Karabiber adındaki bir Türk girişimci tarafından 1989 yılında kurulan şirket, daha sonrasında yazılım

geliştirme ve AR-GE çalışmalarını Türkiye'ye de devam ederek ERP pazarına da atılmıştır. Türkiye'de IAS şirketinin 50'den fazla müşterisi ile son yıllarda ERP'nin tamamının web(tarayıcı) üzerinden çalışmasını sağlayan java programlama dili ile geliştirilmiş yeni versiyonu CANIAS ile teknolojik üstünlüğünü kanıtlamıştır. Yabancı ERP kategorisinde yer almasına rağmen Türkiye'de geliştirildiğinden yerli ERP'lerin lokalizasyon avantajlarına da sahiptir.

ABAS: 1980 yılında Almanya'nın Karlsruhe kentinde kurulan ABAS, dünya üzerinde 2.100'ü aşkın müşteri ve 45.000'den fazla kullanıcıya 28 dilde ERP ve e-Business çözümleri ile fayda oluşturmaktadır. Eğitim, danışmanlık, uyarlama ve proje yönetimi hizmetleri ABAS'ın dünya üzerindeki 50 partner şirketi ve yaklaşık 580 eğitimli personeli tarafından destek verilmektedir.

Abas Business Software çözümleri Linux, Unix ve Windows platformlarında çalışabilmekte ve bugüne kadar yapılan teknik kurulumların yaklaşık %80'inde Linux işletim sistemi tercih edilmiştir. (endustrimuhendisligim, 2015)

2.7. Kurumsal Kaynak Planlamasına Karar Verme

ERP sistemleri açık kaynak kodlu yapılardır. Bu sistemleri almak isteyen şirketler bu kurumsal çözüme karar vermeden önce dikkat etmesi gereken en önemli nokta; kendi iş süreçlerini net ve basit olarak kurgulamaları, ihtiyaçlarına, hedeflerine, amaçladığı şeylere en uygun ve yakın yazılıma karar vermelidir. Hangi yapının daha uygun olacağına karar vermek adına aşağıdaki 5 önemli nokta düşünülmelidir:

Sebepler iyi değerlendirilmeli: Öncelik olarak yeni bir sistemi neden satın alınmak istendiği sorgulanmalıdır. Depo ve stok takibinin yeterli düzeyde sağlanamadığından mı yoksa birden fazla ve birbirlerinden bağımsız sistemlerin tutarsızlıklarının ortadan kaldırılması için mi? Sorunlar şirket özelinde farklılaşabilmektedir. Bu sorunlara çözüm yolları aranırken yeni sistemin getireceği avantajlar ile beraber kritik sorunların çözüme ulaşacağı değerlendirilmelidir.

Sistem değişikliğini bir IT (Information Technologies) kararı yerine iş kararı haline getirin: Üst yönetim çoğu zaman yazılım seçimini sadece IT biriminin sorumluluğu olarak görme hatasına düşer. Günün sonunda kullanacak ve etkilenecek tüm birimleri sürece katmaksızın işin sorumluluğunu ve yükümlülüğünü sadece IT'ye

devretmek, son seçim kararının kullanıcıların bilgisi ve desteği dışında alınması anlamına gelebilir. Firma adına alınabilecek bu büyük kararın yanlış olması organizasyon açısından kötü sonuçlara neden olabilecek durumdadır.

Doğru ekibi görevlendirin: Kurumsal yazılım işlemini ciddi olarak başlatırken bilinmesi gereken üç temel rol tanımlar; Sponsor, proje yöneticisi ve şirket içi konularda uzman kadro. Her yazılım projesinin en az bir sponsora ihtiyacı vardır. Yeni yapının destekleyeceği tüm kurumsal alanlar üzerine etkisi olan bir yönetim seviyesinde yer almalıdır. Yazılım işleyişi aynı zamanda iş sürecine başından sonuna kadar hakim ve tecrübeli bir proje yöneticisi gerektirmektedir. En son olarak firma içerisinde belirli konularda işinin ehli ve uzman kadrosu yani günlük işlemlerin nasıl yapıldığına dair detaylı bilgilere sahip çalışanları yeni proje sürecine dahil etmek gerekecektir.

İhtiyaçlar listesinin çıkarılması: Başlangıç aşamasında düşünülmesi gereken kriterler yalnızca ilgili iş süreçlerini desteklemek için gerekli olan özellikler ve fonksiyonlardır. Bunların dışında kalan her şey maliyetler de dahil ikinci sıradadır. Öncelikle iş süreçleri için gerekli ihtiyaçlar belirlenmelidir. Bu hangi ERP sistemini almanız gerektiği konusunda size zaman kazandırır.

Partner şirketlerin satış yeterliliklerini yazılım seçimi uzmanlığıyla değerlendirin: Yazılım tedarikçileri, alıcı tarafından da eşit uzmanlık seviyesinde karşılanması gereken alanlarında oldukça uzmanlaşmış profesyonel satıcılardır. Yazılımın mevcut danışman ve yazılımcı kadrosu ile yazılımın yapabilecekleri üzerinden işin doğru bir şekilde nasıl yapılabileceği hakkında bilgilendirebilir ve özel işletmelerin iş konularına bunu uygulayabilirler. Bu, firmaya getireceği sistem yapısı ile avantajları konusunda büyük ölçüde yardımcı olacaktır. (endustrimuhendisligim, 2015)

2.8. ERP Yazılım Firmasında Yazılımcılar Ne Yapıyorlar? ERP Sorumlusu Olarak Şirketlerde Nasıl Sorumluluklara Sahipler?

Sektörde çeşitli ERP programları ve bu yazılımları geliştiren ve pazarlayan firmalar mevcuttur. Bir şirket ERP yazılımı değiştirmeye ve yeni bir yapıya geçmeye karar verdiğinde sektörüne uygun bir ERP firmasının programını alır. Fakat alınan bu programlar doğrudan kullanıcıların bilgisayarlarına kurularak kullanılamaz. Her

firmanın iş süreçleri, iş kültürü farklılık gösterdiğinden firmaya uygun değişikliklerin yapılması gereklidir.

Yazılım geliştiriciler belirli noktalarda dahil olarak ERP yazılımına ve sistemine ek modüller yazar ya da değişiklikler yaparlar. Örnek olarak; bir şirket fatura kesim ekranlarında çalışanların kendisine ait bir not girebilmesi için alan isteyebilir veya belli kriterlere göre bir uyarı veya hata çıkması gibi fonksiyonlar talep edebilirler. Ek olarak piyasadan hazır ERP satın alıp şirkete özel olarak uyarlamak yerine tamamen kendine ait ve özel sıfırdan sistem yazan/yazdıran şirketlerde bulunmaktadır. Bu kısımlarda yazılım geliştiriciler devreye girmektedir.

Yazılım geliştiricinin bir yazılım veya program geliştirebilmesi için sorunun ve çözümünün analiz ekibi tarafından tespit edilip geliştiricilere iletilmesi gerekmektedir. Örneğin; gelir ve giderlerini takip edebilmesi amacı ile bir yazılıma ihtiyacı olan bir işletme varsayalım. Yazılım geliştirici kod yazabilir ama gelir gider yapısının nasıl işlenerek hesaplayabileceğini bilmeyebilir. Bir takım hesaplamalar yapılması gerekecektir. Burada hesaplamayı geliştiriciye tarif ederek yardımcı olabilecek bir iş analisti veya danışmana ihtiyaç duyulacaktır. Düzgün ve çalışan bir yapının ortaya çıkması için danışman ve geliştiricinin birbirlerini anlaması gereklidir. Danışmanın biraz yazılım ve programlama dillerine, geliştiricinin de biraz sektör ve iş süreci bilgisi olmalıdır. Kullanılan ERP sistemine göre mevcut programlama dili de firmadan firmaya farklılık gösterecektir. (endustrimuhendisligim, 2015)

BÖLÜM 3

KURUMSAL KAYNAK PLANLAMASI (ERP) PROGRAMI: DYNAMICS AX

3.1. Dynamicx Ax Hakkında

Microsoft Dynamics AX veya eski adı ile Axapta, Microsoft tarafından satın alınıp geliştirilerek günümüzdeki adını almıştır. Microsoft Dynamics yeni versiyonu olan Dynamics 365 ile web ve azure teknolojisini kullanarak ERP sektöründeki yatırımlarına ve geliştirmelerine devam etmektedir. Axapta programı, Danimarka'lı Damgaard kardeşlerin firması olan Damgaard tarafından üretilmiş ve ilk olarak Mart 1998'de Danimarka ve ABD pazarında ERP sektöründe yer edinmeye başlanarak satışa sunulmuştur. Damgaard Data 2000 yılında bir başka yazılım üreticisi olan Navision Software ile birleşerek önce Navision Damgaard daha sonra da Navision adını almıştır. Axapta, Navision 2003 yılında Microsoft tarafından satın alınması ile Microsoft Business Solutions ürün ailesine dâhil olan yazılım günümüzde 45 dili desteklemekte ve ERP sektöründe firmalar tarafından yaygın olarak kullanılmaktadır. Microsoft Business Solutions ürün grubu 2006 yılında Microsoft adını almıştır. Axapta'nın adı bu andan itibaren Dynamics AX olmuştur.

Dynamics Ax, insan kaynakları ve bordro yönetimi, müşteri ve tedarikçi ilişkileri, tedarik zinciri ve depo yönetimi, finans ve mali işler yönetimi gibi modül yapıları ile kullanıcıların ihtiyaçlarına uygun çözümler sunar. İşletmelerin özel gereksinimlerine karşı istenildiği gibi düzenlenebilen ve uyarlanabilen Dynamics Ax, teknolojiyi ve gelişimi bir rekabet aracı olarak gören tüm şirketlere değer katar. (Boztaş, 2012, s. 124) (Wikipedia, 2010)

3.2. Dynamics Ax'ın Özelikleri

Birçok Microsoft ürünleriyle (Office, SQL, Visual Studio vb.) Dynamics Ax uyumlu ve entegre bir şekilde çalışabilmektedir. Kurum çalışanlarının alıştıkları işletim sistemi olan Windows arayüzleri sayesinde görevlerini ve sorumluluklarını aksatmadan yürütebilmelerine ve bu özellik sayesinde eğitim giderlerinin minimuma düşürülmesinde önemli bir rol oynamaktadır. Tamamıyla entegre bir tasarıma sahip Dynamics Ax ile beraber tedarikçiler, müşteriler ve personeller çok daha verimli bir şekilde iş birliği çerçevesinde çalışabilmektedirler.

Dynamics Ax'ın hızlı uygulanması ve firmaya göre özelleştirilebilmesi, etkin üretim planlama ve stok takibinin yapılabilmesi, müşteri ve tedarikçi arasındaki bağlantılarının takip edilebilmesi, departmanlar arasında doğru ve tutarlı bilgi entegrasyonunun sağlanması, artan verimlilik, finans ve mali işler yönetimi, insan kaynakları ve bordro yönetimi gibi modülleri sayesinde herhangi bir programlama bilgisine gerek olmadan karmaşık 28 modül kolaylıkla kullanılabilmektedir. Sayılan bu iş süreçleri ve modüller içeriğinde birden fazla form ve rapor yapılarını da bünyesinde barındırmaktadır. İşletmelerin bu yapıları kendi sistemlerine entegre etmesi hem zaman hem de maliyet kalemlerini ciddi oranda arttıran bir gider olarak karşılarına çıkacaktır. Fakat Dynamics Ax'in hızlı özelleştirme ve efektif geliştirme araçları sayesinde, bu maliyetler ve zaman kaybı en aza indirgenmektedir. (Arslan, 2015, s. 27)

3.3. Dynamics Ax'ın Avantajları

Dynamics Ax'ın sahip olduğu bu gelişmiş özellikler aşağıda sıralanmıştır;

- Düşük maliyetler ile işlemlerin tamamlanabilmesi
- Diğer ERP sistemleri ile karşılaştırıldığında daha düşük seviyelerde bakım ihtiyacı
- Kullandığı kontrol ve doğrulama araçları ile tekrarsız veri depolama
- Birbirleri ile entegre çalışan modülleri sayesinde tek iş mantığı
- Her sektöre uygun ve firma bazında özelleştirilme kolaylığı

- Versiyonlar arası değişkenlik ve geçişin kolay olması ile beraber gelen güncelleme kolaylığı
- Esneklik ve ölçeklenebilirlik
- Yüksek Teknoloji: Internet, 3 katmanlı yapı, Microsoft teknolojileriyle tam entegrasyon kolaylığı
- Sektöre ait ve firmalara özgü kendi içlerinde çözdükleri ve bu çözümlerin paylaşıldığı ortak bir havuzdan (Microsoft Solution Finder) kendi sistemlerine entegre edebileceği (add-on) yöntemi ile ERP sistemlerine kolayca uyarlanabilmesi. (Boztaş, 2012, s. 131)

3.4. Microsoft Dynamics Ax'ın Şirketlere Kazandırdıkları

Microsoft Dynamics Ax ERP yazılımı ile işletmeler, yazılımla birlikte gelen bir çok avantajdan faydalanabilmektedir. Dynamics Ax ile gelen özellikler sayesinde işletmeler, klasik bir ERP çözümünün getirilerinin çok ötesinde avantajlara sahip olurlar. Microsoft Dynamics Ax'a ait bu farklı özellikleri aşağıda maddeler halinde inceleyebiliriz:

3.4.1. Uzun Vadede Güven Desteği

Dynamics Ax uzun yıllardır gelişimini sürdüren ve Microsoft tarafından ciddi yatırımlar alan bir ERP yazılımı konumundadır. Microsoft ve Partner şirketler ile yürüttüğü ERP yazılımının satışından, eğitimine, kurulumundan son kullanıcı desteğine kadar sunduğu mimari yapısından dolayı firmalar tarafından kabul görmektedir.

3.4.2. Kolay Entegre

Microsoft Dynamics Ax'ın en önemli değerlerinden biri de kolay entegre olabilme özelliğidir. Yani Microsoft Dynamics Ax, gereksinimlere ve iş süreçlerine en esnek biçimde yanıt verebilir ve uyum sağlayabilir.

3.4.3. Süreçlerin Optimizasyonu

Tek bir veritabanı üzerinden çalışan Microsoft Dynamics Ax, kolayca takip edilebilirliği sayesinde hem orta ölçekli hem büyük ölçekli şirketlerin kullanım amaçlarını ortak paydada buluşturabilen bir özelliğe sahiptir. Kurumsal sistemlerin

tamamında uygulanabilen fonksiyonel yapısı sayesinde Microsoft Dynamics AX, karmaşık iş süreçlerini her anında takip edebilme ve yönetme imkânı sağlar.

3.4.4. Güçlü Yapısı

Microsoft Dynamics Ax, ürün, hizmet ve araçlarının ERP sektörüne sunulmasından müşteri istek ve önerilerini uygun bir şekilde hizmet yapısına yansıtılarak karşılanmasına kadar her adımında etkin ve başarılı bir yapı ortaya koyabilmektedir.

3.4.5. Zaman Ve Maliyet Tasarrufu

Microsoft Dynamics Ax, entegre ve esnek yapısı sayesinde kurulum ve canlı kullanıma geçme aşamasında, benzer ERP çözümlerinden çok daha uygun bir maliyet ile çözüm sunmaktadır. İş süreçleri diğer ERP çözümlerine göre çok daha kısa bir sürede Microsoft Dynamics Ax üzerinde kullanıma başlanabilmektedir.

3.4.6. Sürekli Destek

Microsoft Dynamics Ax, yerel ve global partner şirketler sayesinde, her türlü ihtiyaca çözüm ve destek olabilecek nitelikte olup yerel koşulların ve mevzuatın gerektirdiği değişikliklere göre özelleştirilebilen bir sistem sunmaktadır. Aynı zamanda kurumların özel ihtiyaçları Microsoft'un yetkili partner şirketleri tarafından analizlerinin yapılıp, geliştiricilerin destekleri ile anında çözüme ulaşılarak sisteme yansıtabilmektedirler. (Arslan, 2015, s. 29)

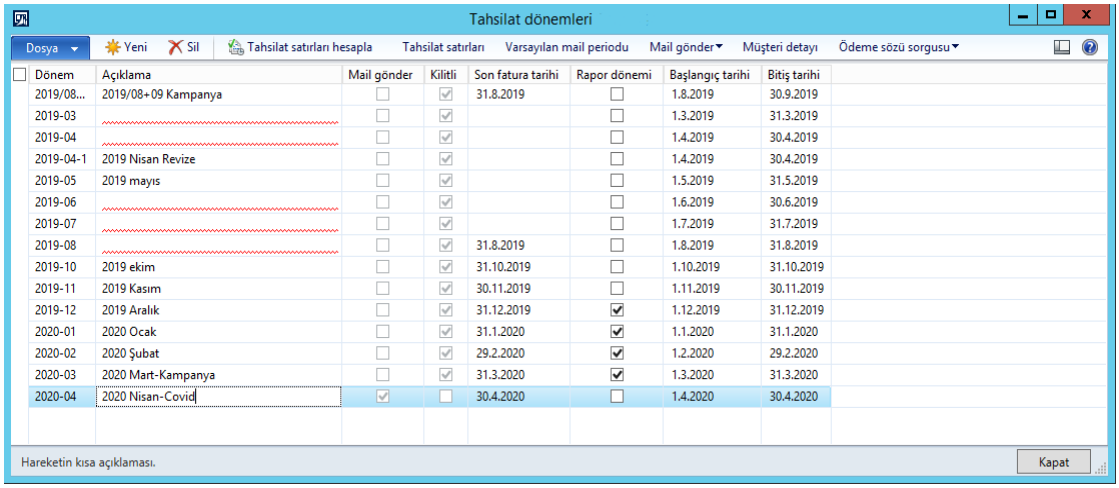
BÖLÜM 4

DYNAMICS AX ÜZERİNDE YAPILAN ÖZELLEŞTİRME ÖRNEKLERİ

4.1. Tahsilat Modül Geliştirmesi

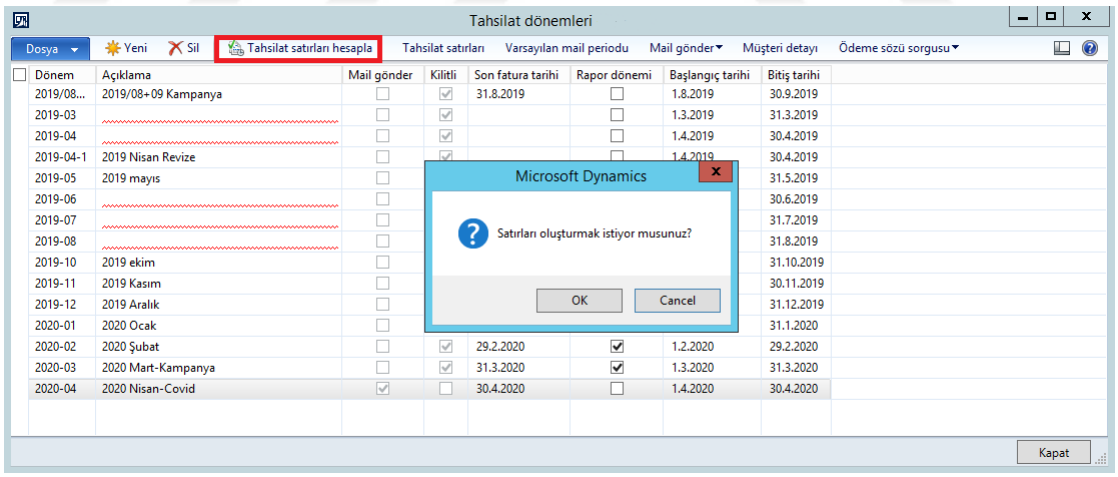
Dynamics Ax programını kullanan şirket çalışanlarının ihtiyaçları doğrultusunda bazı talepleri olmaktadır. Bu talepler çoğu zaman çalışanların işlerini kolaylaştırıcı ve zaman olarak yapılması uzun süren rapor içeriklerin hazırlanması olarak karşımıza çıkmaktadır. Çoğu zaman ilgili istekler programın danışmanı ve yazılımcı ekiplerine bildirilerek yapılması istenmektedir. Aşağıda bu koşula uyan bir talebin yazılımcılar tarafından geliştirme yapılması sonucunda kullanıcının ihtiyacının nasıl karşılandığı ve yapılan işin özelliklerinden bahsedilmiştir.

Müşteriden gelen tahsilatların ve ödemelerin raporlandığı, tek bir ekran üzerinden yönetilen, kullanıcı dostu bir form yapısı talep edilmektedir. Bu talebe karşılık aşağıdaki form üzerinden tahsilat satırlarının hesaplandığı, görüntülendiği ve gerektiğinde bu kayıtların ilgili şube müdürlerine, bölge müdürlerine ve genel müdürlerine mail olarak excel çıktısı gönderildiği bir yapı görüntülenmektedir.



Şekil 4.1. Tahsilat dönem ekranı.

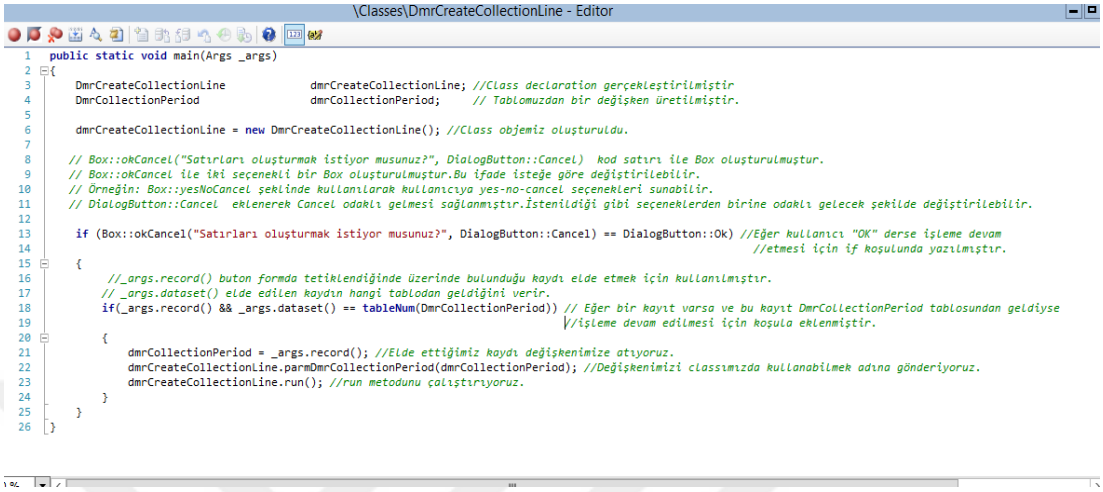
Şekil 4.1'de görüntülenebildiği üzere tahsilatların dönem bazında kullanıcıların kendilerine ait raporlama yapabildiği esnek bir yapısı mevcuttur. Bu rapor üzerine, tek bir mali dönem olabildiği gibi istenilen iki tarih arasında da ilgili tarih filtrelerini kullanarak raporlama yapma imkanı sunmaktadır.



Şekil 4.2. Tahsilat satırlarının hesaplanması.

Şekil 4.2'de rapor formuna başlangıç olarak kullanıcıların raporlama yapacağı tarih filtrelerini girmesi beklenmektedir. Bu bilgiler sonucunda 4.2'deki "Tahsilat satırları hesapla" butonuna basılarak kullanıcının karşısına bir "message box" açılarak

işlem için ek bir onay istenmektedir. Bu ekran üzerinde kullanıcın “OK” veya “Cancel” seçimine göre işlemler sürdürülmektedir.




```
public static void main(Args _args)
{
    DmrCreateCollectionLine            dmrCreateCollectionLine; //Class declaration gerçekleştirilmiştir
    DmrCollectionPeriod                dmrCollectionPeriod;     // Tablomuzdan bir değişken üretilmiştir.

    dmrCreateCollectionLine = new DmrCreateCollectionLine(); //Class objemiz oluşturuldu.

   // Box::okCancel("Satırları oluşturmak istiyor musunuz?", DialogButton::Cancel)  kod satırı ile Box oluşturulmuştur.
   // Box::okCancel ile iki seçenekli bir Box oluşturulmuştur.Bu ifade isteğe göre değiştirilebilir.
   // Örneğin: Box::yesNoCancel şeklinde kullanılarak kullanıcıya yes-no-cancel seçenekleri sunabilir.
   // DialogButton::Cancel  eklenerek Cancel odaklı gelmesi sağlanmıştır.İstenildiği gibi seçeneklerden birine odaklı gelecek şekilde değiştirilebilir.

    if (Box::okCancel("Satırları oluşturmak istiyor musunuz?", DialogButton::Cancel) == DialogButton::Ok) //Eğer kullanıcı "OK" derse işleme devam
                                                                                                          //etmesi için if koşulunda yazılmıştır.
    {
        //_args.record() buton formda tetiklendiğinde üzerinde bulunduğu kaydı elde etmek için kullanılmıştır.
       // _args.dataset() elde edilen kaydın hangi tablodan geldiğini verir.
        if(_args.record() && _args.dataset() == tableNum(DmrCollectionPeriod)) // Eğer bir kayıt varsa ve bu kayıt DmrCollectionPeriod tablosundan geldiyse
                                                                               //işleme devam edilmesi için koşula eklenmiştir.
        {
            dmrCollectionPeriod = _args.record(); //Elde ettiğimiz kaydı değişkenimize atıyoruz.
            dmrCreateCollectionLine.parmDmrCollectionPeriod(dmrCollectionPeriod); //Değişkenimizi classımızda kullanabilmek adına gönderiyoruz.
            dmrCreateCollectionLine.run(); //run metodunu çalıştırıyoruz.
        }
    }
}
```

Şekil 4.3. Message box yapısı.

Şekil 4.3 ile kullanıcının seçmesi için açılan bir message box içeriğinin nasıl yazılması gerektiği hakkında bir kod bloğu paylaşılmıştır.

```
delete_from dmrCollectionLine
where dmrCollectionLine.PeriodCode == dmrCollectionPeriod.PeriodCode;

delete_from dmrCustCollectionTable
where dmrCustCollectionTable.PeriodCode == dmrCollectionPeriod.PeriodCode;
```

Şekil 4.4. Delete_from fonksiyonu kullanımı.

Şekil 4.4’deki kod bloğunda; rapor çalıştırılırken daha önce ilgili periyod üzerinde var olan kayıtların silinmesi amaçlanarak doğru kayıtların getirilmesi sağlanmaktadır.

```
company = curext();  //Mevcutta çalışılan şirketin kodunu verir.
while select DmrCompny //Tablonun verilerinin satır satır dolaşıldığı kod satırı
{
    changeCompany(DmrCompny.company) //Şirket değiştirme
    {
        q = new Query(); //query nesnesinin oluşturulması
        qbds = q.addDataSource(tableNum(CustTable)); //Sorguda kullanılacak olan tablonun eklenmesi
        qbds.addOrderByField(fieldNum(CustTable,AccountNum)); //AccountNum alanına göre sıralama
        qbds.addRange(fieldNum(CustTable, DmrDomestic)).value(queryValue(NoYes::No));//DmrDomestic alanının No olması için verilen filtre
        qRun = new QueryRun(q); //Query nesnesinin yürütülmesi için nesne oluşturulması
        ttsBegin;

        while(qRun.next()) //Query içindeki verilerin dolaşılması
        {
            isInsertLine = false;
            custTable = qRun.get(tableNum(CustTable)); //CustTable kayıtlarının elde edilmesi

                this.calculateAmounts(custTable); //Hesaplama için verilerin gönderilmesi

        }
        ttsCommit;
    }
}
```

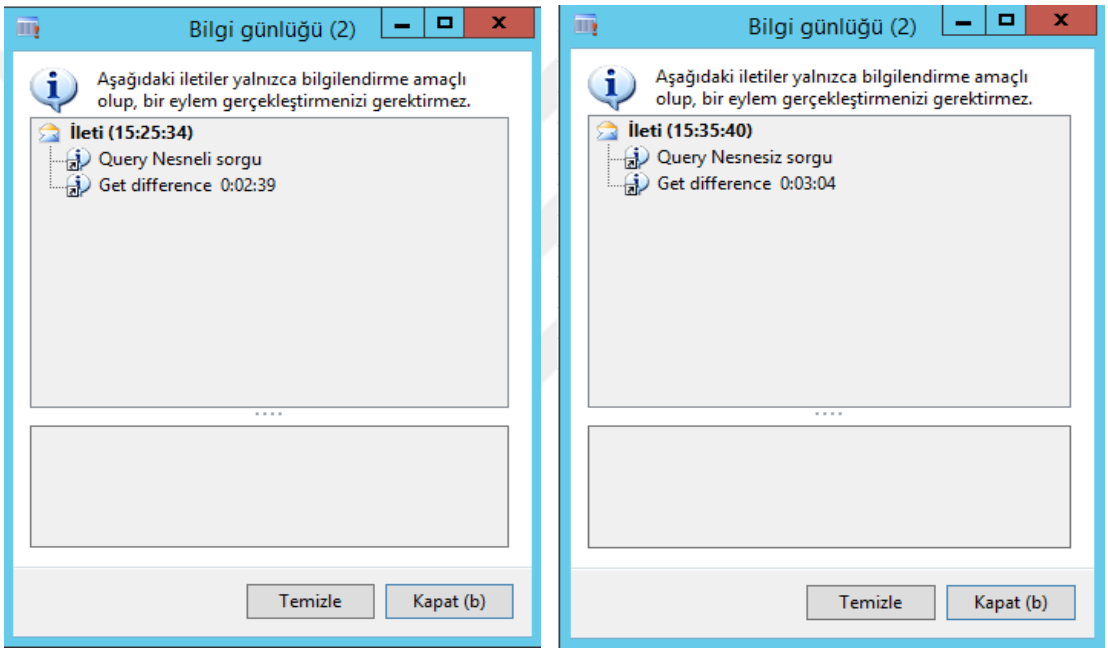
Şekil 4.5. Veri setinin dolaşımında query nesnesi örneği.

Şekil 4.5'de gösterilen "while select" fonksiyonu kullanılarak "DmrCompny" tablosundaki veriler dolaşılmaktadır. Her bir satırın "company" alanında yazan şirket koduna "changeCompany" fonksiyonu kullanılarak gidilmiştir. Her bir şirkette "CustTable" tablosu query nesnesi kullanılarak "accountNum" alanına göre sıralanmış ve "DmrDomestic" alan değeri "No" olacak şekilde filtre verilerek dolaşılmış olup bulunan her bir kaydın hesaplanması için "calculateAmounts()" metoduna gönderilmiştir. Bu işlemlerin hepsi "Query" sorgu nesnesi kullanılarak gerçekleştirilmiştir.

```
public void run2()
{
    Query                       q;
    QueryRun                    qRun;
    QueryBuildDataSource        qbds;
    CustTable                   custTable;
    DmrCustCollectionTable      dmrCustCollectionTable;
    DmrCollectionLine           dmrCollectionLineGroup,dmrCollectionLineName;
    DmrProjTransferTable        dmrProjTransferTable;

    ;
    delete_from dmrCollectionLine
    where dmrCollectionLine.PeriodCode == dmrCollectionPeriod.PeriodCode;

    delete_from dmrCustCollectionTable
    where dmrCustCollectionTable.PeriodCode == dmrCollectionPeriod.PeriodCode;

    company = curext();  //Mevcutta çalışılan şirketin kodunu verir.
    while select DmrCompny //Tablonun verilerinin satır satır dolaşıldığı kod satırı
    {
        changeCompany(DmrCompny.company) //Şirket değiştirme
        {
            ttsBegin;
            while select custTable //CustTable Tablosunun verilerinin satır satır dolaşıldığı kod satırı
            order by AccountNum    //AccountNum alanına göre sıralama
            where custTable.DmrDomestic == NoYes::No //DmrDomestic alanının No olması için verilen filtre
            {
                this.calculateAmounts(custTable);
            }

            ttsCommit;
        }
    }
```

Şekil 4.6. Veri setinin dolaşımında query nesnesiz sorgu örneği.

Şekil 4.6‘da gösterilen “while select” fonksiyonu kullanılarak “DmrCompny” tablosundaki veriler dolaşılmaktadır. Her bir satırın “company” sütununda yazan şirket koduna “changeCompany” fonksiyonu kullanılarak gidilmiştir. Her bir şirkette “CustTable” tablosu “while select” fonksiyonu kullanılarak “accountNum” alanına göre sıralanmış ve “DmrDomestic” alanı “No” filtresi verilerek veri dolaşımı sağlanmış olup bulunan her bir kaydın hesaplanması için “calculateAmounts()” metoduna gönderilmiştir. Bu işlemlerin hepsi Query sorgu nesnesi kullanılmayarak gerçekleştirilmiştir.



Şekil 4.7. Query nesneli sorgu ile query nesnesiz sorgu performans karşılaştırması.

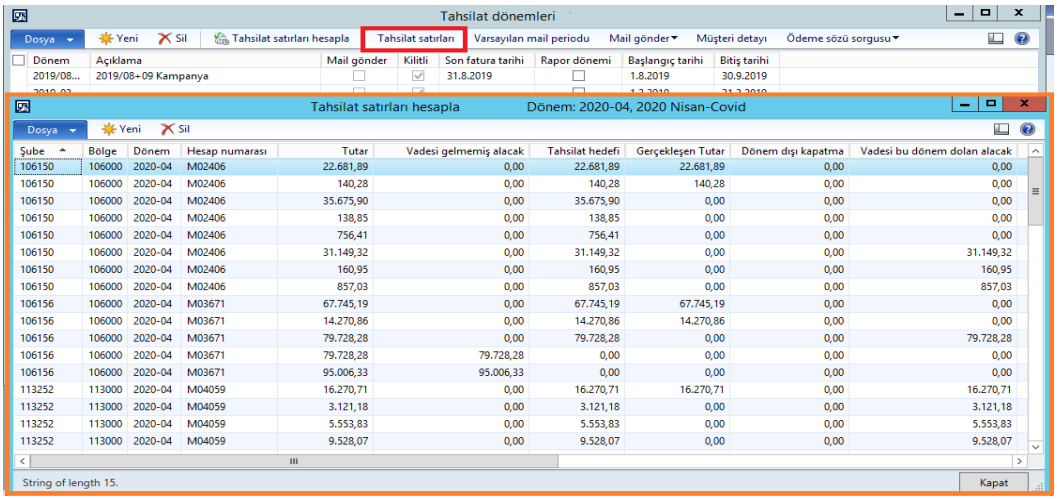
Şekil 4.7’de gösterildiği gibi Query sorgu nesnesi kullanılan sorgunun daha performanslı çalıştığı gözlenmiştir. Bu sebeple Query nesnesine sorgularımızda yer vermek raporların performanslı ve verimli çalışmasını olumlu etkilediğini söylemekte fayda vardır. Ek olarak, Şekil 4.5’de gösterilmiş olan kod bloğunun Şekil 4.6’da gösterilen kod bloğuna göre yazımının daha uzun eforlar harcattığı söylenebilir. “Best practice” olarak kod bloglarının yazımında Query sorgu nesnelerini kullanmayı tercih etmeye özen göstermemiz gerektiği sonucuna ulaşabiliriz. Şekil 4.7’de elde edilen

sonuçlar az veriler ile incelenmiş olup çok kayıtlı tablolarda arasındaki zaman farkının daha fazla olacağını belirtmekte fayda vardır.

```
private void insertLine(CustTable  _custTable,CustTrans _custTrans,str 30 _sube,str 30 _bolge)
{
    ProjFundingSource     projFundingSource;

    AccountNum accountnum = _custTable.AccountNum;
    CustName   name       = _custTable.name();


      changeCompany(company)
      {

           dmrCollectionLine.clear();

           dmrCollectionLine.AccountNum      = accountnum;
           dmrCollectionLine.CustName        = name;
           dmrCollectionLine.PeriodCode      = dmrCollectionPeriod.PeriodCode;
           dmrCollectionLine.Company         = DmrCompny.company;

           projFundingSource.clear();
           select firstOnly projFundingSource
           where projFundingSource.CustAccount == accountnum;

           dmrCollectionLine.PaymentTerm     = projFundingSource.PaymentTermsId;
           dmrCollectionLine.Amount2         = amount;
           dmrCollectionLine.sube            = _sube;
           dmrCollectionLine.Bolge           = _bolge;
           dmrCollectionLine.RefRecId        = _custTrans.RecId;
           dmrCollectionLine.DueDate         = _custTrans.DueDate;
           dmrCollectionLine.AmountMST       = Amount;

           dmrCollectionLine.insert();
```

Şekil 4.8. Tahsilat satırları tablosunun doldurulması.

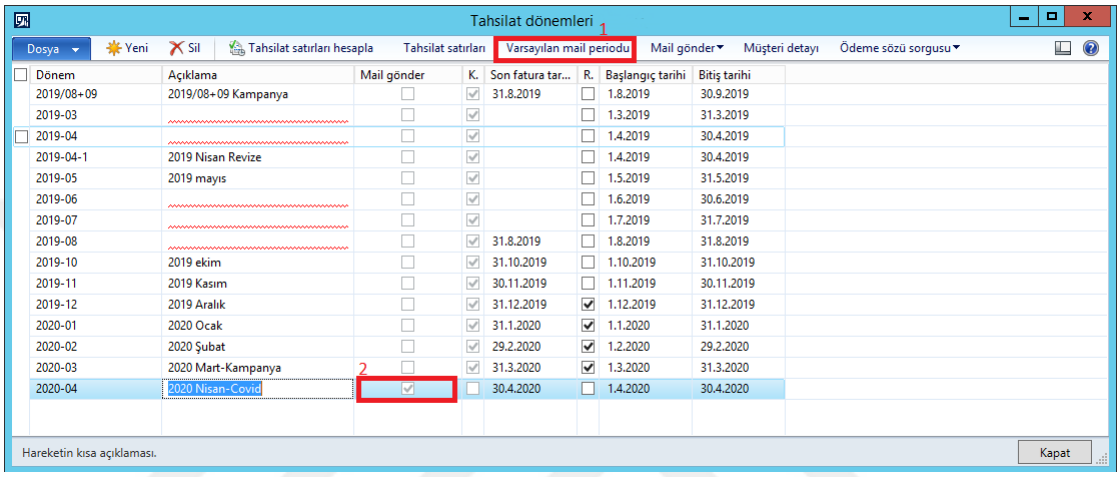
Şekil 4.8 ‘de “calculateAmounts()” metoduna gönderilen CustTable kayıtları gerekli hesaplamalar yapılarak mevcutta bulunulan şirketten “changeCompany()” fonksiyonu kullanılarak istenilen şirkete geçilmesi sağlanarak tabloya tahsilat satırları kayıtları atılmıştır.



Şekil 4.9. Tahsilat satırlarının görüntülenmesi.

Şekil 4.9'da üzerinde bulunulan dönemin daha önceden çalıştırılan sorgu üzerinden hesaplanmış kayıtları "tahsilat satırları" butonu üzerinden kullanıcının görüntülemesi ve kontrol etmesi sağlanmıştı. Görüntülenen "Tahsilat satırları hesapla" formu "Tahsilat satırları" butonuna basıldığında ekrana getirilmiştir.

Not: Bu formda Şekil 4.8'de oluşturduğumuz tahsilat satırları kayıtları listelenmektedir.



Şekil 4.10. Mail gönderimi için dönemin belirlenmesi.

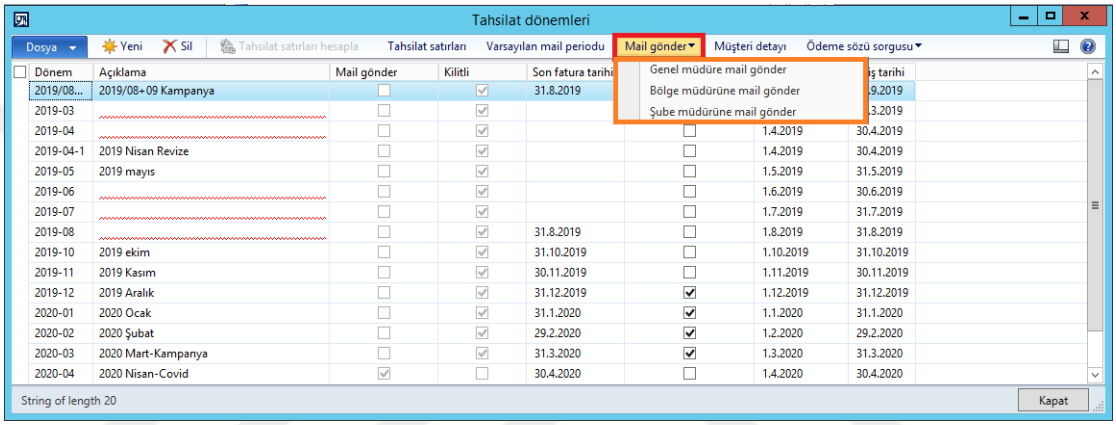
Şekil 4.10'da (1) ile belirtilen "Varsayılan mail periodu" butonuna tıklandığında (2) olarak belirtilen "Mail gönder" checkbox alanı işaretlenmektedir.

Mail gönderim işlemi gerçekleştirilirken bu alan değeri kontrol edilecek, alanın işaretli olduğu dönem mail gönderimi için işleme alınacağı belirtilmiştir.

```
void clicked()
{

    ttsBegin;

    select forUpdate DmrCollectionPeriodOther //Sütunda başka SentMail alanı işaretli kayıt var mı diye kontrol edildi.
    where  DmrCollectionPeriodOther.SentMail;//Alanın işaretli olup olmaması filtresi
    if(DmrCollectionPeriodOther) //Eğer SentMail alanı işaretli kayıt varsa if koşuluna girer.
    {
        DmrCollectionPeriodOther.SentMail = NoYes::No; //Bulunan kayıt No olarak güncelleniyor.
        DmrCollectionPeriodOther.update();//Güncelleme işlemi yapılmaktadır.
    }

    DmrCollectionPeriod.selectForUpdate(true); //Üzerinde bulunulan satır seçilir.
    DmrCollectionPeriod.SentMail = NoYes::Yes;// Üzerinde bulunulan satırın SentMail alanı Yes olarak güncellenmektedir.
    DmrCollectionPeriod.update();//Güncelleme işlemi yapılmaktadır.


    ttsCommit;
```

Şekil 4.11. Mail gönderme fonksiyonu için referans kabul edilecek dönemin tekilliği.

Mail gönderme fonksiyonunun aynı anda yalnızca bir dönem periyodu üzerinden işleme alınarak çalışması istendiğinden Şekil 4.11'deki kod bloğuna ek bir kontrol eklenmiştir. Bu kontrol işlevi; üzerinde bulunan kayıt için mail gönderme checkbox'ını işaretleme yaparken daha önce başka bir dönem üzerinde işaretli olan alan değerini kaldırarak mail gönderme chechbox değerinin tablo üzerinde tekilliğini sağlamaktadır. Bu kontrol ile sistemin otomatik mail gönderirken tek bir dönem üzerindeki kayıtlar özelinde mail gönderilmesi ve dönem bazında kayıtların karışmaması amaçlanmaktadır.



Şekil 4.12. Mail gönderim butonları.

Şekil 4.12'de gösterilen "Mail gönder" butonunun içeriğine baktığımızda karşımıza üç fonksiyon çıkmaktadır. Bu fonksiyonlar; "Genel müdüre mail gönder", "Bölge müdürüne mail gönder" ve "Şube müdürüne mail gönder" olarak görüntülenmektedir.

Aşağıda örnek olarak "Genel müdüre mail gönder" fonksiyonu incelenmekte ve yorumlanmaktadır.

```
void run()
{

    str emailBodyForBrachManager,emailBodyForBusinessManager;
    ;
    select firstOnly dmrCollectionPeriod //Mail gönderilecek dönemin elde edilmesi
    where dmrCollectionPeriod.SentMail == NoYes::Yes;

       emailBodyForBusinessManager = this.createBusUnitManagerEmailBody(); //Mailin mesaj kısmının oluşturulması
       if(emailBodyForBusinessManager) //Eğer mailin mesaj içeriği doluysa gönderme işlemine geçilmesi için koşul
        {
           this.sendMail(emailBodyForBusinessManager);//Mailin gönderilmesi
        }

    info("Genel Müdür Mailleri Gönderildi"); //Mail gönderimi sonunda bilgilendirme mesajı
}
```

Şekil 4.13. Mail içeriğinin kontrolünün sağlanması, oluşturulması ve gönderilmesi.

İlk olarak kullanıcının üzerinde bulunan dönem için “Genel müdüre mail gönder” fonksiyonunu çalıştırdığında “Mail gönder” checkbox değerinin “Yes” veya “No” alan değerinin kontrolü gerçekleştirilmektedir. İlgili fonksiyon bu checkbox değerinin “Yes” olan kayıtlar özelinde çalıştığı kontrol edildiğinde işlemler devam ettirilerek mail içeriğinin oluşturulması ve gönderilmesi için ilgili kod blogları yazılmaktadır.

```
str createBusUnitManagerEmailBody()
{
    DmrCustTransParams          DmrCustTransParams;
    AmountMST                   totalAmountMst;
    str                         sube, bodyheadForBusUnit, bodyForBusUnit , footerForBusUnitBranchHead, footerForBusUnitBranchBody, allMailbody;
    str                         bodyheadForBusUnitBolge,bodyGeneralDetail,bodyGeneral;
    DmrCollectionLine           collectionLine;
    DmrCollectionActivities     collectionActivities;
    DmrCollectionTasks          collectionTasks,collectionTasksCompleted;
    boolean                     flag=false;
    ;
    sube="";
    select DmrCustTransParams;

    bodyheadForBusUnit = "Sayın Yetkili,<br/><br/>";
    bodyheadForBusUnit += strFmt("%1 tarihi itibari ile",DmrCustTransParams.EndDate);
    bodyheadForBusUnit += strFmt("tahsilat hedefi raporunuz aşağıda %1 dönemi için Türkiye geneli , şube ve bölge bazlı olarak belirtilmiştir.",
                          dmrCollectionPeriod.PeriodCode);
    bodyheadForBusUnit += "<br/>Saygılarımızla, <br/>";
    bodyheadForBusUnit += CompanyInfo::find().Name; //Firma adının otomatik gelmesi için yazıldı.
    bodyheadForBusUnit += " Muhasebe Departmanı <br/>";

    //Verilerin oluşumu için gerekli sorgular
    select sum(ActualAmount),sum(AmounTarget),sum(DueDateOldPeriod),sum(DueDateCurrentPeriod)   from collectionLine
    where collectionLine.Bolge && collectionLine.Sube
    &&    collectionLine.PeriodCode ==  dmrCollectionPeriod.PeriodCode;

    select count(RecId) from collectionActivities
    where  collectionActivities.PeriodCode == dmrCollectionPeriod.PeriodCode;

    select count(RecId) from collectionTasks
    where  collectionTasks.PeriodCode   == dmrCollectionPeriod.PeriodCode;

    select count(RecId) from collectionTasksCompleted
    where  collectionTasksCompleted.PeriodCode   == dmrCollectionPeriod.PeriodCode;
    //Verilerin oluşumu için gerekli sorgular
```

```
bodyGeneralDetail += '<p <font size=2 color=black face=Arial><b>TÜRKİYE GENELİ</b></p>'+'<table border = 1>';

bodyGeneralDetail += '<tr>';
bodyGeneralDetail += '<td align = "left" style="font-size:95%" font face="Verdana"><b>Vadesi Geçmiş Alacak</b></td>';
bodyGeneralDetail += '<td align = "left" style="font-size:95%" font face="Verdana"><b>Vadesi Bu Dönem Olan Tahsilat </b></td>';
bodyGeneralDetail += '<td align = "left" style="font-size:95%" font face="Verdana"><b>Hedef Tahsilat </b></td>';
bodyGeneralDetail += '<td align = "left" style="font-size:95%" font face="Verdana"><b>Gerçekleşen Tahsilat </b></td>';
bodyGeneralDetail += '<td align = "left" style="font-size:95%" font face="Verdana"><b>Kalan Tahsilat</b></td>';
bodyGeneralDetail += '<td align = "left" style="font-size:95%" font face="Verdana"><b>Tahsilat oranı </b></td>';
bodyGeneralDetail += '<td align = "left" style="font-size:95%" font face="Verdana"><b>Aktivite sayısı </b></td>';
bodyGeneralDetail += '<td align = "left" style="font-size:95%" font face="Verdana"><b>Görev sayısı </b></td>';
bodyGeneralDetail += '<td align = "left" style="font-size:95%" font face="Verdana"><b>Tamamlanan görev sayısı</b></td>';
bodyGeneralDetail += '</tr>';

    bodyGeneral  += '<tr>';
    bodyGeneral += strfmt('<td align = "left" style="font-size:90%" font face="Verdana"> %1 </td>'
    +'<td align = "left" style="font-size:90%" font face="Verdana"> %2 </td>'
    +'<td align = "left" style="font-size:90%" font face="Verdana"> %3 </td>'
    +'<td align = "left" style="font-size:90%" font face="Verdana"> %4 </td>'
    +'<td align = "left" style="font-size:90%" font face="Verdana"> %5 </td>'
    +'<td align = "left" style="font-size:90%" font face="Verdana"> %6 </td>'
    +'<td align = "left" style="font-size:90%" font face="Verdana"> %7 </td>'
    +'<td align = "left" style="font-size:90%" font face="Verdana"> %8 </td>'
    +'<td align = "right" style="font-size:90%" font face="Verdana"> %9  </td> </tr>'
    , collectionLine.DueDateOldPeriod
    , collectionLine.DueDateCurrentPeriod
    , collectionLine.AmounTarget
    , collectionLine.ActualAmount
    , (collectionLine.AmounTarget - collectionLine.ActualAmount)
    , abs((abs(collectionLine.ActualAmount)/collectionLine.AmounTarget)*100)
    , collectionActivities.RecId
    , collectionTasks.RecId
    , collectionTasksCompleted.RecId);
```

```
bodyheadForBusUnitBolge += '<p <font size=2 color=black face=Arial><b>Bölge bazlı tahsilat hedefi</b></p>'+
'<table border = 1>';

bodyheadForBusUnitBolge += '<tr>';
bodyheadForBusUnitBolge +=    '<td align = "left" style="font-size:80%" font face="Verdana"><b>Bölge Kodu</b></td>';
bodyheadForBusUnitBolge +=    '<td align = "left" style="font-size:80%" font face="Verdana"><b>Vadesi Geçmiş Alacak</b></td>';
bodyheadForBusUnitBolge +=    '<td align = "left" style="font-size:80%" font face="Verdana"><b>Vadesi Bu Dönem Olan Tahsilat</b></td>';
bodyheadForBusUnitBolge +=    '<td align = "left" style="font-size:80%" font face="Verdana"><b>Hedef Tahsilat</b></td>';
bodyheadForBusUnitBolge +=    '<td align = "left" style="font-size:80%" font face="Verdana"><b>Gerçekleşen Tahsilat</b></td>';
bodyheadForBusUnitBolge +=    '<td align = "left" style="font-size:80%" font face="Verdana"><b>Kalan Tahsilat</b></td>';
bodyheadForBusUnitBolge +=    '<td align = "left" style="font-size:80%" font face="Verdana"><b>Tahsilat oranı</b></td>';
bodyheadForBusUnitBolge +=    '<td align = "left" style="font-size:80%" font face="Verdana"><b>Aktivite sayısı</b></td>';
bodyheadForBusUnitBolge +=    '<td align = "left" style="font-size:80%" font face="Verdana"><b>Görev sayısı</b></td>';
bodyheadForBusUnitBolge +=    '<td align = "left" style="font-size:80%" font face="Verdana"><b>Tamamlanan görev sayısı</b></td>';
bodyheadForBusUnitBolge += '</tr>';

footerForBusUnitBranchHead += '<p <font size6e=2 color=black face=Arial><b>Şube Bazlı Tahsilat Hedefi</b></p>'+
'<table border = 1>';
footerForBusUnitBranchHead += '<tr>';
footerForBusUnitBranchHead +=    '<td align = "left" style="font-size:80%" font face="Verdana"><b>Bölge kodu</b></td>';
footerForBusUnitBranchHead +=    '<td align = "left" style="font-size:80%" font face="Verdana"><b>Şube Kodu</b></td>';
footerForBusUnitBranchHead +=    '<td align = "left" style="font-size:80%" font face="Verdana"><b>Vadesi Geçmiş Alacak</b></td>';
footerForBusUnitBranchHead +=    '<td align = "left" style="font-size:80%" font face="Verdana"><b>Vadesi Bu Dönem Olan Tahsilat</b></td>';
footerForBusUnitBranchHead +=    '<td align = "left" style="font-size:80%" font face="Verdana"><b>Hedef Tahsilat</b></td>';
footerForBusUnitBranchHead +=    '<td align = "left" style="font-size:80%" font face="Verdana"><b>Gerçekleşen Tahsilat</b></td>';
footerForBusUnitBranchHead +=    '<td align = "left" style="font-size:80%" font face="Verdana"><b>Kalan Tahsilat</b></td>';
footerForBusUnitBranchHead +=    '<td align = "left" style="font-size:80%" font face="Verdana"><b>Tahsilat oranı</b></td>';
footerForBusUnitBranchHead +=    '<td align = "left" style="font-size:80%" font face="Verdana"><b>Aktivite sayısı</b></td>';
footerForBusUnitBranchHead +=    '<td align = "left" style="font-size:80%" font face="Verdana"><b>Görev sayısı</b></td>';
footerForBusUnitBranchHead +=    '<td align = "left" style="font-size:80%" font face="Verdana"><b>Tamamlanan görev sayısı</b></td>';
footerForBusUnitBranchHead += '</tr>';


while select sum(ActualAmount),sum(AmounTarget),sum(DueDateOldPeriod),sum(DueDateCurrentPeriod)   from collectionLine //sum() fonksiyonu ile verilen filtreye göre
                                                                                                  //gelen değerlerin verilen alanlara göre toplanması
group by Bolge //Bölge bazlı sonuç elede etmek için gruplama yapıldı.
where collectionLine.Bolge && collectionLine.Sube //Bölge ve şube alanlarının dolu olması için filtre
&&    collectionLine.PeriodCode ==  dmrCollectionPeriod.PeriodCode //Verinin alınacağı
{
    select count(RecId) from collectionActivities //Bölge bazında aktivite sayısını bulmak için gerekli sorgu
    where collectionActivities.OMOperatingUnit == collectionLine.Bolge  //Bölge kodu filtresi
    &&    collectionActivities.PeriodCode      == dmrCollectionPeriod.PeriodCode;  //Dönem filtresi

    select count(RecId) from collectionTasks //Bölge bazında görev sayısını bulmak için gerekli sorgu
    where collectionTasks.OMOperatingUnit == collectionLine.Bolge //Bölge kodu filtresi
    &&    collectionTasks.PeriodCode      == dmrCollectionPeriod.PeriodCode; //Dönem filtresi

    select count(RecId) from collectionTasksCompleted //Bölge bazında tamamlanan görev sayısı bulmak için gerekli sorgu
    where collectionTasksCompleted.OMOperatingUnit == collectionLine.Bolge //Bölge kodu filtresi
    &&    collectionTasksCompleted.TaskStatus      == DmrTaskStatus::Completed //Görev durumu tamamlandı olan kayıtların gelmesi için filtre
    &&    collectionTasksCompleted.PeriodCode      == dmrCollectionPeriod.PeriodCode; //Dönem filtresi


     //Mailin mesaj kısmının doldurulması
     bodyForBusUnit  += '<tr>';
     bodyForBusUnit += strfmt('<td align = "left" style="font-size:70%" font face="Verdana"> %1 </td>'
     +'<td align = "left" style="font-size:70%" font face="Verdana"> %2 </td>'
     +'<td align = "left" style="font-size:70%" font face="Verdana"> %3 </td>'
     +'<td align = "left" style="font-size:70%" font face="Verdana"> %4 </td>'
     +'<td align = "left" style="font-size:70%" font face="Verdana"> %5 </td>'
     +'<td align = "left" style="font-size:70%" font face="Verdana"> %6 </td>'
     +'<td align = "left" style="font-size:70%" font face="Verdana"> %7 </td>'
     +'<td align = "left" style="font-size:70%" font face="Verdana"> %8 </td>'
     +'<td align = "left" style="font-size:70%" font face="Verdana"> %9 </td>'
     +'<td align = "right" style="font-size:90%"  font face="Verdana"> %10  </td> </tr>'
     , collectionLine.Bolge
     , collectionLine.DueDateOldPeriod
     , collectionLine.DueDateCurrentPeriod
     , collectionLine.AmounTarget
     , abs(collectionLine.ActualAmount)
     , abs(collectionLine.AmounTarget-abs(collectionLine.ActualAmount))
     , abs((abs(collectionLine.ActualAmount)/collectionLine.AmounTarget)*100)
     , collectionActivities.RecId
     , collectionTasks.RecId
     , collectionTasksCompleted.RecId);

     flag =true;
}

bodyGeneral += '</table></br>'; //Tablo görünümünün sağlanması
bodyForBusUnit += '</table></br>';//Tablo görünümünün sağlanması

allMailbody = bodyheadForBusUnit + bodyGeneralDetail+bodyGeneral+ bodyheadForBusUnitBolge +  bodyForBusUnit +footerForBusUnitBranchHead + footerForBusUnitBranchBody;
//allMailbody içerisine mailin içeriği aktarıldı.

return allMailbody;
```

Şekil 4.14. Mail mesajının oluşturulması.

Şekil 4.14 kod satırları ile mailin içeriğini oluşturmuş olduk. Mailin içeriğinde; "Türkiye geneli", "Bölge bazlı tahsilat hedefi" ve "Şube bazlı tahsilat hedefi" fonksiyonları olmak üzere üç ayrı detay kayıtları ayrı ayrı verilerek mailde belirtilmiştir.

Not: Mail'in "body" ve "footer" kısmının oluşturulmasında HTML (Hypertext Markup Language) kodlamadan faydalanılmıştır.

```
void sendMail(str _messageBody)
{
    SysEmailParameters      parameters = SysEmailParameters::find();
    SMTPRelayServerName     relayServer;
    SMTPPortNumber          portNumber;
    SMTPUserName            userName;
    SMTPPassword            password;
    Str1260                 subject,body;
    InteropPermission       interopPermission;
    SysMailer               mailer;
    System.Exception        e;
    str                     _toMail,_subject;
    Email                   _senderAddr;
    ;
    relayServer = "10.20.30.40";
    portNumber  = 40;
    userName    = "UserName";
    password    = "Password";
    _senderAddr ="SenderAddress";
    CodeAccessPermission::revertAssert();
    _toMail = "ecehankilabaz@gmail.com";//Gönderilecek mail adresi
    _subject = strFmt("%1 Dönemi Hedef Tahsilat Raporu",dmrCollectionPeriod.PeriodCode);// Mailin konusunun girildiği yer

    interopPermission = new InteropPermission(InteropKind::ComInterop);
    interopPermission.assert();
    mailer = new SysMailer();
    mailer.SMTPRelayServer(relayServer,portNumber,userName,password, parameters.NTLM);
    //instantiate email
    mailer.fromAddress(_senderAddr);    //Mailin iletileceği adres bilgisi
    mailer.tos().appendAddress(_toMail);//Mailin gönderileceği adres bilgisi
    mailer.subject(_subject);           //Mail konusunun bilgisi
    mailer.htmlBody(_messageBody);      //Mail içeriğinin bilgisi
    //mailer.attachments().add(_attachmentfileName); //Maile eklenecek bir dosya varsa buraya eklenir
    mailer.sendMail();
    CodeAccessPermission::revertAssert();
```

Şekil 4.15. Mailin gönderimi için oluşturulan metot.

Mail'in gönderimi için gerekli parametreler olan; "relayServer", "portNumber", "userName", "password", "_senderAddr", "_toMail", "_subject" , "messageBody" değerlerine ilgili atamalar yapılmıştır.

Kime: Ecehan Kılabaz

Sayın Yetkili,

13.9.2020 tarihi itibari iletahsilat hedefi raporunuz aşağıda 2020-04 dönemi için Türkiye geneli , şube ve bölge bazlı olarak belirtilmiştir.
Saygılarımızla,

TÜRKİYE GENELİ

Vadesi Geçmiş Alacak	Vadesi Bu Dönem Olan Tahsilat	Hedef Tahsilat	Gerçekleşen Tahsilat	Kalan Tahsilat	Tahsilat oranı	Aktivite sayısı	Görev sayısı	Tamamlanan görev sayısı
67.853.453,98	106.535.170,80	174.388.624,78	105.870.938,38	68.517.686,40	60,71	228	0	0

Bölge bazlı tahsilat hedefi

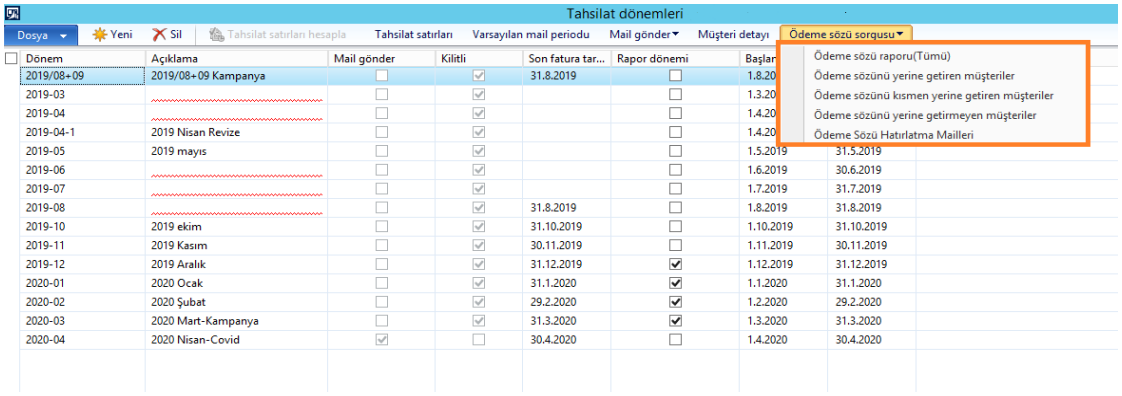
Bölge Kodu	Vadesi Geçmiş Alacak	Vadesi Bu Dönem Olan Tahsilat	Hedef Tahsilat	Gerçekleşen Tahsilat	Kalan Tahsilat	Tahsilat oranı	Aktivite sayısı	Görev sayısı	Tamamlanan görev sayısı
101000	0,00	5.887,15	5.887,15	5.887,15	0,00	100,00	0	0	0
101200	10.133,80	0,00	10.133,80	0,00	10.133,80	0,00	0	0	0
102000	10.427.657,00	10.958.803,59	21.386.460,59	10.187.390,41	11.199.070,18	47,63	20	0	0
103000	13.000.048,90	15.762.749,36	28.762.798,26	14.321.207,03	14.441.591,23	49,79	101	0	0
104000	7.653.803,51	16.717.973,72	24.371.777,23	17.105.249,57	7.266.527,66	70,18	16	0	0
105000	1.308.812,00	3.168.794,20	4.477.606,20	3.314.437,23	1.163.168,97	74,02	1	0	0
106000	11.757.125,69	18.420.720,04	30.177.845,73	15.188.259,98	14.989.585,75	50,33	12	0	0
107000	7.161.731,34	6.727.758,22	13.889.489,56	9.429.743,82	4.459.745,74	67,89	23	0	0

Şube Bazlı Tahsilat Hedefi

Bölge kodu	Şube Kodu	Vadesi Geçmiş Alacak	Vadesi Bu Dönem Olan Tahsilat	Hedef Tahsilat	Gerçekleşen Tahsilat	Kalan Tahsilat	Tahsilat oranı	Aktivite sayısı	Görev sayısı	Tamamlanan görev sayısı
101000	101001	0,00	5.887,15	5.887,15	5.887,15	0,00	100,00	0	0	0
101200	101202	8.202,00	0,00	8.202,00	0,00	8.202,00	0,00	0	0	0
101200	101203	1.931,80	0,00	1.931,80	0,00	1.931,80	0,00	0	0	0
102000	102150	101.246,56	758.797,16	860.043,72	761.256,05	98.787,67	88,51	1	0	0
102000	102151	509.688,36	874.366,16	1.384.054,52	899.085,54	484.968,98	64,96	3	0	0
102000	102152	734.061,90	513.897,45	1.247.959,35	474.611,83	773.347,52	38,03	0	0	0
102000	102153	545.519,43	912.829,38	1.458.348,81	814.056,52	644.292,29	55,82	0	0	0
102000	102154	1.431.709,93	615.931,33	2.047.641,26	599.982,11	1.447.659,15	29,30	11	0	0
102000	102155	3.343.821,69	3.783.102,70	7.126.924,39	3.690.085,91	3.436.838,48	51,78	0	0	0

Şekil 4.16. Mailin içeriği.

Şekil 4.16'da Gönderilen mail içeriğinin kullanıcı tarafından rapor detayı olarak nasıl görüntülendiği gösterilmektedir.



Şekil 4.17. Ödeme sözü sorgu raporları.

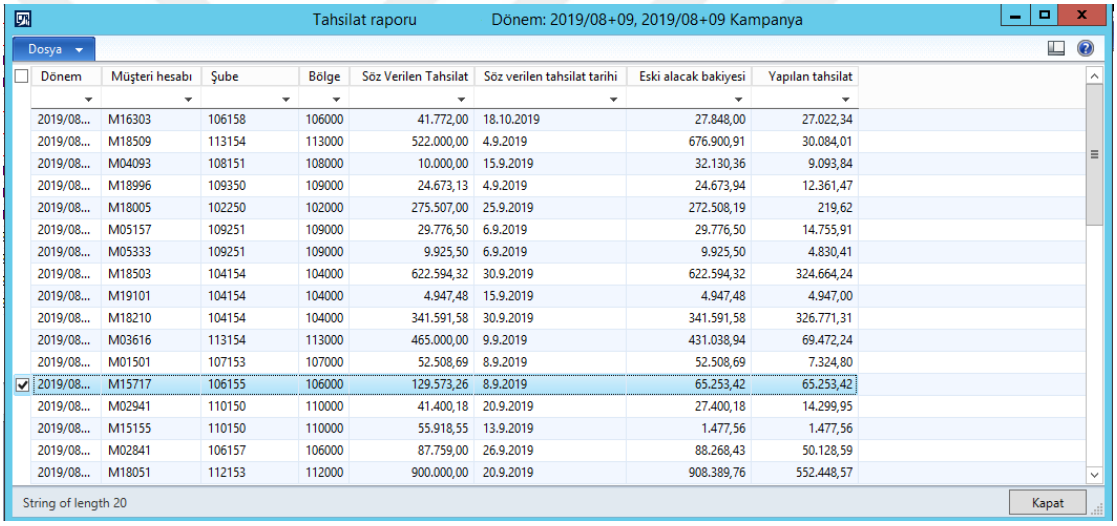
Şekil 4.17'de gösterilen "Ödeme sözü sorgusu" butonunun içeriğine baktığımızda karşımıza beş fonksiyon çıkmaktadır. Bu fonksiyonlar; "Ödeme sözü raporu(Tümü)", "Ödeme sözünü yerine getiren müşteriler" , "Ödeme sözünü kısmen yerine getiren müşteriler" , "Ödeme sözünü yerine getirmeyen müşteriler" ve "Ödeme sözü hatırlatma mailleri" olarak görüntülenmektedir.

Aşağıda örnek olarak "Ödeme sözünü kısmen yerine getiren müşteriler"" fonksiyonu ele alınarak incelenecektir.

```
public DmrCollectionPromiseTmp missingPayment()//Kısmi ödeme yapanlar
{


DmrCollectionPromiseTmp   tmp;
DmrCollectionActivities   dmrCollectionActivities;
DmrCustCollectionTable    dmrCustCollectionTable;


while select dmrCollectionActivities
where dmrCollectionActivities.PeriodCode == periodCode //üzerinde bulunduğumuz dönem filtresi
&&    dmrCollectionActivities.PromiseDate <= today()   //ödeme günü geçmemiş filtresi
&&    dmrCollectionActivities.IsDeptPromise == DmrNoYesBlank::Yes //Müşterinin tahsilat sözü var mı? filtresi
{
    select dmrCustCollectionTable
    where  dmrCustCollectionTable.PeriodCode        == periodCode //üzerinde bulunduğumuz dönem filtresi
    &&     dmrCustCollectionTable.BranchId          == dmrCollectionActivities.BranchId //Şube filtresi
    &&     dmrCustCollectionTable.OperatingUnit     == dmrCollectionActivities.OMOperatingUnit //Bölge filtresi
    &&     dmrCustCollectionTable.CustAccount       == dmrCollectionActivities.Customer; //Müşteri filtresi
    if((dmrCollectionActivities.DeptBeforePromise - (dmrCustCollectionTable.AmounTarget - dmrCustCollectionTable.PaymentAmount)) < dmrCollectionActivities.PaymentPromise
        && (dmrCollectionActivities.DeptBeforePromise - (dmrCustCollectionTable.AmounTarget - dmrCustCollectionTable.PaymentAmount))> 0 )
    {//ödemeye dair kontrollerin yapıldığı koşul

        //Forma gelmesi için tablonun doldurulduğu bölüm
        tmp.PaymentPromise      = dmrCollectionActivities.PaymentPromise;
        tmp.ReceivablePayment   = dmrCollectionActivities.DeptBeforePromise;
        tmp.DoneCollection      = dmrCollectionActivities.DeptBeforePromise - (dmrCustCollectionTable.AmounTarget - dmrCustCollectionTable.PaymentAmount);
        tmp.PeriodCode          = periodCode;
        tmp.BranchId            = dmrCollectionActivities.BranchId;
        tmp.OMOperatingUnit     = dmrCollectionActivities.OMOperatingUnit;
        tmp.Customer            = dmrCollectionActivities.Customer;
        tmp.PromiseDate         = dmrCollectionActivities.PromiseDate;
        tmp.Name                = dmrCustCollectionTable.Name;
        tmp.insert();

    }
}
return tmp;
```

Şekil 4.18. Ödeme sözünü kısmen yerine getiren müşteriler raporunun kod bloğu.

Şekil 4.18'de "dmrCollectionActivities" tablosu "while select" fonksiyonu kullanılarak "PeriodCode" alan değeri kullanıcının üzerinde bulunduğu dönem olmak üzere, "PromiseDate" alan değeri raporu çalıştırdığımız tarih ve "IsDeptPromise" alan değeri "Yes" olan kayıtları elde edilerek bulunmaktadır. Bu filtrelere uygun kayıtlar bulunmasına ek olarak dmrCustCollectionTable tablosundan da "PeriodCode", "BranchId", "OperatingUnit" ve "CustAccount" filtreleri kullanılarak ilgili tahsilat müşterisinin ödeme bilgisi elde edilmiştir. Bu sorgu sonucundaki kayıtlar bizim için ödeme sözünü kısmen yerine getiren müşterileri ve tutarları ifade etmektedir. Son olarak elde edilen kayıtların kullanıcı tarafından görüntülenebilmesi için "Tmp" tipindeki tabloya "insert()" fonksiyonu kullanılarak kayıtlar oluşturulmuştur.



Dönem	Müşteri hesabı	Şube	Bölge	Söz Verilen Tahsilat	Söz verilen tahsilat tarihi	Eski alacak bakiyesi	Yapılan tahsilat
2019/08...	M16303	106158	106000	41.772,00	18.10.2019	27.848,00	27.022,34
2019/08...	M18509	113154	113000	522.000,00	4.9.2019	676.900,91	30.084,01
2019/08...	M04093	108151	108000	10.000,00	15.9.2019	32.130,36	9.093,84
2019/08...	M18996	109350	109000	24.673,13	4.9.2019	24.673,94	12.361,47
2019/08...	M18005	102250	102000	275.507,00	25.9.2019	272.508,19	219,62
2019/08...	M05157	109251	109000	29.776,50	6.9.2019	29.776,50	14.755,91
2019/08...	M05333	109251	109000	9.925,50	6.9.2019	9.925,50	4.830,41
2019/08...	M18503	104154	104000	622.594,32	30.9.2019	622.594,32	324.664,24
2019/08...	M19101	104154	104000	4.947,48	15.9.2019	4.947,48	4.947,00
2019/08...	M18210	104154	104000	341.591,58	30.9.2019	341.591,58	326.771,31
2019/08...	M03616	113154	113000	465.000,00	9.9.2019	431.038,94	69.472,24
2019/08...	M01501	107153	107000	52.508,69	8.9.2019	52.508,69	7.324,80
2019/08...	M15717	106155	106000	129.573,26	8.9.2019	65.253,42	65.253,42
2019/08...	M02941	110150	110000	41.400,18	20.9.2019	27.400,18	14.299,95
2019/08...	M15155	110150	110000	55.918,55	13.9.2019	1.477,56	1.477,56
2019/08...	M02841	106157	106000	87.759,00	26.9.2019	88.268,43	50.128,59
2019/08...	M18051	112153	112000	900.000,00	20.9.2019	908.389,76	552.448,57

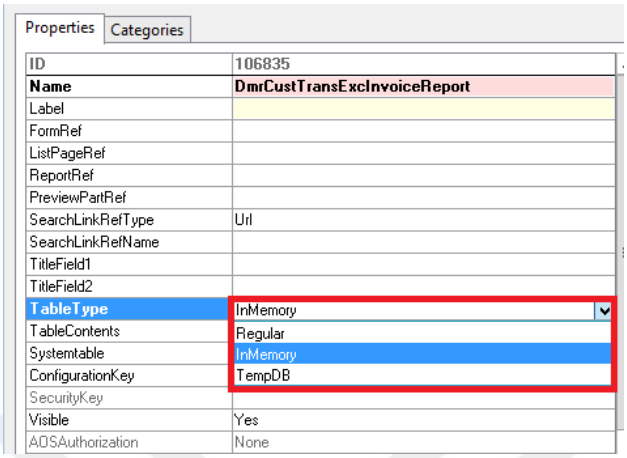
Şekil 4.19. Ödeme sözünü kısmen yerine getiren müşteriler formu.

Şekil 4.19'da gösterilen form'da Şekil 4.18'deki sorgu sonucu elde edilen kayıtların kullanıcının kontrol edeceği form üzerinde listelenerek görüntülenmesi sağlanmıştır.

4.2. Banka Bazında Müşteri Tahsilat Raporu Geliştirmesi

Kullanıcının müşteri cari kartlarına atılan banka ödeme kayıtlarının olduğu bir tahsilat formu talebi bulunmaktadır. İlgili tahsilat formunun belirtilen iki tarih arasındaki verileri getirebilecek özellikte olması istenmektedir.

Bunun için bilmemiz gereken birkaç tane nokta mevcut. Öncelikle tablonun özelliklerine değinerek başlayabiliriz.



Şekil 4.20. Tablo tipleri.

Regular: Bu tipteki tablolar içerisinde fiziksel veri tutan ve disk üzerinde belli bir boyutu olan kalıcı tablolardır.

TempDB: Temel SQL Server'ın TempDB veritabanında bulunan fiziksel veri ve disk üzerinde boyutu olmayan geçici tablolardır.

InMemory: InMemory tabloları, işlemin her katmanı üzerinde çalıştığı aktif bellekte, istemci veya sunucu katmanı olarak somutlaştırılır. InMemory tabloları asla veritabanı yönetim sisteminde temsil edilmez. Veri miktarı az olduğunda InMemory tipindeki geçici tabloları kullanılmalıdır. TempDP tablo tipini kullanmaktan kaçınarak, Microsoft SQL Server gidiş dönüşleri önlenmelidir.

InMemory tabloları	**TempDB tabloları**
1. Verileri geçici olarak istemci veya sunucu katmanında tutar	1. Kapsam geçerli olana kadar verileri geçici olarak veritabanında tutar
2. Bu tablolar Veritabanında saklanamaz	2. Bu tablolar veritabanında saklanır
3. Güvenlik uygulanmaz	3. Güvenlik uygulanabilir

Kullanıcının isteğine dönersek üç tablo tipini de kullanabiliriz. Fakat mevcut durum için en uygununu seçmemiz gerekecektir. Bunun için nedenleriyle hangi tipleri seçip seçemeyeceğimizden bahsedebiliriz.

"Regular" tipinde bir tablo kullanırsak her tabloya kayıt atışımızda önceki tablo verilerinin silinmesi gerekir. Aynı an da birden fazla kullanıcı raporu almak istediğinde en son raporu çalıştıran önceki rapor çalışmalarının silinmesine neden olacaktır. Buradaki veri karışıklığı ve tutarsızlığı konusu göz ardı edilmemeli ve istenmeyen sonuçlara neden olabileceği unutulmamalıdır.

Bu konu hakkında birkaç çözüm yolumuz olabilir. Örneğin; tablonun bir alanında kaydı oluşturan kişinin bilgisi tutulduğunda, sorgu yazılırken kaydı oluşturan kullanıcı bazında filtreleyerek silinmesi sağlanabilir. Bu şekilde rapor çalıştırıldığında her kullanıcının önceden oluşturduğu kayıtlarını sildirecek bir yapı üzerinden kurulumlarını sağlayabiliriz. Aynı yapıyı verilerin form üzerinde yazdırma işleminde de yapılması gerekecektir. Birden fazla kullanıcı kaydı attığında veri üzerinde çoklama meydana gelecektir. Kayıtların form üzerinde kullanıcı bazlı filtre verilerek herkesin kendi kaydını görmesi sağlanılmalıdır.

Sonuç olarak kullanıcının ilgili talebini "Regular" tablo tipinde gerçekleştirildiği takdirde yapılacak işlemlerin daha kontrollü ve dikkat edilmesi gereken kısımların fazlalığı dikkat çekmektedir.

InMemory veya TempDB tipinde bir tablo kullanıldığında "Regular" tablo için yazılan herhangi bir işlemin gerçekleştirilmesine gerek kalmayacaktır. Nedeni ise; bu tipteki tabloların geçici tablolar olduğundan bahsedip fiziksel olarak bir veri içeriğini tutmadığını söyleyebiliriz. Geçici tablo yapısını detaylı inceleyecek ve tanımlamak istersek en basit anlatımı ile; veri kümesini belli süre ön belleğinde tutarak form üzerinde geçici bir gösterim sağlaması olarak nitelendirebiliriz. Bu gösterimin bitiminde kullanıcının formu kapatma işlemi sonucu bu veri kümesinin silinmesi işlemidir olarak bahsetmek mümkündür.

Peki kullanıcının talebine karşılık "Regular" tablo yapmamamız gerektiğine karar verildiğinde "InMemory" mi yoksa "TempDB" mi tablo tipleri kullanılmalıdır sorusuna cevap arayalım. Burada ne tür bir veri boyutu ile çalışıldığının bilgisini biliyor olunması gerekecektir. Eğer büyük bir veri kümesi üzerinde çalışılıyor ise "InMemory" tablo tipi

kullanılması Microsoft SQL Server üzerindeki işlem yoğunluğunu azaltacağından performans açısından daha verimli olacağı düşünülmelidir. Ortalama bir veri kümesi üzerinde çalışılıyor ise "TempDB" tablo tipi de kullanılabilir.

```
public DmrCustTransExcInvoiceReport calculate()
{
    DmrCustTransExcInvoiceReport   tmp;
    CustTrans                      custTrans;
    DirPartyTable                  dirpartyTable;
    CustTable                      custTable;
    BankAccountTable               bankAccountTable;
    utcDateTime                    startTime,endTime;

    startTime = DateTimeUtil::utcNow();
    insert_recordset tmp(Accountnum,AmountCur,AmountMST,CurrencyCode,ExchRate,TransDate,Voucher,CompanyBankAccountId)
    select Accountnum,AmountCur,AmountMST,CurrencyCode,ExchRate,TransDate,Voucher,CompanyBankAccountId
    from  custTrans
    where custTrans.TransDate >= StartDate.dateValue()
    &&    custTrans.TransDate <= EndDate.dateValue()
    &&    !custTrans.Invoice;

    update_recordSet tmp
    setting name = dirpartyTable.Name
    join  custTable
    where custTable.AccountNum == tmp.AccountNum
    join  name from dirpartyTable
    where dirpartyTable.RecId == custTable.Party;

    update_recordSet tmp
    setting ExchRate = tmp.ExchRate /100;

    update_recordSet tmp
    setting BankName = bankAccountTable.Name
    join bankAccountTable
    where bankAccountTable.AccountID == tmp.CompanyBankAccountId;

    endTime = DateTimeUtil::utcNow();
    info(strfmt("Bitiş zamanı: %1 - Başlangıç zamanı: %2 - Zaman farkı: %3",endTime ,startTime,DateTimeUtil::time(endTime)- DateTimeUtil::time(startTime) ));
    return tmp;
```

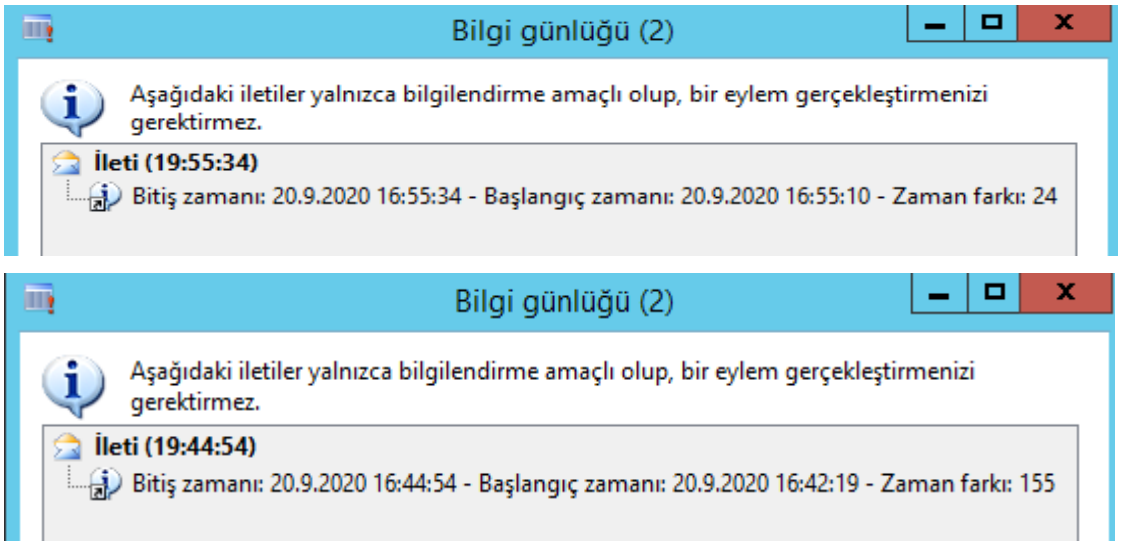
Şekil 4.21. Insert_recordSet ve update_recordSet fonksiyonları ile veri oluşturulması.

Şekil 4.21'deki kod bloğu "insert_recordSet" ve "update_recordSet" fonksiyonları üzerinden yazılmıştır. Bu fonksiyonlar "InMemory" tablo tipindeki bir tabloya kullanıcının istediği filtreler kapsamında veri kümesinin toplanması ve oluşturularak form üzerinde görüntülenmesi işlemini kapsamaktadır.

```
public DmrCustTransExcInvoiceReport calculate2()
{
    DmrCustTransExcInvoiceReport   tmp;
    CustTrans                      custTrans;
    DirPartyTable                  dirpartyTable;
    CustTable                      custTable;
    BankAccountTable               bankAccountTable;
    utcDateTime                     startTime,endTime;

    startTime = DateTimeUtil::utcNow();
    while select custTrans
    where custTrans.TransDate >= StartDate.dateValue()
    &&    custTrans.TransDate <= EndDate.dateValue()
    &&    !custTrans.Invoice
    {
        tmp.Accountnum              = custTrans.Accountnum;
        tmp.AmountCur               = custTrans.AmountCur;
        tmp.AmountMST               = custTrans.AmountMST;
        tmp.CurrencyCode            = custTrans.CurrencyCode;
        tmp.ExchRate                = custTrans.ExchRate /100;
        tmp.TransDate               = custTrans.TransDate;
        tmp.Voucher                 = custTrans.Voucher;
        tmp.CompanyBankAccountId    = custTrans.CompanyBankAccountId;

        select bankAccountTable
        where bankAccountTable.AccountID == tmp.CompanyBankAccountId;
        tmp.BankName                = bankAccountTable.Name;
        select custTable
        where custTable.AccountNum == tmp.AccountNum
        join  name from dirpartyTable
        where dirpartyTable.RecId == custTable.Party;
        tmp.Name                    = dirpartyTable.Name;
        tmp.insert();
    }
     endTime = DateTimeUtil::utcNow();
     info(strfmt("Bitiş zamanı: %1 - Başlangıç zamanı: %2 - Zaman farkı: %3",endTime ,startTime,DateTimeUtil::time(endTime)- DateTimeUtil::time(startTime) ));
     return tmp;
```

Şekil 4.22. While select fonksiyonu ile veri oluşturulması.

Şekil 4.22'deki kod bloğu "while select" fonksiyonu ve "Insert()" metodu kullanılarak yazılmıştır. Bu fonksiyon "InMemory" tablo tipindeki bir tabloya kullanıcının istediği filtreler kapsamında veri kümesinin toplanması ve oluşturularak form üzerinde görüntülenmesi işlemini kapsamaktadır.



Şekil 4.23. Performans karşılaştırması.

Şekil 4.21'de kullanılan "insert_recordSet" ve "update_recordSet" fonksiyonlarının kullanımı sonrası Şekil 4.23'de bir performans çıktısı üzerinde raporun

başlangıcından sonuna kadar toplam 24 saniyelik bir çalışması söz konusu olduğu görülmektedir. Şekil 4.22'de kullanılan "while selec"t fonksiyonunun kullanımı sonrası Şekil 4.23'e bakıldığında 155 saniyelik bir çalışması olduğu söylenebilir. Bu çalışma sürelerinin kullanıcının butona bastıktan sonra raporu alana kadar ki geçen süre olarak düşünüldüğünde yaklaşık performans kaybı "while select" fonksiyonu kullanıldığında 6 kat daha fazla uzadığı görüntülenmektedir. Bu süreler her ne kadar kısa olsa da veri kümesinin büyük olduğu "big data" tablolar üzerinde çalışıldığında performans kaybının ne kadar artacağı unutulmamalıdır. Her ne kadar bu tip sorgular performans kaybı olarak düşünülse de bir diğer kısım ise kullanıcının çalışırken ki verimini de olumsuz etkilediği göz ardı edilmemelidir.

File New Sil

Başlangıç Tarihi: 1.1.2019

Bitiş Tarihi: 20.9.2020

Hesapla

Müşteri hesabı	Para birimi tutarı	Tutar	Para bi...	Banka hesabı	Döviz kuru	Tarih	Fiş
12001-00001	50,00	50,00	TRY	1000100001	1,000000000000	28.2.2020	YNP-000002095
12001-00002	1,00	1,00	TRY	1080101016	1,000000000000	16.12.2019	KKT-000002969
12001-00002	1,00	1,00	TRY	1080101016	1,000000000000	16.12.2019	KKT-000002970
12001-00002	1,00	1,00	TRY	1080101016	1,000000000000	16.12.2019	KKT-000002972
12001-00002	1,00	1,00	TRY	1080101016	1,000000000000	16.12.2019	KKT-000002974
12001-00002	1,00	1,00	TRY	1080101016	1,000000000000	16.12.2019	KKT-000002975
12001-00556	4.000,00	4.000,00	TRY	1080101013	1,000000000000	19.6.2019	KKT-000002965
12001-01441	6.000,00	6.000,00	TRY	1080102021	1,000000000000	19.6.2019	KKT-000002964
12001-01593	2.000,00	2.000,00	TRY	1080102023	1,000000000000	19.6.2019	KKT-000002962
12001-05094	525,00	525,00	TRY	1080103024	1,000000000000	19.6.2019	KKT-000002960
12001-05359	6.000,00	6.000,00	TRY	1080102023	1,000000000000	19.6.2019	KKT-000002961
12001-00080	2.553,00	2.553,00	TRY	1020104001	1,000000000000	18.6.2019	YNP-000002090
12001-00451	7.706,00	7.706,00	TRY	1080102023	1,000000000000	18.6.2019	KKT-000002958
12001-00736	614,00	614,00	TRY	1210100002	1,000000000000	18.6.2019	CEQ-000001785
12001-00736	794,00	794,00	TRY	1020104001	1,000000000000	18.6.2019	YNP-000002089
12001-01081	2.096,00	2.096,00	TRY	1020101001	1,000000000000	18.6.2019	YNP-000002091
12001-01439	490,00	490,00	TRY	1080101013	1,000000000000	18.6.2019	KKT-000002957
12801-00481	350,00	350,00	TRY	1020102001	1,000000000000	18.6.2019	YNP-000002093
12801-00495	1.985,00	1.985,00	TRY	1020102001	1,000000000000	18.6.2019	YNP-000002092
12001-00062	4.122,00	4.122,00	TRY	1080102023	1,000000000000	17.6.2019	KKT-000002949
12001-00095	700,00	700,00	TRY	1020104001	1,000000000000	17.6.2019	YNP-000002085
12001-00129	10.000,00	10.000,00	TRY	1020104001	1,000000000000	17.6.2019	YNP-000002086
12001-00451	7.000,00	7.000,00	TRY	1080104015	1,000000000000	17.6.2019	KKT-000002950
12001-00818	10.000,00	10.000,00	TRY	1210100001	1,000000000000	17.6.2019	CEQ-000001782
12001-00818	10.000,00	10.000,00	TRY	1210100001	1,000000000000	17.6.2019	CEQ-000001782
12001-00818	10.000,00	10.000,00	TRY	1210100001	1,000000000000	17.6.2019	CEQ-000001782

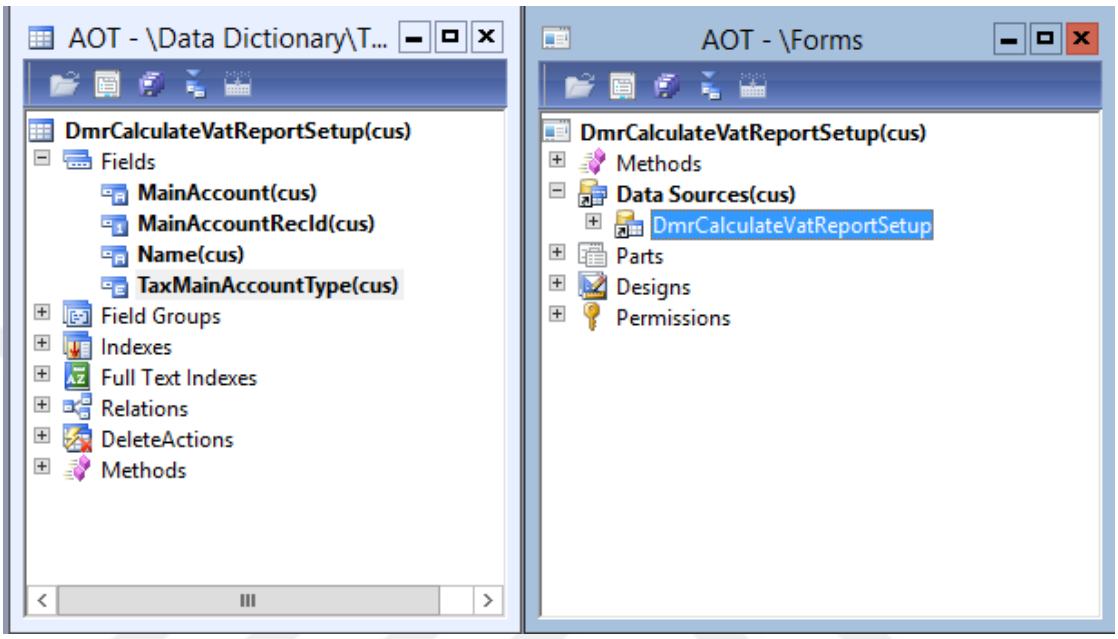
Şekil 4.24. Banka bazında müşteri tahsilat formu.

Şekil 4.24 üzerinde; Şekil 4.21 ve Şekil 4.22'de belirtilen farklı sorgu tipleri ile oluşturulan veri kümelerinin, kullanıcının ekranına form olarak raporlanmasıdır.

4.3. Dönem Sonu KDV (Katma Değer Vergisi) Tahakkuk İşlemleri

Kullanıcıların dönem sonlarında devlete beyan ettikleri ödenecek veya indirilecek KDV tutarlarının Dynamics Ax üzerinde muhasebe kayıtlarının atılması ve beyannamelerinde belirtmesi gerekmektedir. Bu işlem zaman alıcı ve mesai saatleri

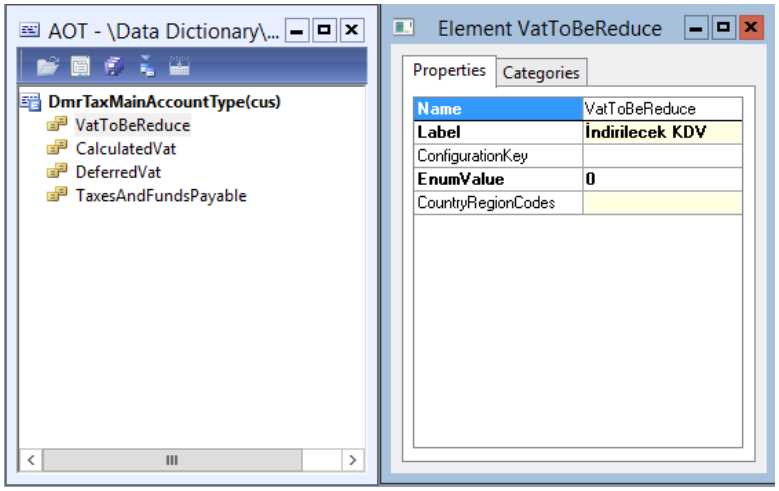
içerisinde kullanıcıların verimliliklerini düşüren bir işlem olarak görülmektedir. Bu KDV tahakkuku her ayın sonunda yapılan bir işlem olduğundan otomatikleşmesi, hesaplanması ve kullanıcı hatalarını engellemek amacıyla bir yapı tasarlanması gerekmektedir.



Şekil 4.25. Ana hesap kurulumu için tablo ve formlarının oluşturulması.

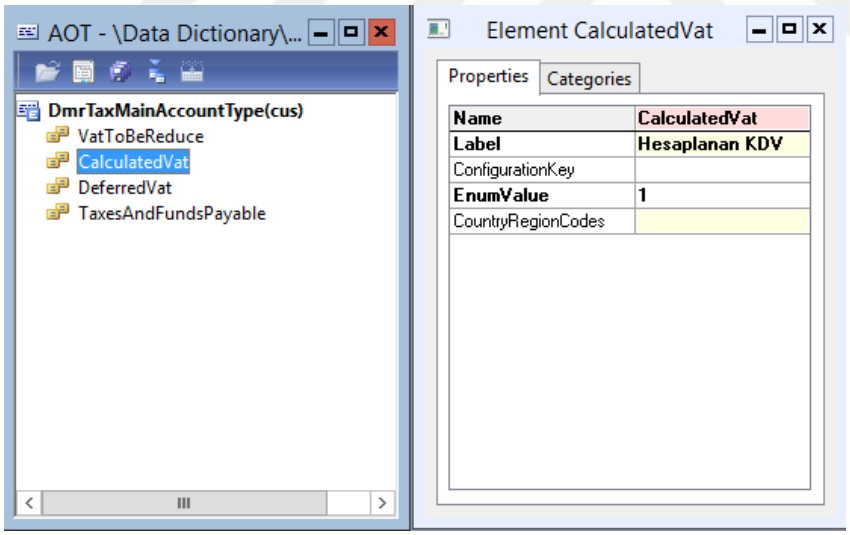
KDV tahakkuk işlemleri için, başlangıç olarak kullanıcıların tanımlaması gereken bir ana hesap tablosu ve formu oluşturulması gerekecektir. Bu tablo geliştirmenin ileriki aşamaları için "masterData" kabul edilerek işlemlere tabii tutulacaktır.

Oluşturulan "DmrCalculateVatReportSetup" tablosunun "TaxMainAccountType" alan değeri "enum" tipinde oluşturulmuştur. "Enum" değerleri aşağıdaki gibidir;



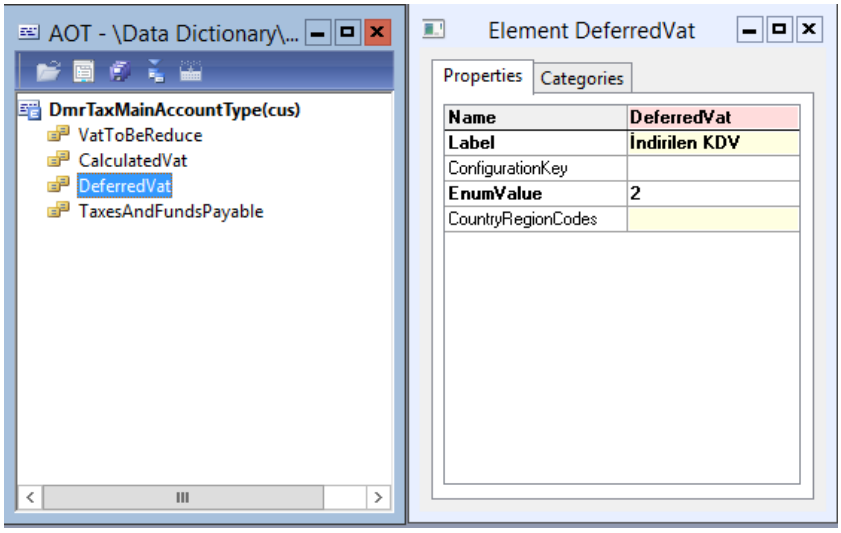
Şekil 4.26. VatToBeReduce Enum değeri.

Şekil 4.26'da kurulum tablosunda seçilebilecek değerlerden birisi "label" değeri "İndirilecek KDV" ve "enumValue" değeri "0(sıfır)" olan "VatToBeReduce" değeri görüntülenmektedir.



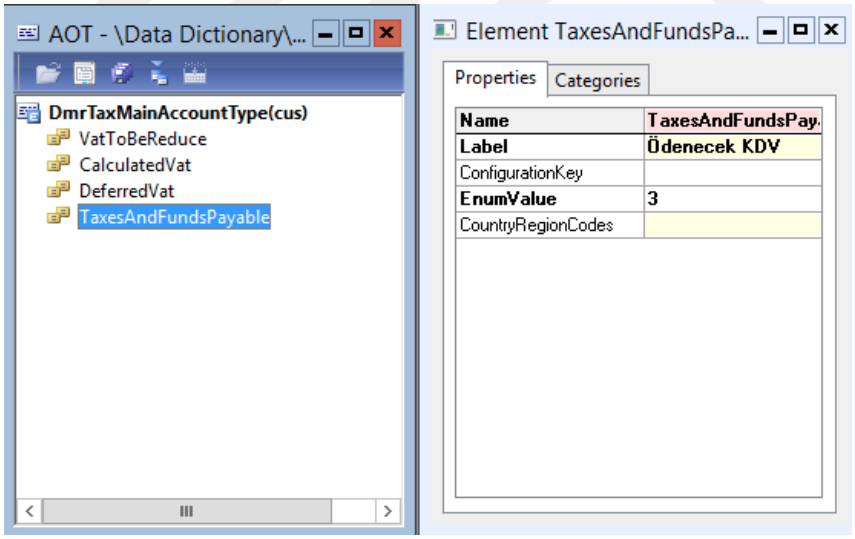
Şekil 4.27. CalculatedVat Enum değeri.

Şekil 4.27'de kurulum tablosunda seçilebilecek değerlerden diğeri "label" değeri "Hesaplanan KDV" ve "enumValue" değeri "1(bir)" olan "CalculatedVat" değeri görüntülenmektedir.



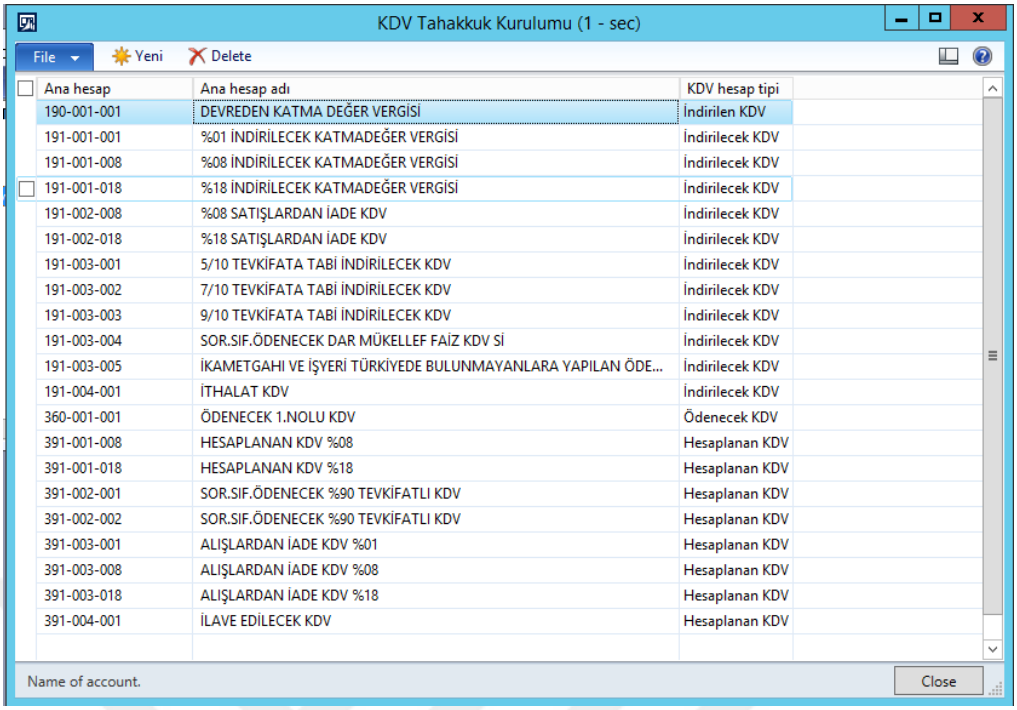
Şekil 4.28. DeferredVat Enum değeri.

Şekil 4.28'de kurulum tablosunda seçilebilecek değerlerden diğeri "label" değeri "İndirilen KDV" ve "enumValue" değeri "2(iki)" olan "DeferredVat" değeri görüntülenmektedir.



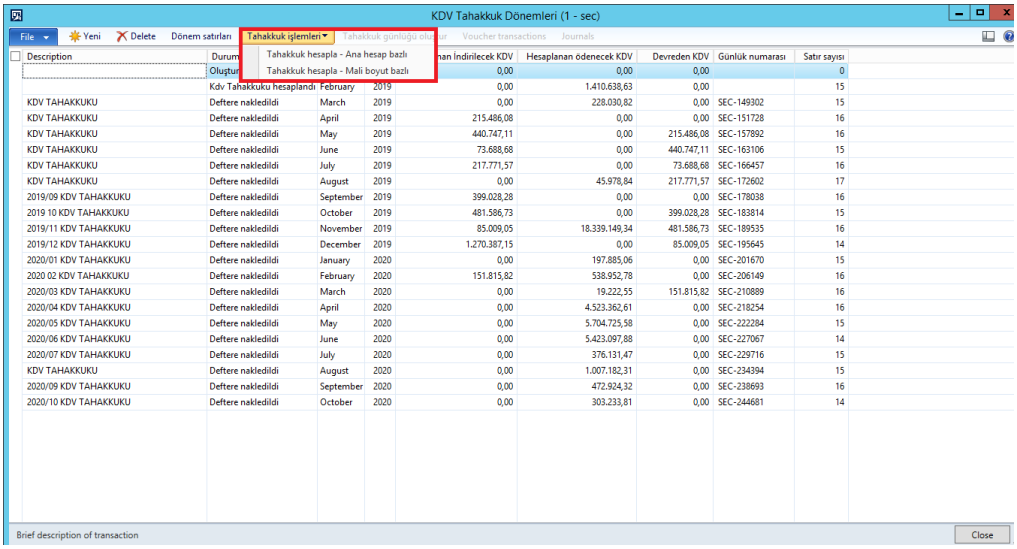
Şekil 4.29. TaxesAndFundsPayable Enum değeri.

Şekil 4.29'da kurulum tablosunda seçilebilecek değerlerden diğeri "label" değeri "Ödenecek KDV" ve "enumValue" değeri "3(üç)" olan "TaxesAndFundsPayable" değeri görüntülenmektedir.



Şekil 4.30. KDV tahakkkuk kurulumu formu.

Şekil 4.30 üzerinde her bir KDV ana hesabına karşılık kullanıcılar KDV hesap tipi tanımlaması beklenmektedir. Ayrıca enum değerleri olan; "İndirilecek kdv", "Hesaplanan kdv", "Ödenecek kdv" ve "İndirilen kdv" değerlerini "Kdv hesap tipi" alanında tanımladığı formun görüntüsü paylaşılmıştır.



Şekil 4.31. KDV tahakkuk dönemleri formu.

Tahakkuk hesaplama butonunun altında iki türde çalıştırılabilen bir yapı bulunmaktadır. Aşağıda bu yapılardan "Tahakkuk hesapla – Ana hesap bazlı" olan kısımdan bahsedilecektir.

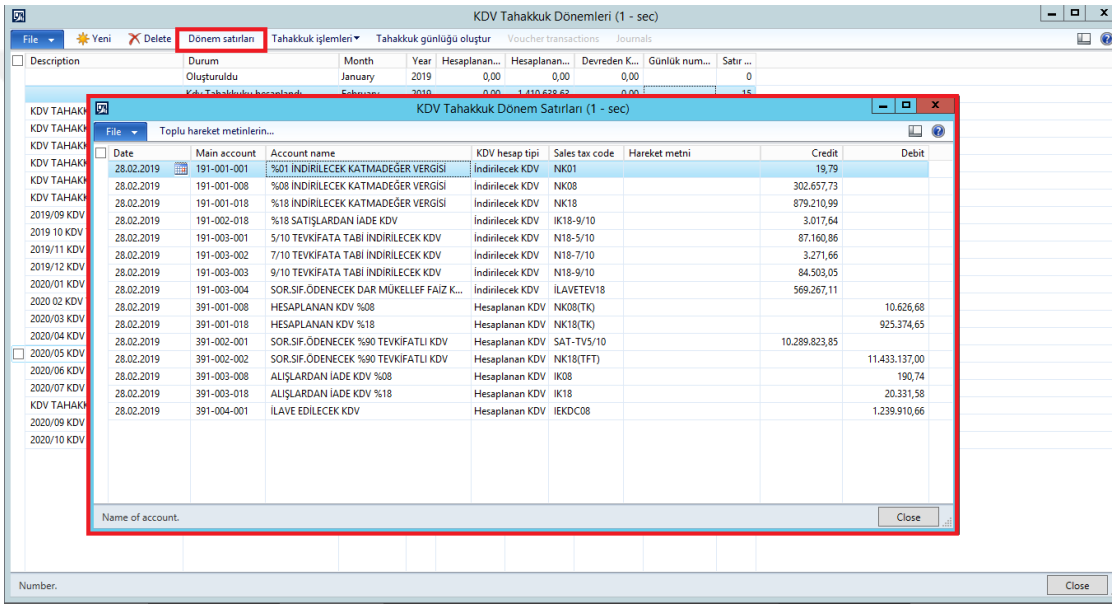
Tahakkuk hesapla – Ana hesap bazlı; KDV Tahakkuk kurulumu tablosunda belirtilen ana hesap ve rapor çekilmesi istenen tarih aralığında hesaplar üzerinde bulunan bakiyeler üzerinden çalışmaktadır. Bu bakiyeler günün sonunda kullanıcıların karşısına muhasebe kayıtlarını oluşturacak ve devlete verilecek beyannameler için referans olarak kullanılacaktır.

```
public void CalculatePeriodVatMainAccount()
{
    DmrCalculateVatReportLine           dmrCalculateVatReportLine,dmrCalculateVatReportLineNew;
    DmrCalculateVatReportSetup          dmrCalculateVatReportSetup;
    GeneralJournalEntry                 generalJournalEntry;
    GeneralJournalAccountEntry          generalJournalAccountEntry;
    DmrCalculateVatReportTmp            tmp;
    GeneralJournalSubledgerVoucher      GeneralJournalSubledgerVoucher;
    MainAccount                         mainaccount;
     DmrCalculateVatReportHeader        DmrCalculateVatReportHeaderBefore;
    DimensionAttributeValueCombination  DimensionAttributeValueCombination;
    boolean                             flag = false;
    RefRecId                            mainAccountId;
    int                                 mnth,yr,day;
    StartDate                           startDate;

    ttsBegin;
    if(box::yesno('Satırlar oluşturulsun mu?', DialogButton::No) == DialogButton::Yes)
    {
        delete_from dmrCalculateVatReportLine
        where  dmrCalculateVatReportLine.CalculateVatReportHeader == DmrCalculateVatReportHeader.RecId;

        mnth = mthOfYr(DmrCalculateVatReportHeader.StartDate);
        yr   = year(DmrCalculateVatReportHeader.StartDate);
        day  = dayOfMth(DmrCalculateVatReportHeader.StartDate);
        startDate = element.startDateCalc(day,mnth,yr);

        while select dmrCalculateVatReportSetup
        where (dmrCalculateVatReportSetup.TaxMainAccountType == DmrTaxMainAccountType::CalculatedVat
        || dmrCalculateVatReportSetup.TaxMainAccountType == DmrTaxMainAccountType::VatToBeReduce)
        {
            while select sum(AccountingCurrencyAmount) from GeneralJournalSubledgerVoucher
            group by MainAccountRecId
            where  GeneralJournalSubledgerVoucher.AccountingDate >= startDate
            &&     GeneralJournalSubledgerVoucher.AccountingDate <= DmrCalculateVatReportHeader.EndDate
            &&     GeneralJournalSubledgerVoucher.SubledgerVoucherDataAreaId == curext()
            &&    GeneralJournalSubledgerVoucher.MainAccountRecId == dmrCalculateVatReportSetup.MainAccountRecId
            {
              if(GeneralJournalSubledgerVoucher.AccountingCurrencyAmount !=0)
              {
                  tmp.MainAccount             = dmrCalculateVatReportSetup.MainAccount;
                  tmp.AccountingCurrencyAmount = GeneralJournalSubledgerVoucher.AccountingCurrencyAmount;
                  tmp.TaxMainAccountType       = dmrCalculateVatReportSetup.TaxMainAccountType;
                  tmp.LedgerDimension          = DimensionDefaultingService::serviceCreateLedgerDimension(
                                                          DimensionStorage::getDefaultAccount(dmrCalculateVatReportSetup.MainAccountRecId),0);

                  tmp.insert();
                  flag = true;
              }
            }
       }
       if(flag)
       {
           while select sum(AccountingCurrencyAmount) from tmp
           group by TaxMainAccountType,MainAccount,LedgerDimension
           {
               if(tmp.AccountingCurrencyAmount !=0)
               {
                   dmrCalculateVatReportLine.clear();
                   dmrCalculateVatReportLine.CalculateVatReportHeader = DmrCalculateVatReportHeader.RecId;
                   dmrCalculateVatReportLine.Date                     = DmrCalculateVatReportHeader.EndDate;
                   dmrCalculateVatReportLine.MainAccount              = tmp.MainAccount;
                   if(tmp.AccountingCurrencyAmount > 0)
                   {
                       dmrCalculateVatReportLine.AmountCurCredit = abs(tmp.AccountingCurrencyAmount);
                   }
                   else
                   {
                       dmrCalculateVatReportLine.AmountCurDebit = abs(tmp.AccountingCurrencyAmount);
                   }
                   dmrCalculateVatReportLine.TaxMainAccountType = tmp.TaxMainAccountType;
                   dmrCalculateVatReportLine.TaxCode  = element.getTaxCodeMainAccount(tmp.MainAccount);
                   dmrCalculateVatReportLine.ledgerdimension = tmp.LedgerDimension;
                   dmrCalculateVatReportLine.Name           = MainAccount::findByMainAccountId( dmrCalculateVatReportLine.MainAccount).Name;
                   dmrCalculateVatReportLine.insert();
               }
           }
    }
```

Şekil 4.32. Tahakkuk hesapla – ana hesap bazlı fonksiyonu kod bloğu.

Şekil 4.32'de verilen kod bloğu ile kullanıcının ekranına bir "messagebox" çıkarttırılarak "satırlar oluşturulsun mu?" seçeneği ile işleme devam edilip edilmeyeceği bilgisi istenmektedir. Eğer seçimi "Yes" ise işlemlere devam edilmektedir. "Delete_from" fonksiyonu kullanılarak başlığa ait satırlar silinmiştir. "mthofYear","year","dayOfMth" fonsiyonları ile başlıkta bulunan "StartDate" alanından yeni bir tarih oluşturulmuştur. Şekil 4.30'daki kurulum tablosuna sorgu çekilerek "Kdv hesap tipi" alanı; "İndirilecek kdv" ve "Hesaplanan kdv" olan kayıtlar

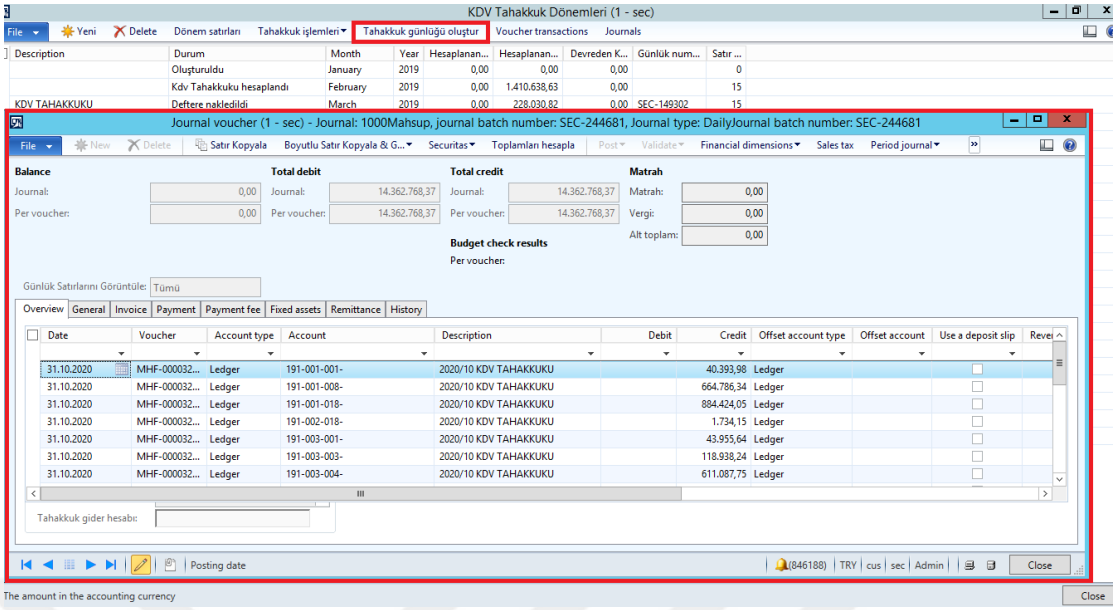
bulunmuştur. Kurulum tablosundan elde edilen ana hesaplar ile muhasebe kayıtları tablosuna gidilmiş ve filtre olarak “curext()” fonksiyonu verilerek mevcut şirket hesabındaki verilerin getirilmesi sağlanmıştır. Elde edilen kayıtlar özet bir rapor alabilmek adına kendi içinde gruplanması gerekmektedir. Verilerin gruplanması işlemi için elde edilen kayıtlar bir “tmp” tablo doldurularak işlem sonuna kadar saklanmaktadır. Bu tmp tabloyu KDV tipi ve ana hesap bazında “group by” fonksiyonu kullanarak gruplama işlemi gerçekleştirilmiştir. Gerçekleşen işlemler sonunda başlığımıza ait dönem satırları oluşmaktadır.



Şekil 4.33. KDV tahakkuk dönem satırları formu.

KDV Tahakkuk Dönemleri formu üzerinden basılan “Dönem satırları” butonu ile Şekil 4.33’deki KDV Tahakkuk Dönem satırları formu açılmaktadır. Bu form ile kullanıcıların daha önce KDV hesaplarını ve tiplerini tanımladıkları tablodaki kayıtlar özelinde ilgili aya ait muhasebe kayıtlarının bakiyeleri getirilmiştir.

Bu form üzerindeki bakiyeler her ne kadar bir sorgu üzerinden getirilse de, form üzerindeki “Alacak” ve “Borç” alan değerleri kullanıcılar tarafından “AllowEdit” fonksiyonuna açık olarak bırakılmıştır. Ek olarak form üzerinde bulunan “Toplu hareket metni güncelle” butonu ile satırlardaki açıklama alanları kullanıcının karşısına “messagebox” açılarak tek seferde güncelleyebilmesine olanak sağlamaktadır.



Şekil 4.34. KDV muhasebe günlükleri formu.

Şekil 4.34'de gösterilen KDV Tahakkuk Dönemleri formu üzerinden "Tahakkuk günlüğü oluştur" butonuna basılarak kullanıcıların kontrol etmesi için muhasebe kaydını oluşturmaktadır. Bu aşamada devlete beyan edilen KDV tutarlarında "Ödenecek KDV" bakiyesi oluşur ise ödemeler banka üzerinden yapılmalıdır. Eğer "İndirilen KDV" bakiyesi oluşur ise ilgili KDV tutarı bir sonraki aya devredilmektedir. Son kontroller ile muhasebenin kayıt işlemleri tamamlanmaktadır.

Aşağıdaki kod bloğu ile muhasebe kaydının başlığı ve satırları oluşturulmaktadır. Muhasebe kaydının satır bilgileri kısmına daha önce oluşturulan (Şekil 4.33 üzerindeki) ana hesap ve bakiyelerin satırlara ataması yapılmaktadır.

```
public void createJournalNum()
{
    DmrCalculateVatReportLine   dmrCalculateVatReportLine;
    LedgerParameters            ledgerParameters = LedgerParameters::find();
    LedgerJournalTable          ledgerJournalTableIns;
    LedgerJournalTrans          trans;
    LedgerJournalName           ledgerJournalName;
    DimensionAttributeValueCombination  DimensionAttributeValueCombination;
    LedgerJournalCheckPost      ledgerJournalCheckPost;
    SysInfoAction_formRun       infoAction;
    Ledger                      ledger;
    DmrCalculateVatReportSetup  dmrCalculateVatReportSetup;
    numberSequenceTable         numberSequenceTable;
    LedgerJournalTransTxt       txt;
    Voucher                     Voucher;

    ttsBegin;

    select ledgerJournalName
    where ledgerJournalName.JournalName == ledgerParameters.DmrVATReportJournlaId;

    numberSequenceTable = NumberSequenceTable::find(ledgerJournalName.NumberSequenceTable);
    Voucher             = NumberSeq::newGetVoucherFromCode(numberSequenceTable.NumberSequence).voucher();

    ledgerJournalTableIns.clear();
    ledgerJournalTableIns.initValue();
    ledgerJournalTableIns.JournalNum       = JournalTableData::newTable(ledgerJournalTableIns).nextJournalId();
    ledgerJournalTableIns.journalType      = LedgerJournalType::Daily;
    ledgerJournalTableIns.JournalName      = ledgerJournalName.JournalName;
    ledgerJournalTableIns.DmrLedgerCategoryTable = ledgerJournalName.DmrLedgerCategoryTable;
    ledgerJournalTableIns.Name             = DmrCalculateVatReportHeader.Description;
    ledgerJournalTableIns.initFromLedgerJournalName(ledgerJournalName.JournalName);

    ledgerJournalTableIns.insert();

    DmrCalculateVatReportHeader.selectForUpdate(true);
    DmrCalculateVatReportHeader.JournalNum = ledgerJournalTableIns.JournalNum;
    DmrCalculateVatReportHeader.VatReportActionType = DmrVatReportActionType::CreateJournalNum;
    DmrCalculateVatReportHeader.update();

   while select dmrCalculateVatReportLine
        where dmrCalculateVatReportLine.CalculateVatReportHeader == DmrCalculateVatReportHeader.RecId
   {
        trans.clear();
        trans.JournalNum            = ledgerJournalTableIns.JournalNum;
        trans.TransDate             = dmrCalculateVatReportLine.Date;
        trans.AccountType           = LedgerJournalACType::Ledger;
        trans.OffsetAccountType     = LedgerJournalACType::Ledger;
        trans.LedgerDimension       = dmrCalculateVatReportLine.LedgerDimension;
        trans.Txt                   = dmrCalculateVatReportLine.Txt;
        txt                         = dmrCalculateVatReportLine.Txt;
        trans.TaxCode               = dmrCalculateVatReportLine.TaxCode;
        trans.AmountCurCredit       = dmrCalculateVatReportLine.AmountCurCredit;
        trans.AmountCurDebit        = dmrCalculateVatReportLine.AmountCurDebit;
        trans.Voucher               = Voucher;
        select ledger
        where  ledger.Name == curext();
        trans.CurrencyCode    = ledger.AccountingCurrency;
        trans.ExchRate        = 100;
        trans.insert();
   }
   if(DmrCalculateVatReportHeader.CalculatedDeferredVat !=0)
   {
        select firstonly dmrCalculateVatReportSetup
        where  dmrCalculateVatReportSetup.TaxMainAccountType == DmrTaxMainAccountType::DeferredVat;

        trans.clear();
        trans.JournalNum        = ledgerJournalTableIns.JournalNum;
        trans.TransDate         = DmrCalculateVatReportHeader.EndDate;
        trans.LedgerDimension   = DimensionDefaultingService::serviceCreateLedgerDimension(DimensionStorage::getDefaultAccount(dmrCalculateVatReportSetup.MainAccountRecId),0);
        trans.Voucher           = Voucher;
        trans.Txt               = txt;
        trans.TaxCode           = element.getTaxCode(trans.LedgerDimension);
        trans.CurrencyCode      = ledger.AccountingCurrency;
        trans.AmountCurDebit    = DmrCalculateVatReportHeader.CalculatedDeferredVat;
        trans.ExchRate          = 100;
        trans.AccountType       = LedgerJournalACType::Ledger;
        trans.OffsetAccountType = LedgerJournalACType::Ledger;
        trans.insert();

   else if(DmrCalculateVatReportHeader.CalculatedDeferredVat == 0 &&  DmrCalculateVatReportHeader.AsigNorKDV == 0)
   {
       select firstonly dmrCalculateVatReportSetup
       where  dmrCalculateVatReportSetup.TaxMainAccountType == DmrTaxMainAccountType::TaxesAndFundsPayable;

       trans.clear();
       trans.JournalNum        = ledgerJournalTableIns.JournalNum;
       trans.TransDate         = DmrCalculateVatReportHeader.EndDate;
       trans.LedgerDimension   = DimensionDefaultingService::serviceCreateLedgerDimension(DimensionStorage::getDefaultAccount(dmrCalculateVatReportSetup.MainAccountRecId),0);
       trans.Voucher           = Voucher;
       trans.Txt               = txt;
       trans.TaxCode           = element.getTaxCode(trans.LedgerDimension);
       trans.CurrencyCode      = ledger.AccountingCurrency;
       trans.AmountCurDebit   = DmrCalculateVatReportHeader.CalculatedTaxesAndFundsPayable;
       trans.ExchRate          = 100;
       trans.AccountType       = LedgerJournalACType::Ledger;
       trans.OffsetAccountType = LedgerJournalACType::Ledger;
       trans.insert();
   }
   else if(DmrCalculateVatReportHeader.CalculatedDeferredVat == 0 &&  DmrCalculateVatReportHeader.AsigNorKDV != 0 &&
           (DmrCalculateVatReportHeader.CalculatedTaxesAndFundsPayable < DmrCalculateVatReportHeader.AsigNorKDV))
   {
       select firstonly dmrCalculateVatReportSetup
       where  dmrCalculateVatReportSetup.TaxMainAccountType == DmrTaxMainAccountType::DeferredVat   ;

       trans.clear();
       trans.JournalNum        = ledgerJournalTableIns.JournalNum;
       trans.TransDate         = DmrCalculateVatReportHeader.EndDate;
       trans.LedgerDimension   = DimensionDefaultingService::serviceCreateLedgerDimension(DimensionStorage::getDefaultAccount(dmrCalculateVatReportSetup.MainAccountRecId),0);
       trans.Voucher           = Voucher;
       trans.Txt               = txt;
       trans.TaxCode           = element.getTaxCode(trans.LedgerDimension);
       trans.CurrencyCode      = ledger.AccountingCurrency;
       trans.AmountCurCredit   = DmrCalculateVatReportHeader.CalculatedTaxesAndFundsPayable;
       trans.ExchRate          = 100;
       trans.AccountType       = LedgerJournalACType::Ledger;
       trans.OffsetAccountType = LedgerJournalACType::Ledger;
       trans.insert();
   }
```

Şekil 4.35. Muhasebe kaydı oluşturma kod bloğu.

BÖLÜM 5

MICROSOFT DYNAMICS AX RAPORLAMA HİZMETİ: SSRS RAPORLARI

5.1. Microsoft Sql Server Raporlama Servisleri (SSRS)

SSRS raporları; analitik, yüksek kullanıcı ölçekli, çok işlevsel ve uygun maliyetli raporlar oluşturmanın en kolay ve verimli raporlama aracı olarak karşımıza çıkmaktadır. SSRS raporları, kullanıcıların istenilen formatta verilerin gruplanıp çıktı üretilmesini ve görüntülenmesini sağlayan bir araç niteliğindedir. SSRS, birbirleri ile etkileşimli raporların geliştirilmesine, tasarlanmasına ve sunulmasına olanak tanıyan bir raporlama platformu sağlar.

SSRS raporları, raporları tasarlamak ve özelleştirmek için Visual Studio (VS) kullanmaktadır. Kapsamlı bir raporlama yeteneğine sahip olan SSRS raporlama aracı excel, word ve PDF gibi sık kullanılan format tiplerine kolay bir şekilde veri kümelerini aktarabilmektedir. Microsoft Dynamics Ax 2012'de SSRS raporları geliştirmenin birçok yöntemi mevcuttur.

SSRS raporların tasarımını ve kullanımını yapabilmek için aşağıdaki araçlara ihtiyaç duyulmaktadır;

- Microsoft Dynamics Ax 2012
    - Raporlama çıktılarının gösterildiği ekranlar, kod bloglarının, classlarının, methodların yazıldığı bölümler,
- Visual studio sürümleri
    - SSRS raporlarının tasarımını ve özelleştirilebilir ekranlarının görsel anlamda yapılmasının sağlanması,

- SQL Server Raporlama Hizmetleri (SSRS) yapılandırılmalıdır
    - SSRS servislerinin çalıştırılması ve kullanılabilir olması,
- Raporlama hizmetleri uzantıları Dynamics Ax'a yüklenmelidir,
    - Dynamics Ax üzerinde ilgili SSRS raporlarının projeler üzerinden Visual Studio ile bağlantılarının yapılması,

5.2. SSRS Raporlarının Yapısı Ve Tasarımı

5.2.1. Rapor Veri Sağlayıcı (RDP) Sınıfı Tanımı

Rapor veri sağlayıcı Sınıfı, bir SSRS raporu için verilere erişmek ve bunları işlemek için kullanılan bir X++ sınıfıdır. RDP sınıfı, iş mantığını belirtilen parametrelere ve istenen sorguya göre raporlama hizmetlerinde bir veri kümesine dönüştürme hizmetini sağlayan yapıdır. Dynamics Ax'da bir RDP sınıfı oluşturmak için, bu sınıfı SRSReportDataProviderBase, SrsReportDataProvidERPreProcess ile genişletilmesi gerekecektir. Dynamics Ax'da bu sınıfın verilerini işlemek için raporlama hizmetleri tarafından kullanılacağının bilgisini aktarmaktadır. RDP sınıflarında iki önemli özellik kullanılır:

SRSReportQueryAttribute: Bu öznitelik raporda hangi AOT (Application Object Tree) sorgusunun kullanılacağını belirtir. RDP sınıfı verileri işlemek için bir AOT sorgusu kullanıyorsa, SRSReportQueryAttribute özniteliğinin sınıfın başında tanımlanması gerekmektedir.

SRSReportParameterAttribute: Parametre değerlerini sormak için rapor tarafından kullanılacak "contract" sınıfını tanımlar. RDP sınıfı herhangi bir parametre içeriyorsa, SRSReportParameterAttribute özniteliğinin sınıfın başında tanımlanması gerekir. Her iki nitelik de isteğe bağlıdır. Rapor herhangi bir sorgu kullanmıyorsa veya herhangi bir parametrenin rapor verilerini filtrelemesini istemiyorsa, bu özelliklerin kullanılmasına gerek yoktur.

5.2.1.1. Srsreportdataproviderbase, Srsreportdataproviderpreprocess Farkları

SRSReportDataProviderBase : Geçici tablo (InMemory) kullanarak SSRS raporu elde etmemizi sağlar. Fakat raporda zaman aşımı görülebilir. Ayrıca kodda debug yapılmasına olanak sağlamayacaktır.

SrsReportDataProvidERPreProcess : Verileri önceden işleyen özel bir Rapor Veri Sağlayıcı sınıfı türüdür. Regular tabloları kullanarak SSRS raporu elde edilmesini sağlar. Zaman aşımının önüne geçer. Kodda debug yapılmasına olanak sağlar. Kullanmak istendiğinde ek işlemler yapılması gereklidir.

Bazen Dynamics Ax'da hesapladığımız veya döndürdüğümüz veri miktarı nedeniyle Dynamics Ax'ta tanımlanan varsayılan çalışma süresi sınırlarını (varsayılan 10 dakikadır) aşan raporlar oluşturulması gerekir.

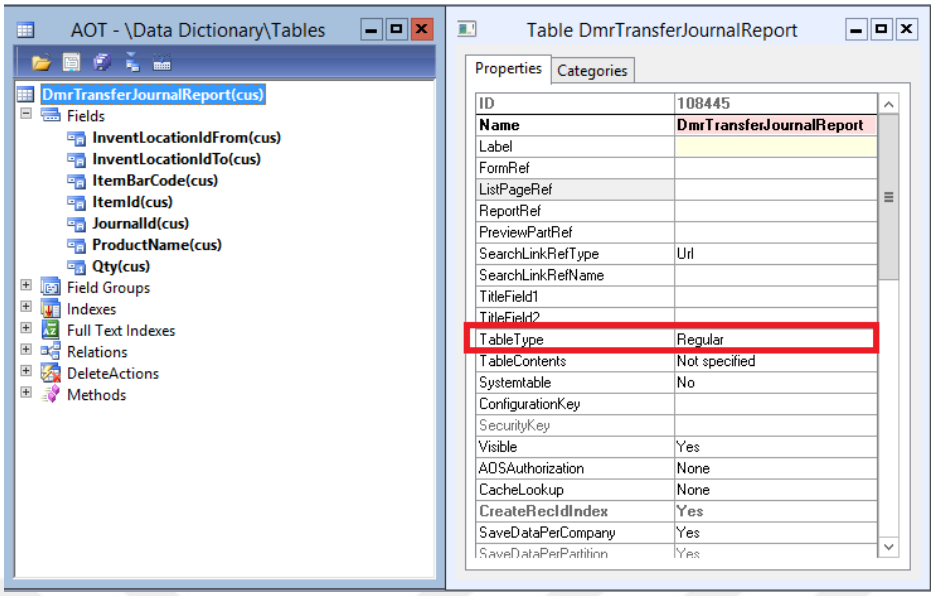
Eğer SRSReportDataProviderBase sınıfından türetildiyse ve fazla verinin işlenmesi gerekiyorsa zaman aşımına uğrayabilir. Zaman aşımı olarak sözü geçen konu bu şekilde açıklanabilir.

5.2.2. Contract Class (Veri Sözleşmesi Sınıfı)

Bir contract sınıfı, yöntemin başında tanımlanan DataMemberAttribute ile parm yöntemlerini içeren bir x++ sınıfıdır. Bu sınıf, bir SSRS raporunda kullanılacak bir veya daha fazla parametreyi tanımlamak için kullanılır.

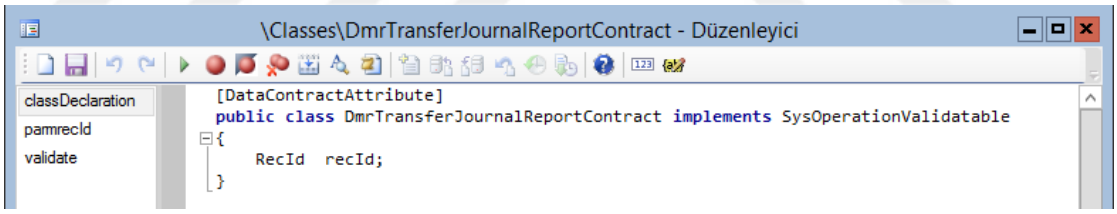
5.2.3. SSRS Rapor Tasarımı İçin Tablo Yapısı

Rapor için verileri depolamak için veri kümesi olarak bir Ax tablosu kullanılır. RDP sınıfı verileri işler ve daha sonra verileri oluşturmak için bir SSRS raporu tarafından kullanılan tabloda depolar.



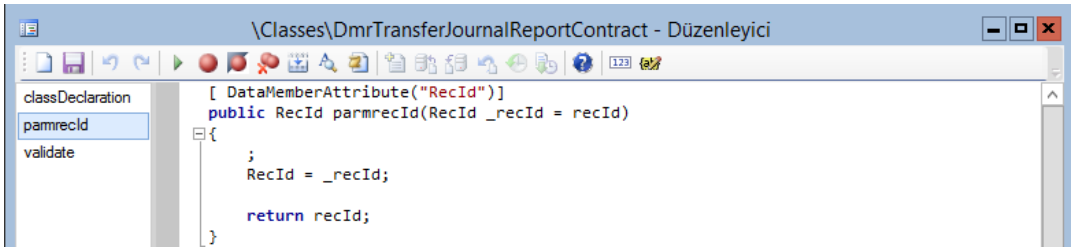
Şekil 5.1. SSRS rapor tasarımı için tablo özellikleri.

Tablo özelliklerinde tanımlanan "TableType" özelliğinin "Regular" tipinde olması tablonun fiziksel tablo olmasını belirler.



Şekil 5.2. Contract class tanımlaması.

DataContractAttribute özniteliği bir X ++ sınıfına uygulanır ve sınıfın bir veri sözleşmesi olarak kullanılabileceğini belirtir.



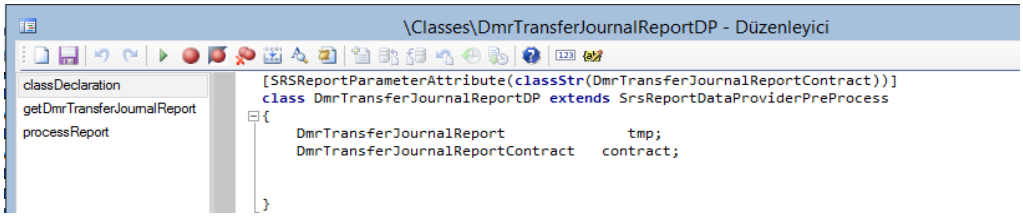
Şekil 5.3. Contract class metotu.

"DataMemberAttribute" özniteliği, bir X ++ veri sözleşmesi sınıfındaki "parm" yöntemine uygulanır. "Parm" metotlar, uzun adıyla parametrik metot olarak nitelendirilir. Genel anlamda değer alıp çıktı olarak değer döndüren metotlardır. Genellikle alınan değerleri bir nesneden başka bir nesneye taşımak için kullanılırlar. Parm metotlarla rapora vermek istediğimiz parametreler alınır. Dönüş türünün veri türü ve parametresi aynı olmalıdır.

Örneğin:

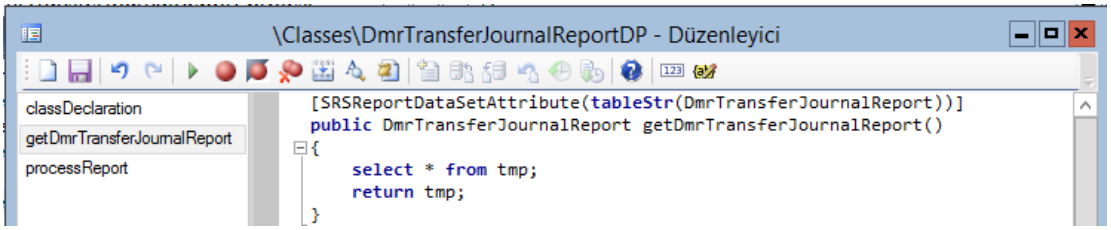
public RecId parmRecId(RecId _recId = recId)

public int parmLine(int _line = line)



Şekil 5.4. RDP sınıfı içerisinde tanımlanan classdeclaration metotu.

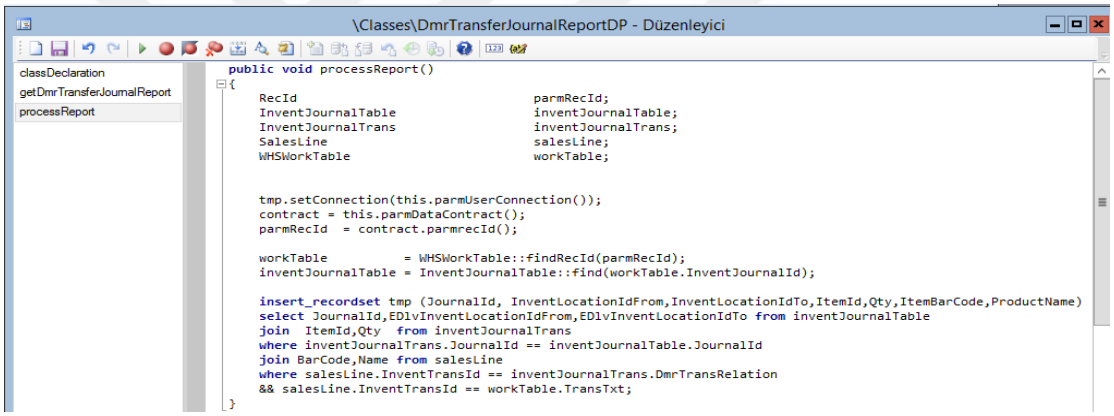
Şekil 5.4'de DmrTransferJournalReportDP adında yeni bir RDP sınıfı örneği bulunmaktadır. Tanımlanan sınıfın adlandırılmasında sonuna "DP" eklemek "Best practice" olarak daha uygun olacaktır. "DmrTransferJournalReportContract" sınıfın da tanımlanan parametreleri raporda kullanabilmek amacıyla "SRSReportParameterAttribute" özniteliği tanımlanmıştır. Verilen sınıf örneğinde "DmrTranferJournalReportDP" sınıfı "SrsReportDataProvidERPreProcess" sınıfından genişletilmiştir.



Şekil 5.5. RDP sınıfı içerisinde tanımlanan getDmrTransferJournalReport metotu.

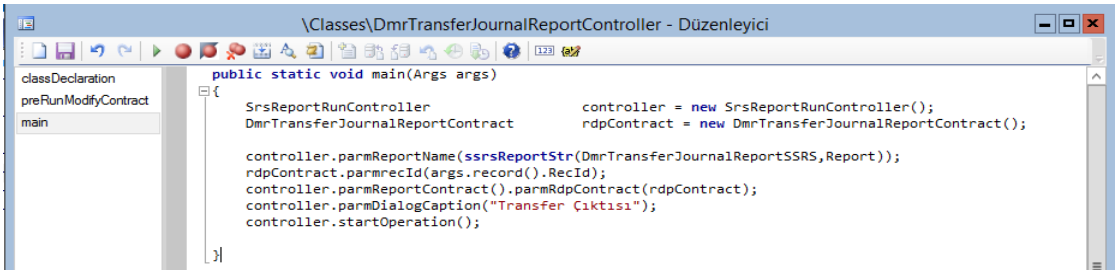
Şekil 5.5'de belirtilen metot SSRS raporları için RDP sınıfında zorunludur. Raporlama hizmetleri, işlenen verileri içeren tablo tamponunu almak için bu yöntemi kullanır.

"SRSReportDataSetAttribute" niteliği geçici tablo adını belirtmek için kullanılır.



Şekil 5.6. RDP sınıfı içerisinde tanımlanan processReport metotu.

Şekil 5.6'da raporun veri kümesini oluşturmak amacıyla yazılan ve raporlama hizmetleri tarafından çalıştırılan metot örneği verilmiştir. "Contract" sınıfı ile elde edilen "recId" üzerinden gerekli sorgular oluşturularak SSRS raporu için veri kümesi hazırlanmıştır.



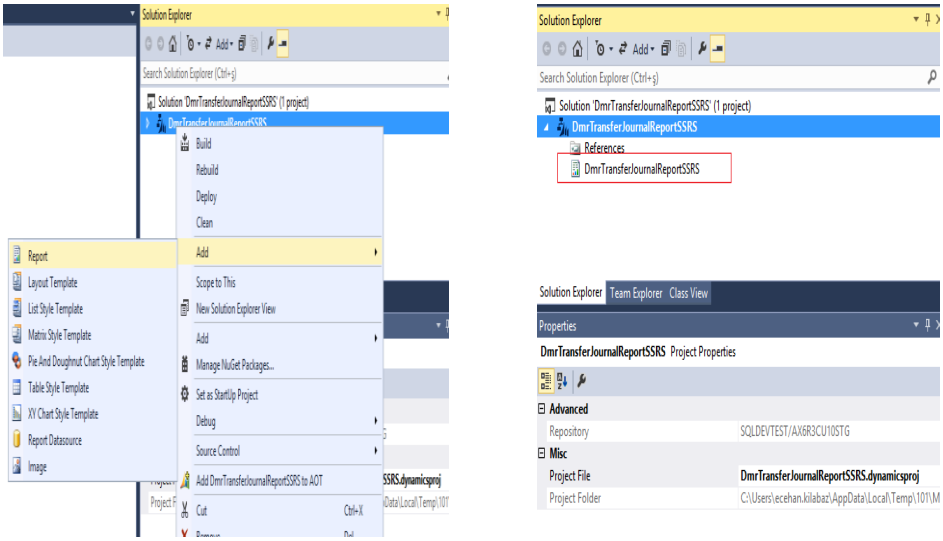
Şekil 5.7. Controller sınıfı içerisinde tanımlanan main metotu.

Bu sınıf raporun ilk tetiklendiği metottur. Bu sınıfın SSRS raporunun tasarımı içerisinde yazılması mecburi değildir. Sınıf yerine bir "buton"dan da aynı parametreleri göndererek SSRS raporu tetiklenebilmektedir. SSRS rapor tasarımları içerisinde birden fazla rapor tasarımı bulunabilmektedir. Bu tasarımları birbirlerinden ayırt etmek ve kullanıcıların raporları ayrı çalıştırmasını sağlanması amacıyla "parmReportName" parametresi kullanılmalıdır.

parmRecId: Kullanıcı tarafından seçilen kayıt üzerinden istenilen bilgilerin getirilmesi için RDP sınıfına kayıt bilgisinin iletilmesi için kullanılan metottur.

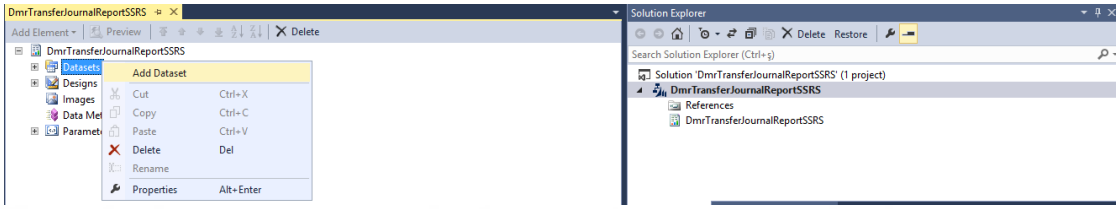
parmDialogCaption: Kullanıcının raporun başlık kısmında görüntüleyeceği ismini belirler.

5.2.4. SSRS Raporunun Visual Studio Ortamında Oluşturulması Ve Tasarımı



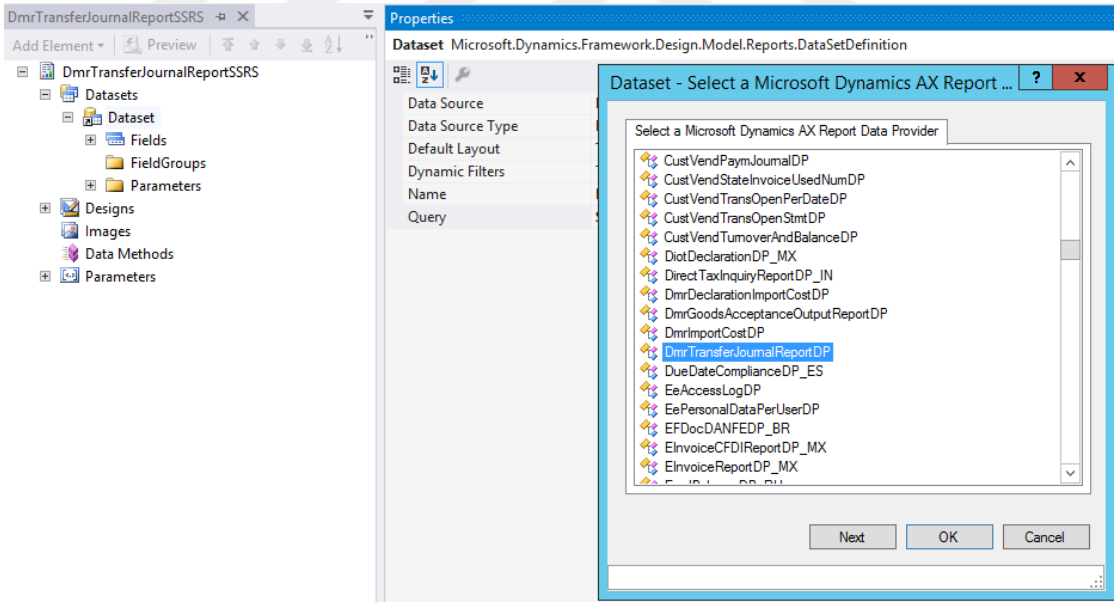
Şekil 5.8. Visual studio ssrs raporu oluşturma.

Şekil 5.8'de adımına gelmeden önce Dynamics Ax tarafında SSRS ile ilgili yapılması gereken adımlar anlatılmıştır. Dynamics Ax'a bağlanmış olan Visual Studio üzerinden gerekli tasarım ve özelleştirme işlemlerine başlanabilir. Şekil 5.8'de gösterildiği üzere projeye bir "Report" ekleyerek gerekli işlemler başlamaktadır. Eklenmiş hali Şekil 5.8'de üzerindeki sağ kısımdaki görselde belirtilmiştir.



Şekil 5.9. SSRS rapor tasarımına dataset eklenmesi.

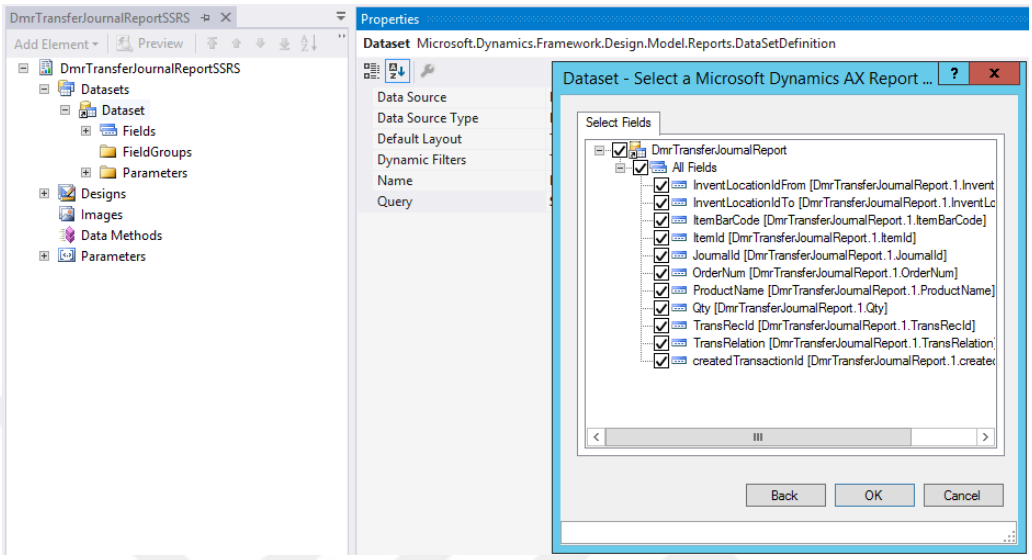
Şekil 5.9'da oluşturulan rapora bir "dataset" oluşturulmuştur.



Şekil 5.10. SSRS raporuna dataset özelliği.

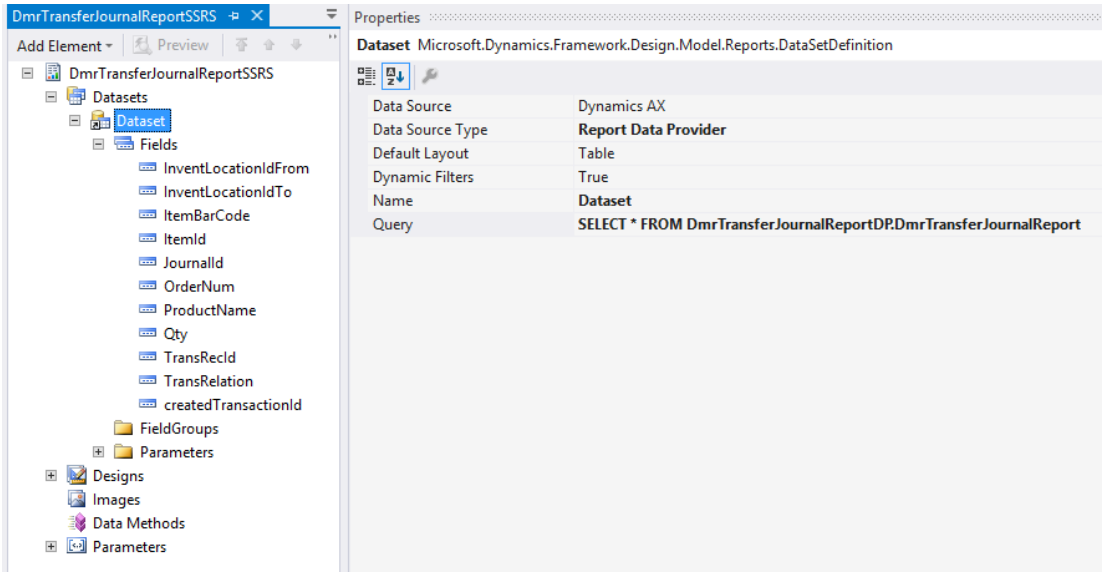
Şekil 5.9'da eklenen "dataSet" ile Dynamics Ax tarafında oluşturulan RDP sınıfı arasındaki bağlantıyı "Query" özelliği üzerinden birleştirilmektedir.

Şekil 5.10'daki ekrandan RDP sınıfı seçilerek "Next"e tıklanır ve sonraki adımlara geçilir.



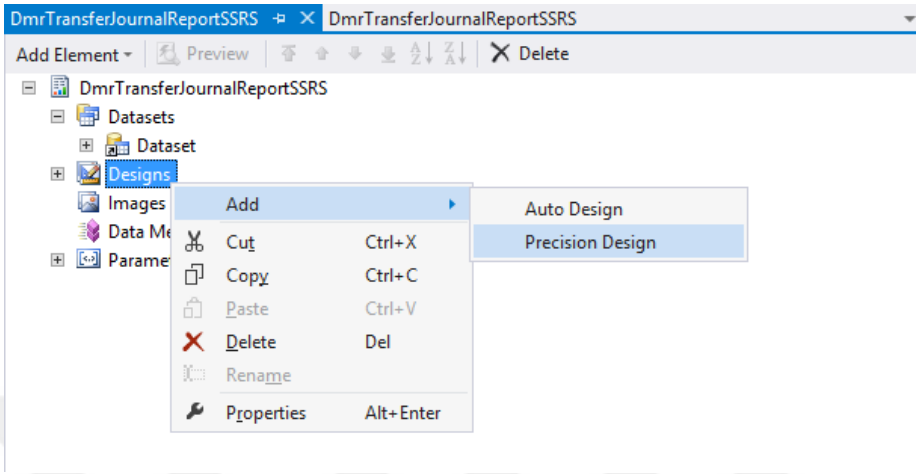
Şekil 5.11. SSRS rapor tasarımında kullanılacak alanların eklenmesi.

Şekil 5.10'da ilerletilen adımdan sonra Şekil 5.11'deki ekran görüntüsü açılmaktadır. Burada Dynamics Ax tarafında oluşturulan tablo alanları seçime sunulmaktadır. Raporda kullanıcıya görüntülemek istenilen alanların seçimini gerçekleştirip "OK" seçeneği üzerinden devam edilmelidir.



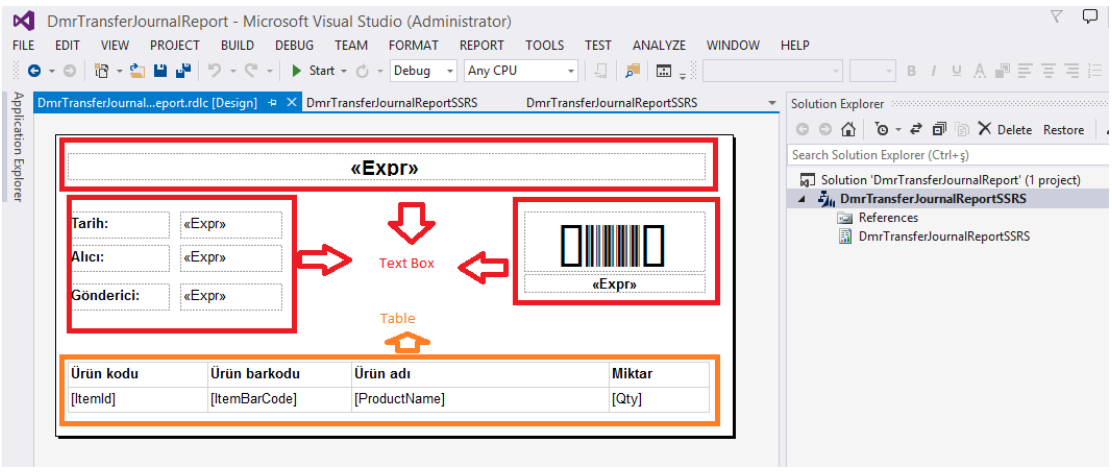
Şekil 5.12. SSRS rapor tasarımında dataset alanlarının oluşması.

Tüm adımlar tamamlandığında Şekil 5.12'de görüldüğü gibi Dynamic Ax tarafında oluşturulan alanlar otomatik olarak RDP sınıfı üzerinden rapora eklenmiştir.



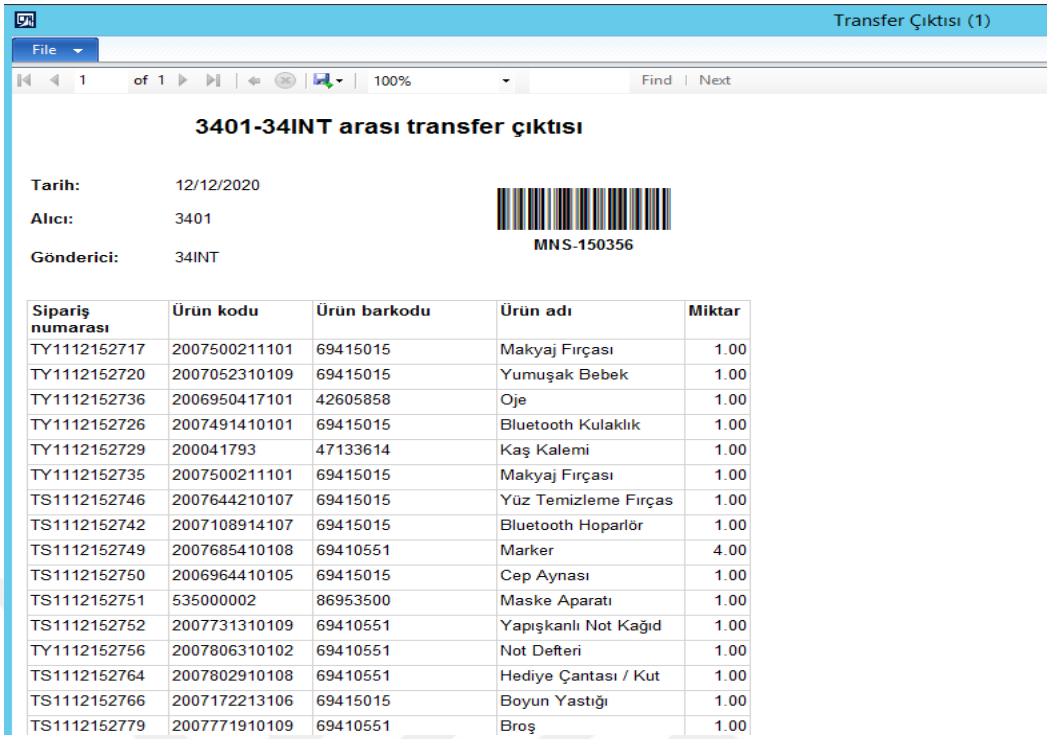
Şekil 5.13. Rapor tasarımının oluşturulması.

Şekil 5.13'de görüntüleneceği üzere "Design" özelliğine sağ tıklanarak "Add" seçeneği ile "Precision Design" fonksiyonu çağrılmaktadır. Bu fonksiyonun kullanılmasının ve seçilmesinin nedeni daha özelleştirilebilir raporlar oluşturulmasına olanak sağlamasıdır.



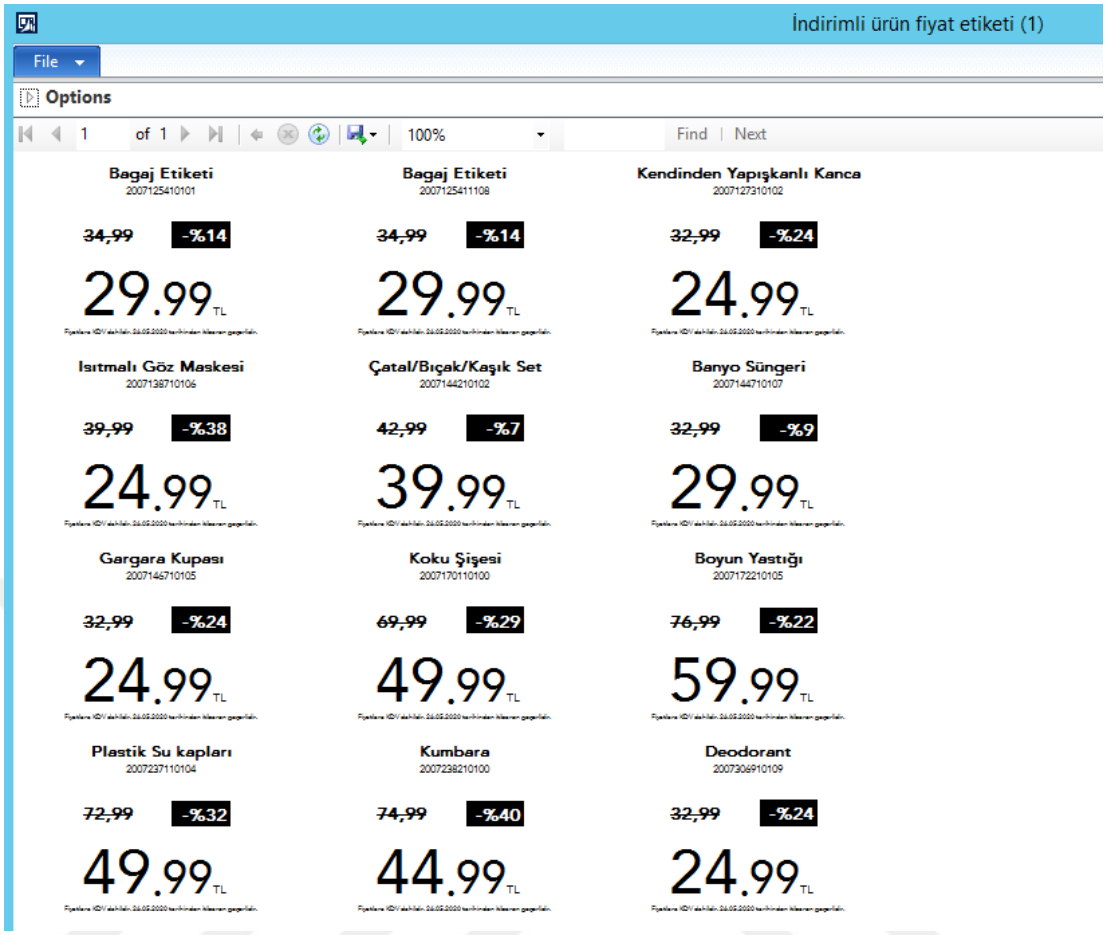
Şekil 5.14. Rapor tasarımının oluşturulması.

Tasarım kısmında istenilen işlemlere uygun objeler eklenir.



Şekil 5.15. SSRS rapor çıktısının kullanıcı tarafından görüntülenmesi.

Adım adım belirtilen işlemlerin sonucunda Şekil 5.15' deki rapor çıktısı elde edilmektedir. Bu çıktı sonuçları, kullanıcılar tarafından Excel,Word,PDF vb. raporlama araçlarına dönüştürülebilir.



Şekil 5.16. İndirimli ürün fiyat etiketi.

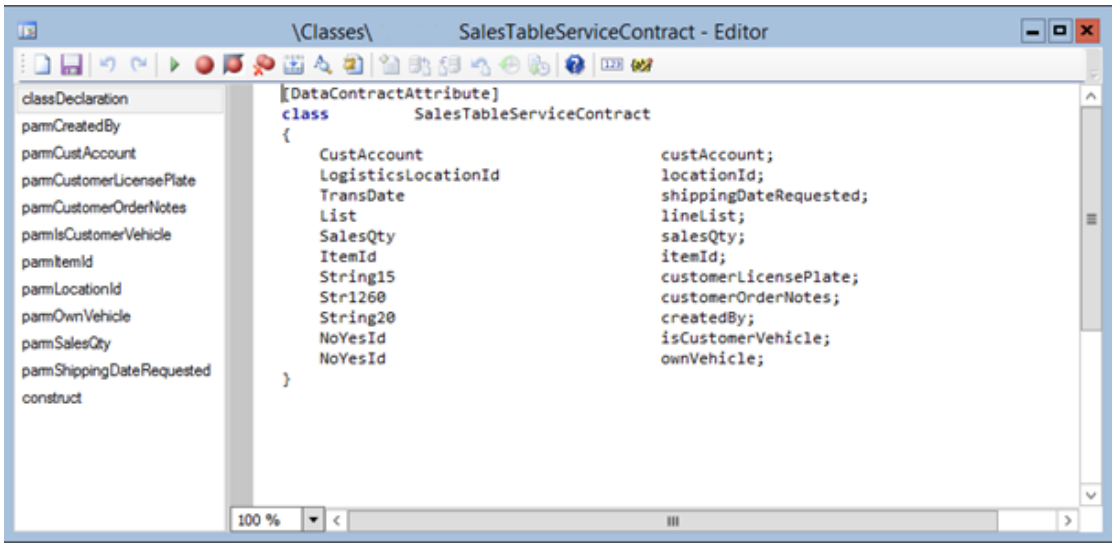
Şekil 5.16’ da bir müşteri için fiyat etiketlendirme çalışması sonucunda elde ettiğimiz görsel verilmiştir. SSRS raporları birçok alanda kullanıcıların işlerini kolaylaştırmakla beraber çok çeşitli kullanım alanları da mevcuttur.

BÖLÜM 6

DYNAMICS AX ÜZERİNDE ENTEGRASYON GELİŞTİRMESİ: WEB SERVİS

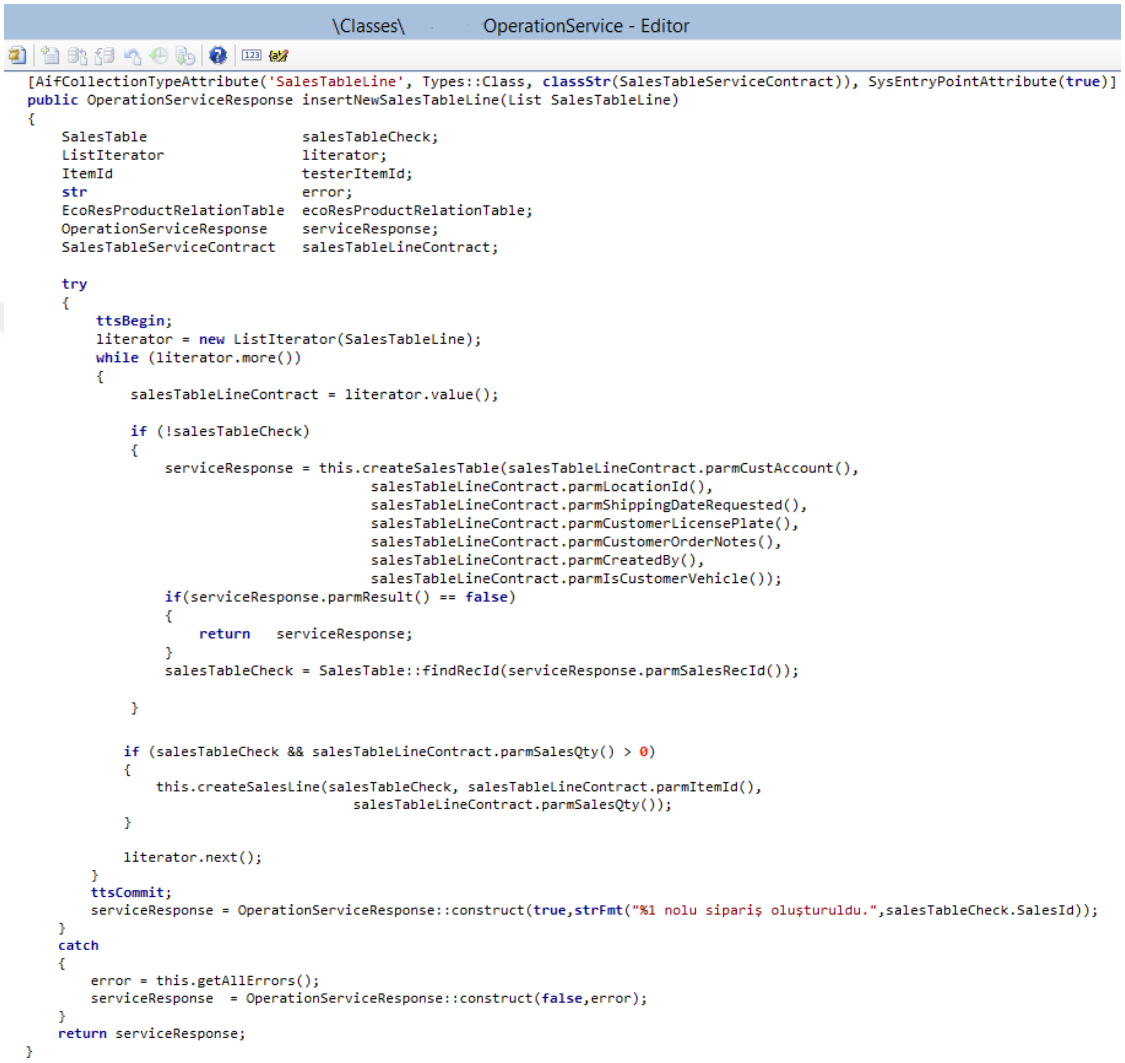
6.1. Web Servis İle Siparişlerin Dynamics Ax Üzerinde Oluşturulma Geliştirmesi

Müşteri, kendi müşterilerinin bir portal üzerinden sipariş verebilmesini, siparişlerinin durumunu takip edebilmesini ve sadece müşterilerinin yetkili oldukları ürünlere sipariş oluşturabilmesini istemektedir. Bunun için yeni bir web servis yazılarak sipariş alma, sipariş durumu gönderme, ürün listesini gönderme fonksiyonları kodlanmıştır.



Şekil 6.1. Service için contract class örneği.

Servis “contract” sınıfı ile SSRS raporlarında yazılana benzer bir işlem yapılmaktadır. “parm” metotları ile servise vermek istediğimiz parametreler alınır veya verilir. Şekil 6.1’de web servisten alacağımız verileri temsil eden türlerine uygun “parm” metotlar tanımlanmıştır.




```
[AifCollectionTypeAttribute('SalesTableLine', Types::Class, classStr(SalesTableServiceContract)), SysEntryPointAttribute(true)]
public OperationServiceResponse insertNewSalesTableLine(List SalesTableLine)
{
    SalesTable                  salesTableCheck;
    ListIterator                literator;
    ItemId                      testerItemId;
    str                         error;
    EcoResProductRelationTable  ecoResProductRelationTable;
    OperationServiceResponse    serviceResponse;
    SalesTableServiceContract   salesTableLineContract;

    try
    {
        ttsBegin;
        literator = new ListIterator(SalesTableLine);
        while (literator.more())
        {
            salesTableLineContract = literator.value();

            if (!salesTableCheck)
            {
                serviceResponse = this.createSalesTable(salesTableLineContract.parmCustAccount(),
                                  salesTableLineContract.parmLocationId(),
                                  salesTableLineContract.parmShippingDateRequested(),
                                  salesTableLineContract.parmCustomerLicensePlate(),
                                  salesTableLineContract.parmCustomerOrderNotes(),
                                  salesTableLineContract.parmCreatedBy(),
                                  salesTableLineContract.parmIsCustomerVehicle());
                if(serviceResponse.parmResult() == false)
                {
                    return   serviceResponse;
                }
                salesTableCheck = SalesTable::findRecId(serviceResponse.parmSalesRecId());

            }

            if (salesTableCheck && salesTableLineContract.parmSalesQty() > 0)
            {
                this.createSalesLine(salesTableCheck, salesTableLineContract.parmItemId(),
                                     salesTableLineContract.parmSalesQty());
            }

            literator.next();
        }
        ttsCommit;
        serviceResponse = OperationServiceResponse::construct(true,strFmt("%1 nolu sipariş oluşturuldu.",salesTableCheck.SalesId));
    }
    catch
    {
        error = this.getAllErrors();
        serviceResponse  = OperationServiceResponse::construct(false,error);
    }
    return serviceResponse;
}
```

Şekil 6.2. Web servisten veri alma.

Bir X ++ koleksiyonunu bir servis işlemine parametre olarak veya bir hizmet işleminden dönüş değeri olarak kullanılıyor ise, “AifCollectionTypeAttribute” özniteliğini belirtmek gerekmektedir.

Şekil 6.2’de gösterilen örnekte bir hizmet işlemi için dönüş değeri olarak kullanılan X ++ koleksiyonu “SalesTableServiceContract”ı gösterir.

Web servisten gelen “list” türündeki veriyi alarak içerdiği değerlerle “SalesTable” ve “SalesLine” kayıtları oluşturulmuştur. “List” ile gelen verilerin oluşumunda herhangi bir hata olmazsa “serviceResponse” ile Web servise oluşturulan siparişin numarası döndürülerek başarılı bir şekilde işlemin gerçekleştiği bilgisi verilmektedir. Aynı şekilde herhangi bir hata oluşursa “serviceResponse” ile hata mesajı gönderilecektir.

```
// Satış sipariş başlığı oluşturma
public OperationServiceResponse createSalesTable(CustAccount            _custAccount,
                                                  LogisticsLocationId    _locationId,
                                                  TransDate              _shippingDateRequested,
                                                  string15               _customerLicensePlate,
                                                  str 1260               _customerOrderNotes,
                                                  String20               _createdBy,
                                                  NoYes                  _isCustomerVehicle)

{
    SalesTable                      salesTable;
    LogisticsLocation               logisticsLocation;
    LogisticsPostalAddress          logisticsPostalAddress;
    DMRStationShipmentDays          stationShipmentDays;
    WeekDays                        WeekDaysValue;
    TmpFrmVirtual                   tmp;
    int                             diff;
    boolean                         flag=false;
    TimeHour24                      createdTime;
    TransDate                       createdDate;
    TransDate                       workDate;
    CustTable                       custTable = CustTable::find(_custAccount);
    SmmParametersTable              smmParametersTable = SmmParametersTable::find();
    OperationServiceResponse        OperationServiceResponse;
    str                             error;

    try
    {
        ttsBegin;
        salesTable.clear();
        salesTable.initValue();
        salesTable.SalesId              = NumberSeq::newGetNumFromId(SalesParameters::numRefSalesId().NumberSequenceId).num();
        salesTable.CustAccount          = _custAccount;
        salesTable.initFromCustTable();
        salesTable.DmrB2BConfirmation = false;
        salesTable.DmrB2BSales        = true;
        salesTable.SalesName          = custTable.name();
       if(custTable.InvoiceAccount)
            salesTable.InvoiceAccount = custTable.InvoiceAccount;
       else
            salesTable.InvoiceAccount = salesTable.CustAccount;
       salesTable.SalesType             = SalesType::Sales;
       salesTable.DmrIsFromPTSOrder     = NoYes::Yes;
       salesTable.DmrPTSCreatedBy       = _createdBy;
       salesTable.SalesStatus           = SalesStatus::Backorder;
       salesTable.DmrCustomerOrderNotes = _customerOrderNotes;
       salesTable.DmrCustomerLicensePlate = _customerLicensePlate;

       if (salesTable.SalesType == SalesType::ReturnItem)
       {
            SalesTable.TaxGroup ="Iade";
       }

       if( _isCustomerVehicle == NoYes::Yes)
            salesTable.DlvMode = dmrparameters::find().DlvModeId;

       select firstOnly logisticsLocation
       where  logisticsLocation.LocationId == _locationId;

       select firstOnly logisticsPostalAddress
       where logisticsPostalAddress.Location == logisticsLocation.RecId;

       salesTable.DeliveryPostalAddress = logisticsPostalAddress.RecId;
       salesTable.DeliveryName = LogisticsLocation.Description;

       salesTable.ShippingDateRequested = _shippingDateRequested;

       if(!SalesTable.validateWrite())
       {
            error = this.getAllErrors();
            OperationServiceResponse = OperationServiceResponse::construct(false,error);
       }
```

```
        else
        {
            salesTable.insert();
            OperationServiceResponse = OperationServiceResponse::construct(true,strFmt("%1 sipariş başlığı oluşturuldu.",SalesTable.SalesId),"",salesTable.RecId);
        }
        ttsCommit;

    }
    catch
    {
        error = this.getAllErrors();
        OperationServiceResponse = OperationServiceResponse::construct(false,error);
    }
    return OperationServiceResponse;
}
```

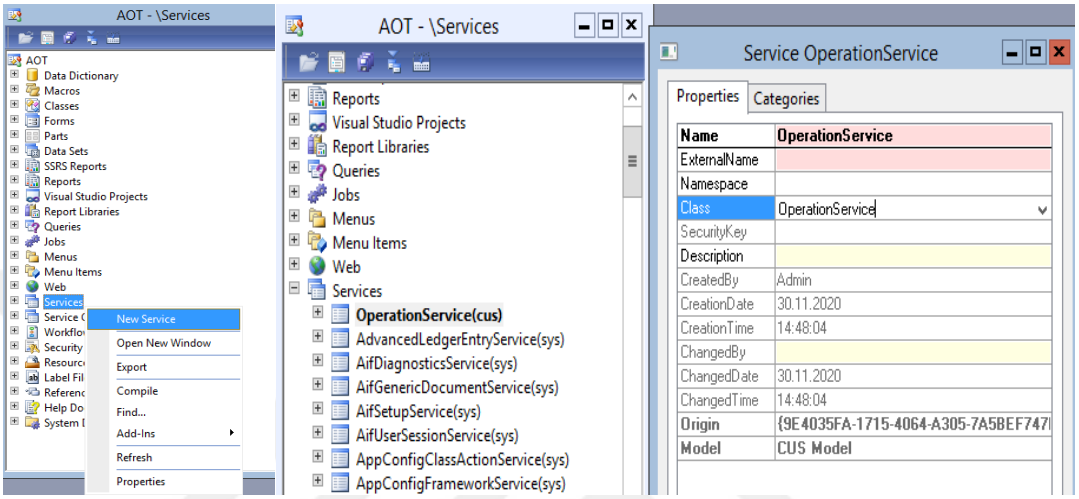
Şekil 6.3. Satış siparişinin başlığını oluşturma.

Şekil 6.3'de verilen kod satırları ile tablonun gerekli alanlarını web servisten gelen veriler doğrultusunda doldurulmaktadır. Kayıt hatasız oluşursa oluşan sipariş numarası kullanıcıya iletilir. Eğer bir hata gerçekleşirse hata mesajı da kullanıcıya iletilmektedir.

```
private OperationServiceResponse createSalesLine(SalesTable _salesTable,ItemId _itemId,SalesQty _salesQty)
{
    SalesLine                       SalesLine;
    str                             error;
    DmrCustumerItemWHSControl       DmrCustumerItemWHSControl;
    smmSalesDistrictId              salesDistrictId = CustTable::find(_salesTable.CustAccount).SalesDistrictId;//
    InventDim                       inventDim;
    CustGroup                       custGroup;
    boolean                         isPanaroma;
    InventItemBarcode               inventItemBarcode;
    DmrCustomSalesLog               DmrCustomSalesLog;
    DmrSalesLineDisc                dmrSalesLineDisc;
    OperationServiceResponse        OperationServiceResponse;
    int                             i = 1;
    ;

    try
    {
        ttsBegin;
        salesLine.clear();
        salesLine.initFromSalesTable(_salesTable);
        salesLine.ItemId                = _itemId;
        salesLine.initFromInventTable(InventTable::find(_itemId));
        salesLine.SalesQty              = _salesQty;
        salesLine.RemainSalesPhysical   = _salesQty;
        salesLine.DefaultDimension =
               DimensionDefaultingService::serviceMergeDefaultDimensions(salesLine.DefaultDimension,InventTable::find(salesLine.ItemId).DefaultDimension);
        inventDim.InventSiteId     = InventParameters::find().DmrPTSDefaultSite;
        inventDim.InventLocationId = _salesTable.InventLocationId;
        inventDim.inventBatchId    = DmrParameters::find().inventBatchId;
        if (_salesTable.InterCompanyDirectDelivery == false)
        {
            inventDim.wMSLocationId    = InventLocation::find(_salesTable.InventLocationId).WMSlocationIdDefaultIssue;
        }
        inventDim = inventDim::findOrCreate(inventDim);
        SalesLine.InventDimId = inventDim.inventDimId;
        salesLine.createLine(true,false,false,true,false,true,false,true,"",true,true, false);
        i++;

            info(strfmt("%1 : %2 ",salesLine.RecId));
            OperationServiceResponse = OperationServiceResponse::construct(true,salesLine.ItemId,salesLine.SalesId);

        ttsCommit;

    }
    catch
    {
        error = this.getAllErrors();
        OperationServiceResponse = OperationServiceResponse::construct(false,error);
    }
    return OperationServiceResponse;
}
```

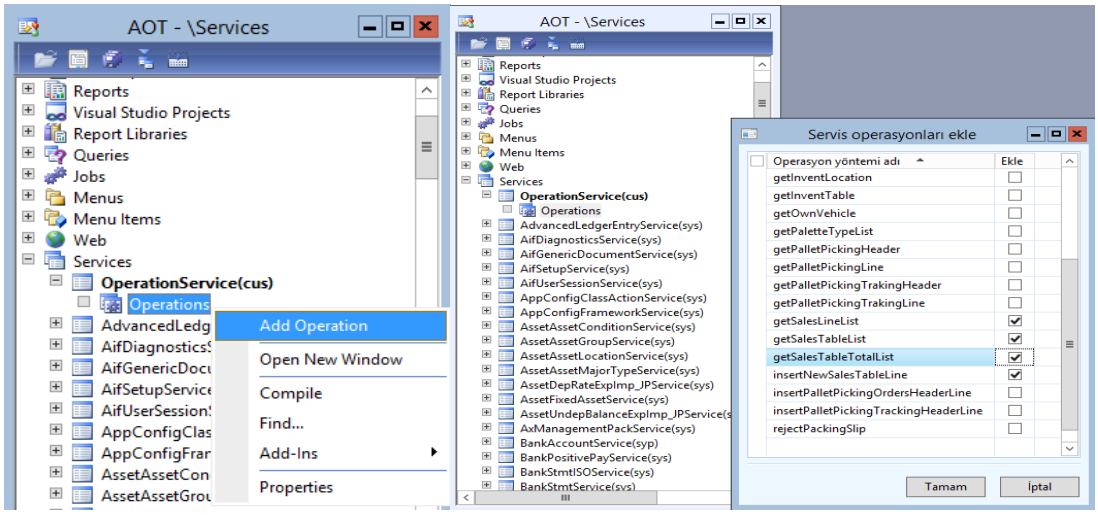
Şekil 6.4. Sipariş satırlarının oluşturulması.

Şekil 6.4'de verilen kod satırları ile tablonun gerekli alanları web servisten gelen veriler doğrultusunda doldurulmaktadır. Atamaları gerçekleştirildikten sonra hatasız oluşan sipariş numarası servise geri döndürülerek kullanıcı bilgilendirilmektedir. Eğer bir hata gerçekleşirse hata mesajı da kullanıcıya iletilmektedir.



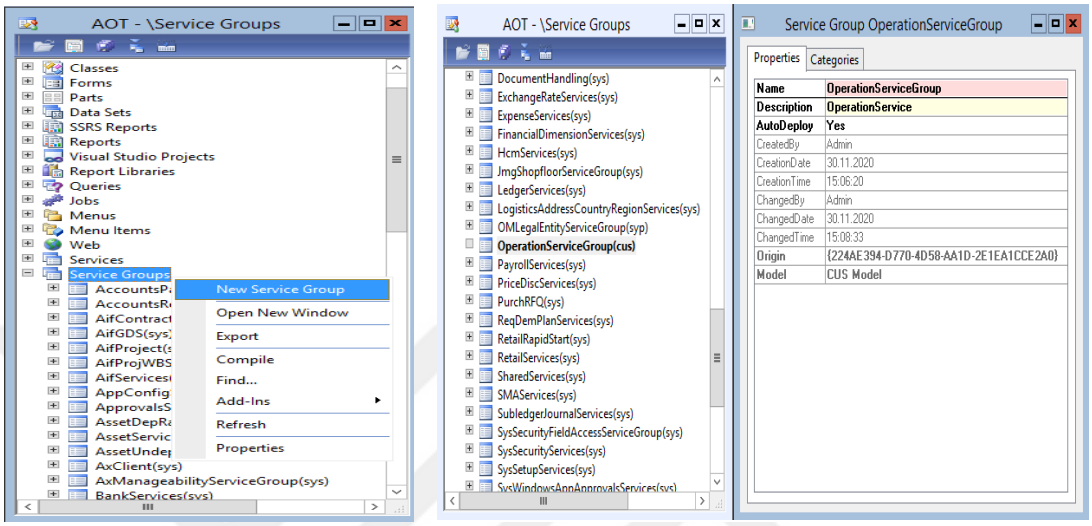
Şekil 6.5. Servis oluşturma ve gerekli tanımlamaların yapılması.

AOT nesnesinden service kısmına gelerek Şekil 6.5'de gösterildiği gibi sağ tık ile "New Service" denilmekte ve oluşan yeni servisin adı verilerek işlemlere devam edilmektedir.



Şekil 6.6. Oluşturulan metotlarının servise eklenmesi.

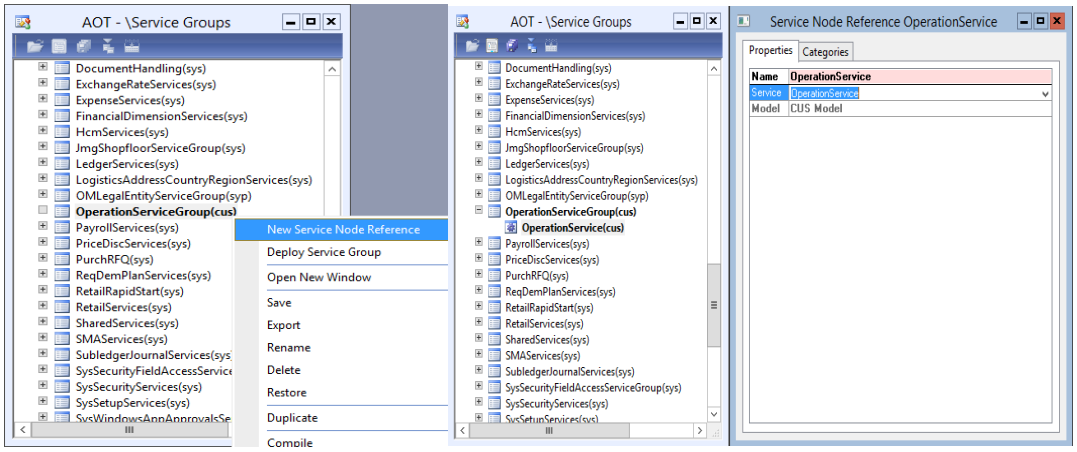
Şekil 6.6'da gösterildiği üzere oluşturulan "OperationService" adındaki servisin altında bulunan "Operation" düğümü üzerinden sağ tık ile "Add Operation" seçilerek açılan ekrandan servis için yazılan metotların seçimi sağlanmaktadır.



Şekil 6.7. Service groups oluşturma ve özellikleri.

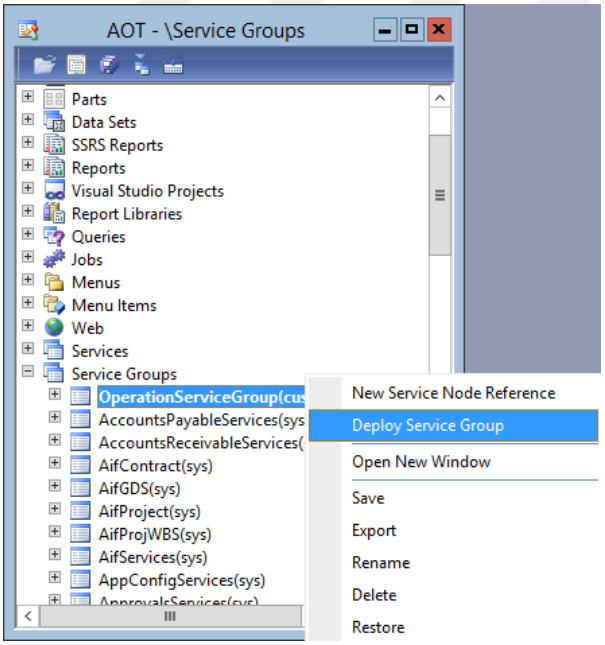
Bir hizmet grubundaki tüm hizmetler tek bir WSDL dosyasında yayınlanır. Tek bir WSDL (Web Services Description Language) dosyasına yayınlamak, geliştiricilerin kodlarındaki hizmetleri kullanma biçimini basitleştirir.

Bir geliştirici, AOT'da bir hizmet grubu oluşturabilir. Geliştiriciler, oluşturdukları hizmet gruplarının otomatik olarak dağıtılacağını ve oluşturulduklarında etkinleştirilip etkinleştirilmeyeceğini belirtmek için AutoDeploy özelliğini kullanabilir. Otomatik olarak etkinleştirilmeyen tüm hizmet gruplarını manuel olarak dağıtılması ve etkinleştirilmesi gerekmektedir.



Şekil 6.8. Service groups ile service ilişkisinin kurulması.

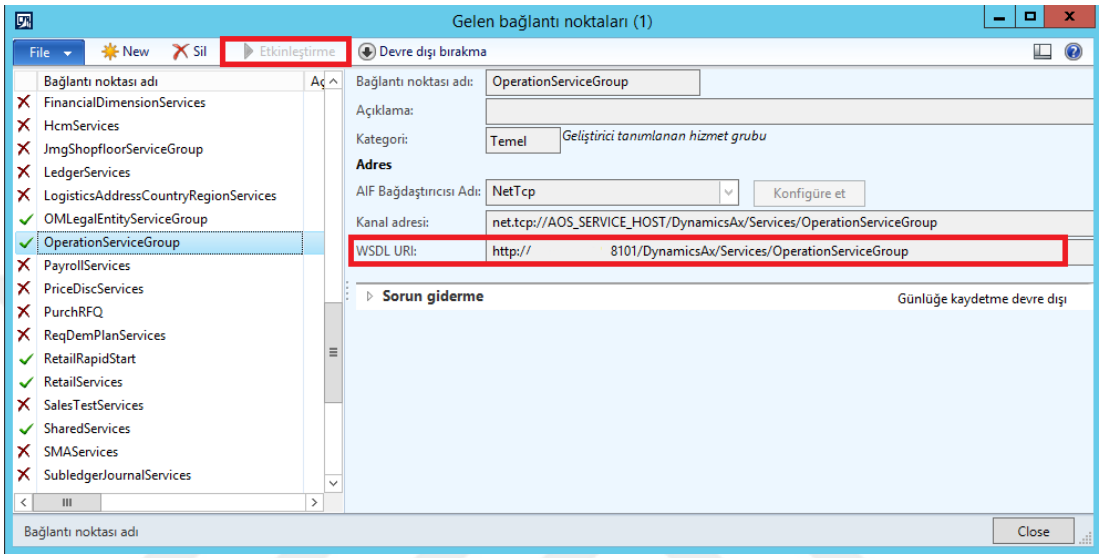
Şekil 6.8'de belirtildiği üzere oluşturulan "OperationServiceGroup" üzerinden sağ tık yapılarak "New Service Node Reference" seçilerek "service node" oluşturulmaktadır "service node"un özelliklerinden mevcut servis ile ilişkisi sağlanır.



Şekil 6.9. Servislerin deploy edilmesi.

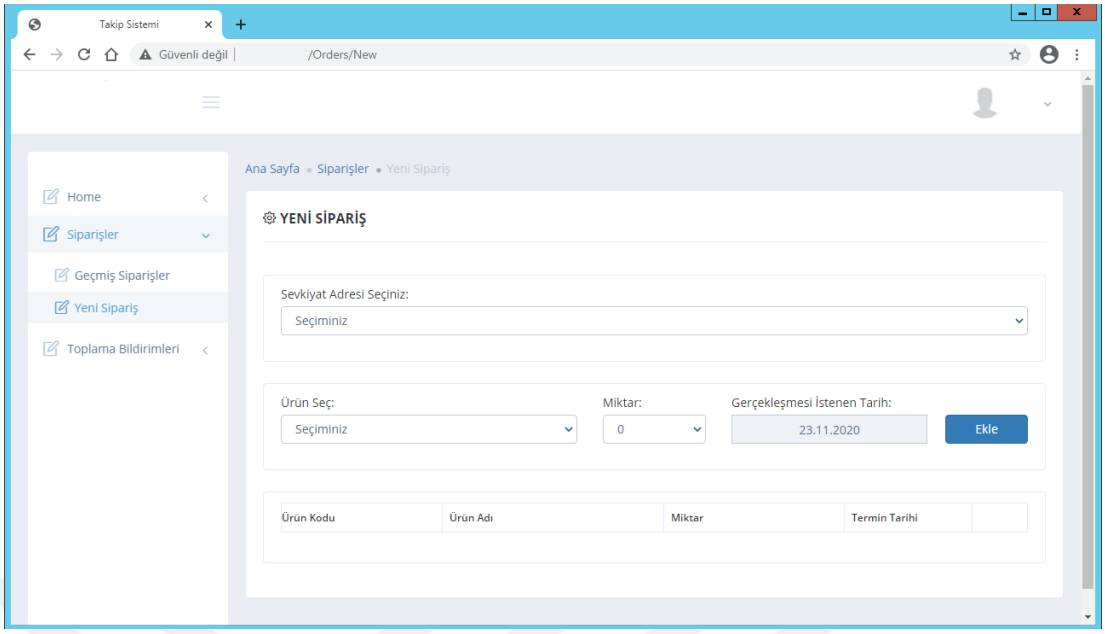
Servis oluşumunun tüm adımları tamamlandıktan sonra Dynamics Ax'da "CIL" derlemesi yapılmalıdır. Sınıflarda herhangi bir hata olmaması durumunda CIL başarılı

bir şekilde gerçekleşecektir. CIL başarılı bir şekilde gerçekleştikten sonra Şekil 6.9'da belirtildiği üzere oluşturulan "OperationServiceGroup" üzerinde sağ tık yapılarak "Deploy Service Group" seçilmelidir. Bu işlem ile tüm hizmetler tek bir WSDL dosyasında yayınlanmış olacaktır.



Şekil 6.10. Servisi etkinleştirme.

Oluşturulan servisi kullanabilmek için etkinleştirilmesi gerekmektedir. "CIL" ve "Deploy" işlemleri hatasız tamamlanan servis için Şekil 6.10'da belirtilen "Etkinleştirme" butonu aktif olur ve tıklandığında servis kullanıma hazır olur. Buton tıklandığında web sayfası ile ilişkinin kurulması için bir "WSDL URL" si elde edilir. Burada oluşan URL (Uniform Resource Loader) üzerinden Web sayfası üzerinden veri alışverişi sağlanır.



Şekil 6.11. Sipariş oluşturma web arayüzü.

Şekil 6.11'de belirtilen görsel üzerinden sipariş kaydı oluşturulur. Oluşan "WSDL URL" üzerinden veri alışverişi yapılarak oluşturulan kayıtların Dynamics Ax tarafında da gerekli tablolara kaydının oluşması sağlanır.

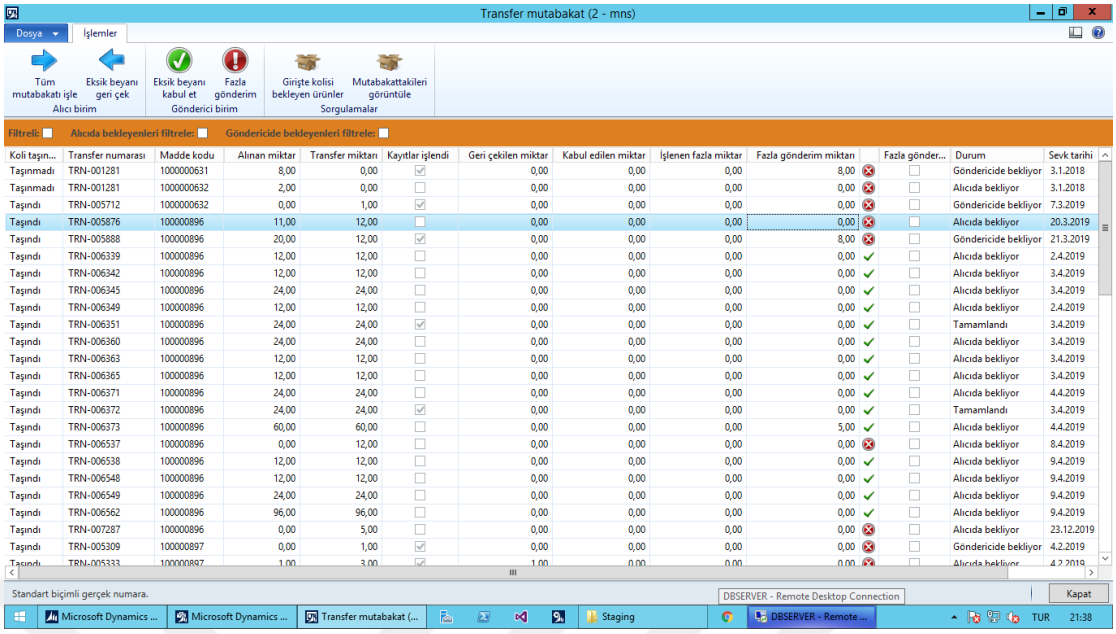
BÖLÜM 7

MICROSOFT DYNAMICS AX ÜZERİNDE DEPO GELİŞTİRMESİ: EL TERMİNALİ

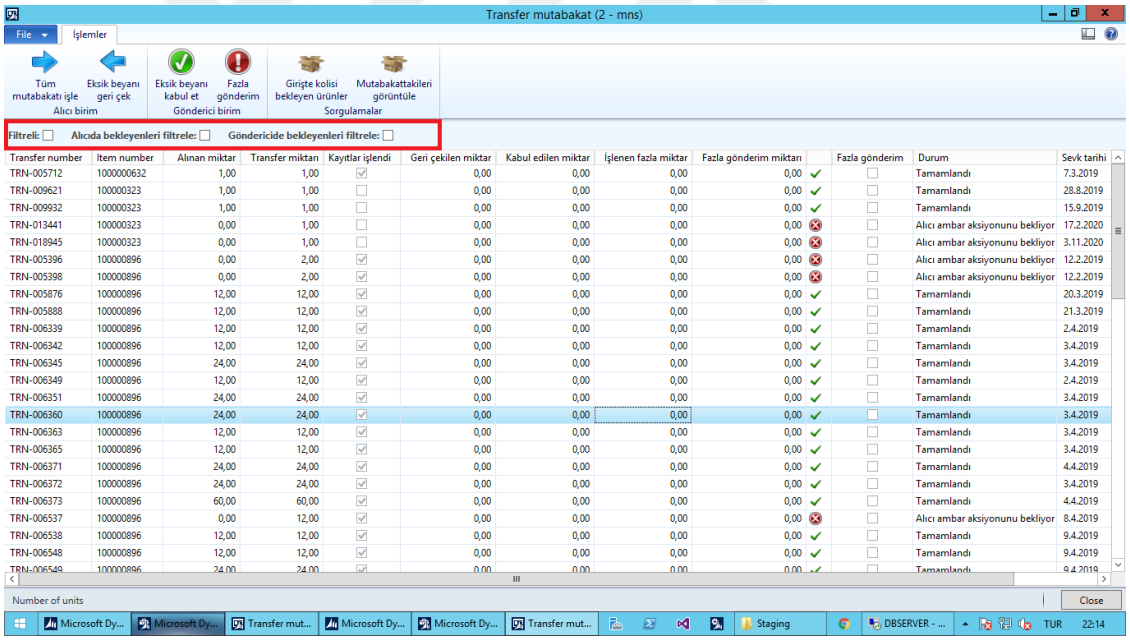
7.1. Mutabakat Modülü Geliştirmesi Ve El Terminalinin Oluşturulması

Müşteri, mağazaları ile depoları arasında gönderilen ürünlerin mutabakatlarının yapılamadığını, hangi ürünlerinin olup hangilerinin olmadığını veya eksik gönderiminin kimden kaynaklı olduğunu tespit edememekte, bu nedenle de stoklarını yönetememektedir. Bunun sonucunda mağazalarda kaydi stokla fiziksel stok arasında tutarsızlık oluşmaktadır. Bu tutarsızlıklar aynı zamanda operasyonel faaliyetleri de etkilemektedir.

Bu sorunun çözülmesi ve çift taraflı stok takibinin yapılabilmesi amacıyla depolar arası transfer mutabakat fonksiyonu kurgulanmıştır. Bu fonksiyona göre ürünler mağazadan merkeze veya merkezden mağazaya gönderildiğinde giriş- çıkış sırasında el terminali ile gönderilen ürünler ve mağazaya gelen ürünler okutularak alınacak, gönderilen ürün miktarıyla kabul edilen ürün miktarı arasında fark olması durumunda ilgili farklar liste halinde alıcı ve göndericinin mail adreslerine ve bölge genel müdürlerine atılacaktır. Aynı zamanda farklar kaydi olarak stok görünmesini engellemek adına mutabakat adında farklı bir depolama boyutunda tutulacaktır.



Şekil 7.1. Sürecin yönetileceği ekran.



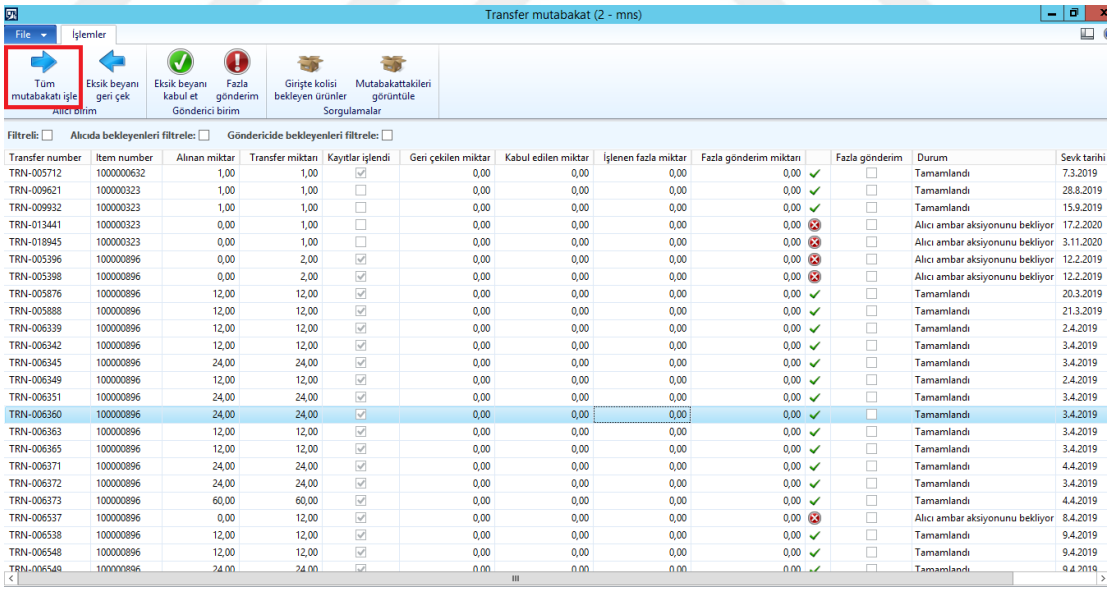
Şekil 7.2. Alıcı ve göndericiye göre filtreleme fonksiyonları.

Şekilde belirtilen kutucuklar yardımıyla alıcı ve gönderici kendisinde bekleyen aksiyonları filtreleyebilmektedir.

```
public boolean modified()
{
    boolean ret;

    ret = super();

    if(ReceiverFilter.value() == NoYes::Yes)
    {
        DmrTransMutabakat_DS.queryRun().query().dataSourceTable(tablenum(DmrTransMutabakat)).addRange(fieldnum(DmrTransMutabakat,TransferMutabakatStatus)).value(queryValue(DmrTransferMutabakatStatus::WaitingToReceiver));
        DmrTransMutabakat_DS.research();
    }
    else if(receiverFilter.value() == NoYes::No)
    {
        DmrTransMutabakat_DS.queryRun().query().dataSourceTable(tablenum(DmrTransMutabakat)).clearRange(fieldnum(DmrTransMutabakat, TransferMutabakatStatus));
        DmrTransMutabakat_DS.research();
    }

    return ret;
}
```

Şekil 7.3. Alıcıda bekleyenleri filtrele fonksiyonu.

"Modified()" metotlar alan bazında düzenlenir. Alanda herhangi bir değişiklik meydana geldiğinde tetiklenmektedirler. Şekil 7.3'de verilen kod bloğunda "ReceiverFilter" alanı için yazılmış olup alanın "Yes" veya "No" olma durumlarına göre farklı işlevler gerçekleştirmektedir. Alan değeri "Yes" olarak değiştirildiğine "DmrTransMutabakat" tablosunun "TransferMutabakatStatus" alanına "WaitingToReceiver" değeri verilerek filtreleme sağlanmıştır. Alan değeri "No" olarak değiştirildiğine ise "TransferMutabakatStatus" alanı üzerindeki tüm filtreler kaldırılmaktadır.



Transfer number	Item number	Alınan miktar	Transfer miktarı	Kayıtlar işlendi	Geri çekilen miktar	Kabul edilen miktar	İşlenen fazla miktar	Fazla gönderim miktarı		Fazla gönderim	Durum	Sevk tarihi
TRN-005712	1000000632	1,00	1,00	☑	0,00	0,00	0,00	0,00	✓	☐	Tamamlandı	7.3.2019
TRN-009621	100000323	1,00	1,00	☐	0,00	0,00	0,00	0,00	✓	☐	Tamamlandı	28.8.2019
TRN-009932	100000323	1,00	1,00	☐	0,00	0,00	0,00	0,00	✓	☐	Tamamlandı	15.9.2019
TRN-013441	100000323	0,00	1,00	☐	0,00	0,00	0,00	0,00	✗	☐	Alıcı ambar aksiyonunu bekliyor	17.2.2020
TRN-018945	100000323	0,00	1,00	☐	0,00	0,00	0,00	0,00	✗	☐	Alıcı ambar aksiyonunu bekliyor	3.11.2020
TRN-005396	100000896	0,00	2,00	☑	0,00	0,00	0,00	0,00	✗	☐	Alıcı ambar aksiyonunu bekliyor	12.2.2019
TRN-005398	100000896	0,00	2,00	☑	0,00	0,00	0,00	0,00	✗	☐	Alıcı ambar aksiyonunu bekliyor	12.2.2019
TRN-005876	100000896	12,00	12,00	☑	0,00	0,00	0,00	0,00	✓	☐	Tamamlandı	20.3.2019
TRN-005888	100000896	12,00	12,00	☑	0,00	0,00	0,00	0,00	✓	☐	Tamamlandı	21.3.2019
TRN-006339	100000896	12,00	12,00	☑	0,00	0,00	0,00	0,00	✓	☐	Tamamlandı	2.4.2019
TRN-006342	100000896	12,00	12,00	☑	0,00	0,00	0,00	0,00	✓	☐	Tamamlandı	3.4.2019
TRN-006345	100000896	24,00	24,00	☑	0,00	0,00	0,00	0,00	✓	☐	Tamamlandı	3.4.2019
TRN-006349	100000896	12,00	12,00	☑	0,00	0,00	0,00	0,00	✓	☐	Tamamlandı	2.4.2019
TRN-006351	100000896	24,00	24,00	☑	0,00	0,00	0,00	0,00	✓	☐	Tamamlandı	3.4.2019
TRN-006360	100000896	24,00	24,00	☑	0,00	0,00	0,00	0,00	✓	☐	Tamamlandı	3.4.2019
TRN-006363	100000896	12,00	12,00	☑	0,00	0,00	0,00	0,00	✓	☐	Tamamlandı	3.4.2019
TRN-006365	100000896	12,00	12,00	☑	0,00	0,00	0,00	0,00	✓	☐	Tamamlandı	3.4.2019
TRN-006371	100000896	24,00	24,00	☑	0,00	0,00	0,00	0,00	✓	☐	Tamamlandı	4.4.2019
TRN-006372	100000896	24,00	24,00	☑	0,00	0,00	0,00	0,00	✓	☐	Tamamlandı	3.4.2019
TRN-006373	100000896	60,00	60,00	☑	0,00	0,00	0,00	0,00	✓	☐	Tamamlandı	4.4.2019
TRN-006537	100000896	0,00	12,00	☑	0,00	0,00	0,00	0,00	✗	☐	Alıcı ambar aksiyonunu bekliyor	8.4.2019
TRN-006538	100000896	12,00	12,00	☑	0,00	0,00	0,00	0,00	✓	☐	Tamamlandı	9.4.2019
TRN-006548	100000896	12,00	12,00	☑	0,00	0,00	0,00	0,00	✓	☐	Tamamlandı	9.4.2019
TRN-006549	100000896	24,00	24,00	☑	0,00	0,00	0,00	0,00	✓	☐	Tamamlandı	9.4.2019

Şekil 7.4. Tüm mutabakatları işle fonksiyonu.

Tüm mutabakatları işle fonksiyonu ile alıcı mağazasına gelen ürünlerin mal kabul işlemlerini yaptıktan sonra eksik ürünler varsa bunlarla ilgili bildirim maillerini

göndermektedir. Bu sayede alıcıya eksik gelen ürünler mutabakat depolama boyutu kırılımında ayrıştırılarak göndericinin aksiyonunu bekler.

```
void clicked()
{
    DmrCreateTransferJournalMutabakat       DmrCreateTransferJournalMutabakat;
    DmrCreateTransferJournalMutabakatMail   DmrCreateTransferJournalMutabakatMail;
    DmrTransMutabakat                       DmrTransMutabakatNew;
    DmrTransMutabakatTmp                    dmrTransMutabakatTmp;
    WHSWorkLine                             WHSWorkLine;
    WHSWorkUser                             wHSWorkUser;
    InventLocationId                        inventLocationIdto;

    super();

    inventLocationIdto = InventTransferTable::find(DmrTransMutabakat.InventTransferId).InventLocationIdto;
    select firstOnly wHSWorkUser
    where  wHSWorkUser.UserDefaultWarehouse == inventLocationIdto
    &&     wHSWorkUser.DmrUserId            == curUserId();

    if(box::yesno('Kayıtlar işlenecek emin misiniz?  ', DialogButton::No) == DialogButton::Yes)
    {
        while select DmrTransMutabakatNew
        where DmrTransMutabakatNew.InventTransferId == DmrTransMutabakat.InventTransferId
        &&    !DmrTransMutabakatNew.IsPosted
        {
            ttsBegin;
            element.insertDmrTransferMutabakatHistory(DmrTransMutabakatNew.RecId,
                                                      DmrTransMutabakatNew.TransferQty,
                                                      DmrTransMutabakatNew.Qty);
            if(DmrTransMutabakatNew.TransferQty < DmrTransMutabakatNew.Qty)
            {
                dmrTransMutabakatTmp.clear();
                dmrTransMutabakatTmp.ItemId             = DmrTransMutabakatNew.ItemId;
                dmrTransMutabakatTmp.InventTransferId   = DmrTransMutabakatNew.InventTransferId;
                dmrTransMutabakatTmp.Qty                = DmrTransMutabakatNew.Qty;
                dmrTransMutabakatTmp.SysUserId          = curUserId();
                dmrTransMutabakatTmp.TransferQty        = DmrTransMutabakatNew.TransferQty;
                dmrTransMutabakatTmp.SessionId          = SessionId();
                select firstOnly WHSWorkLine
                where  WHSWorkLine.OrderNum == dmrTransMutabakatTmp.InventTransferId;

                dmrTransMutabakatTmp.CollectingUser     = WHSWorkLine.UserId;
                dmrTransMutabakatTmp.insert();
                DmrTransMutabakatNew.selectForUpdate(true);
                DmrTransMutabakatNew.IsPosted = NoYes::Yes;
                DmrTransMutabakatNew.SentExtraQty = abs(DmrTransMutabakatNew.Qty - DmrTransMutabakatNew.TransferQty);
                DmrTransMutabakatNew.TransferMutabakatStatus = DmrTransferMutabakatStatus::WaitingToSender;
                DmrTransMutabakatNew.update();

            }
            else if(DmrTransMutabakatNew.TransferQty > DmrTransMutabakatNew.Qty)
            {
                dmrTransMutabakatTmp.clear();
                dmrTransMutabakatTmp.ItemId             = DmrTransMutabakatNew.ItemId;
                dmrTransMutabakatTmp.InventTransferId   = DmrTransMutabakatNew.InventTransferId;
                dmrTransMutabakatTmp.Qty                = DmrTransMutabakatNew.Qty;
                dmrTransMutabakatTmp.SysUserId          = curUserId();
                dmrTransMutabakatTmp.TransferQty        = DmrTransMutabakatNew.TransferQty;
                dmrTransMutabakatTmp.SessionId          = SessionId();
                select firstOnly WHSWorkLine
                where  WHSWorkLine.OrderNum == dmrTransMutabakatTmp.InventTransferId;

                dmrTransMutabakatTmp.CollectingUser     = WHSWorkLine.UserId;
                dmrTransMutabakatTmp.insert();

                DmrCreateTransferJournalMutabakat = new DmrCreateTransferJournalMutabakat();
                DmrCreateTransferJournalMutabakat.parmDmrTransMutabakat(DmrTransMutabakatNew);
                DmrCreateTransferJournalMutabakat.run();
            }
            else
            {
                DmrTransMutabakatNew.selectForUpdate(true);
                DmrTransMutabakatNew.IsPosted = true;
                DmrTransMutabakatNew.update();
            }
            ttsCommit;

        }

        }
        DmrCreateTransferJournalMutabakatMail = new DmrCreateTransferJournalMutabakatMail();
        DmrCreateTransferJournalMutabakatMail.run();

    }
    DmrTransMutabakat_ds.research(true);

}
```

Şekil 7.5. Tüm mutabakatları işle fonksiyonu kod bloğu.

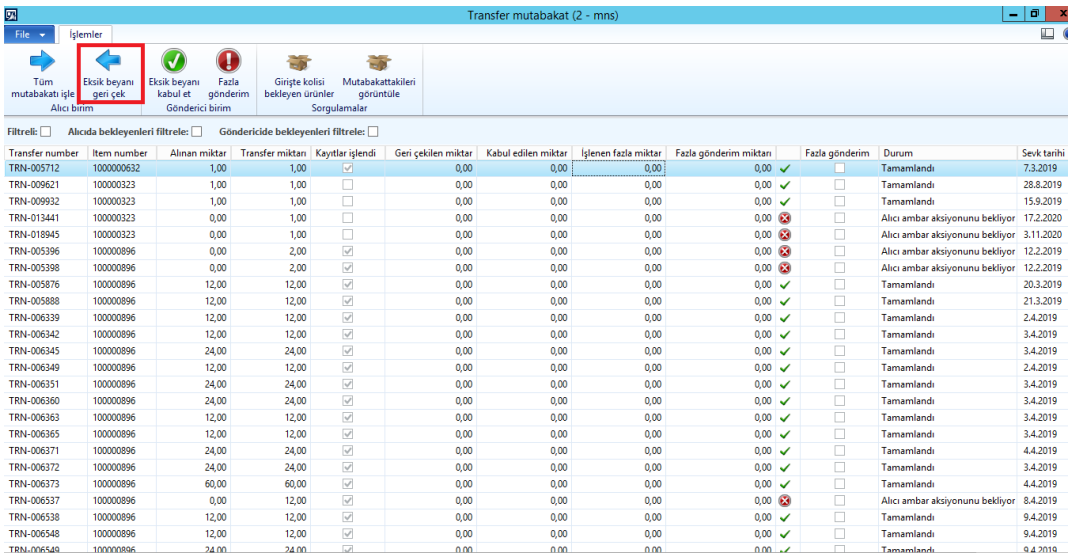
Şekil 7.5'de verilen kod bloğu ile kullanıcının ekranına bir messagebox çıkarttırılarak "Kayıtlar işlenecek emin misiniz?" seçeneği ile işleme devam edilip edilmeyeceği bilgisi istenmektedir. Eğer seçimi "Yes" ise işlemlere devam edilmektedir. "Transfer Mutabakat" tablosundaki verilerden "Kayıtlar işlendi" alanı boş olanlar elde edilmektedir. Elde edilen kayıtlar için üç farklı işlem uygulanır.

Birinci işlem: Transfer miktarı, alınan miktardan küçük olan kayıtlar içindir. Bu kayıtlar daha sonraki adımlarda kullanılabilmek amacıyla "DmrTransMutabakatTmp" adında bir tabloya doldurulmaktadır. "Transfer Mutabakat" tablosundaki Fazla gönderim miktarı alanı için hesaplamalar yapılır. "Durum" alanı "Göndericide bekliyor" ve "Kayıtlar işlendi" alanı "Yes" olmak üzere güncellenirler.

İkinci işlem: Transfer miktarı, alınan miktardan büyük olan kayıtlar içindir. Bu kayıtlar daha sonraki adımlarda kullanılabilmek amacıyla "DmrTransMutabakatTmp" adında bir tabloya doldurulmaktadır. Sonrasında "Transfer mutabakat" günlükleri oluşturulmuştur.

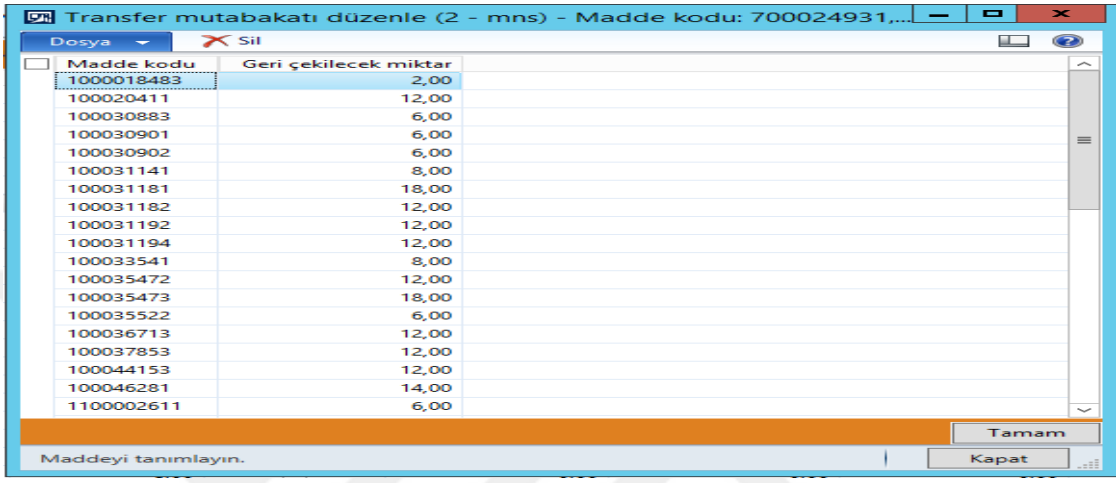
Üçüncü işlem: Transfer miktarı ile alınan miktarın eşit olduğu kayıtlar içindir. "Kayıtlar işlendi" alanı "Yes" olarak güncellenmektedir.

Bu üç işlemin sonunda "DmrTransMutabakatTmp" tablosunda oluşturulan kayıtlar ile kullanıcıya Transfer mutabakat sonuçları listelenir. Listelenen bu kayıtlar kullanıcılara mail gönderimi ile beraber bilgilendirme yapılmaktadır.



Şekil 7.6. Eksik beyanı geri çek fonksiyonu.

Tüm mutabakatları işle fonksiyonu ile yapılan eksik bildiriminin hatalı yapıldığı tespit edildiği durumda gönderici eksik beyanı geri çek fonksiyonunu görselde belirtilen buton aracılığıyla tetikleyerek açılan form üzerinde belirttiği ürünleri deposuna geri almış olur.



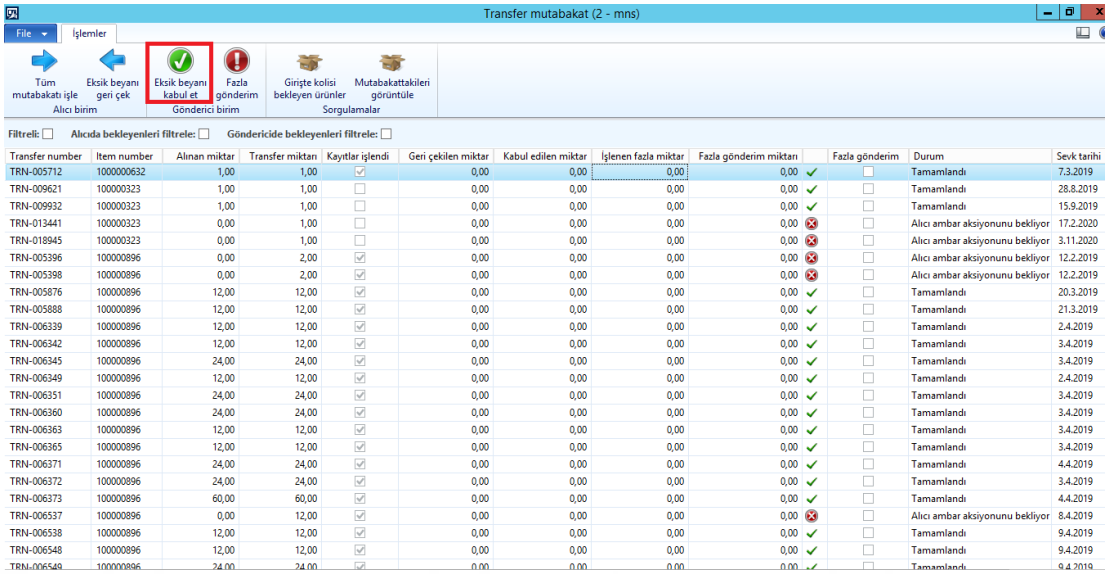
Şekil 7.7. Eksik beyanı geri çek butonu ile açılan form.

Alıcı hangi ürünlerden kaçar adet geri çekileceğini Şekilde gösterilen form üzerinde belirterek tamam butonun tıklar. Böylece belirttiği ürünleri deposuna geri almış olur.

```
public void closeOk()
{
    DmrTransferWhsTmp tmp;
    InventJournalTable      journalTable;
    InventParameters        InventParameters;
    InventjournalTableData  journalTableData;
    JournalCheckPost        journalCheckPost;
    DmrTransMutabakatTmp    dmrTransMutabakatTmp;
    WHSWorkLine             WHSWorkLine;
    DmrCreateTransferJournalMutabakatMail   DmrCreateTransferJournalMutabakatMail;
    str emailBodyForBrachManager;
    DmrTransMutabakat   mutabakat;
    boolean ret;

    tmp.setTmpData(DmrTransferWhsTmp);
    super();
    if(box::yesno('Kayıtlar işlenecek emin misiniz?  ', DialogButton::No) == DialogButton::Yes)
    {
        ttsBegin;
        journalTable.clear();
        journalTable.JournalNameId      = DmrMutabakatTransferJournals::find().RejectJournalNameId;
        journalTableData                = JournalTableData::newTable(journalTable);
        journalTable.JournalId          = journalTableData.nextJournalId();
        journalTable.Reservation        = ItemReservation::Automatic;
        journalTable.JournalType        = InventjournalType::Transfer;
        journalTable.DmrReferanceId     = transferTable.TransferId;

        journalTableData.initFromJournalName(journalTableData.JournalStatic().findJournalName(journalTable.JournalNameId));
        journalTable.Description            = strfmt ( " %1 nolu transfer mutabakatı", transferTable.TransferId) ;
        journalTable.insert();

        while select tmp
            where tmp.Qty
        {
            ret = element.createTransferjournal(tmp.ItemId,tmp.Qty,journalTable);

            mutabakat =  DmrTransMutabakat::find(tmp.InventTransferId,tmp.ItemId,true);
            if (mutabakat && ret)
            {
                mutabakat.QtyReject = mutabakat.QtyReject + tmp.Qty;
                mutabakat.ApprovedDate = systemDateGet();
                mutabakat.TransferMutabakatStatus = DmrTransferMutabakatStatus::WaitingToReceiver;
                mutabakat.update();
            }
        }

        journalCheckPost  = InventJournalCheckPost::newPostJournal(journalTable);

        if(journalCheckPost.validate())
        {
            journalCheckPost.run();
        }
        else
        {
            throw error("Deftere nakil sırasında hata oluştu");
        }
        ttsCommit;
```

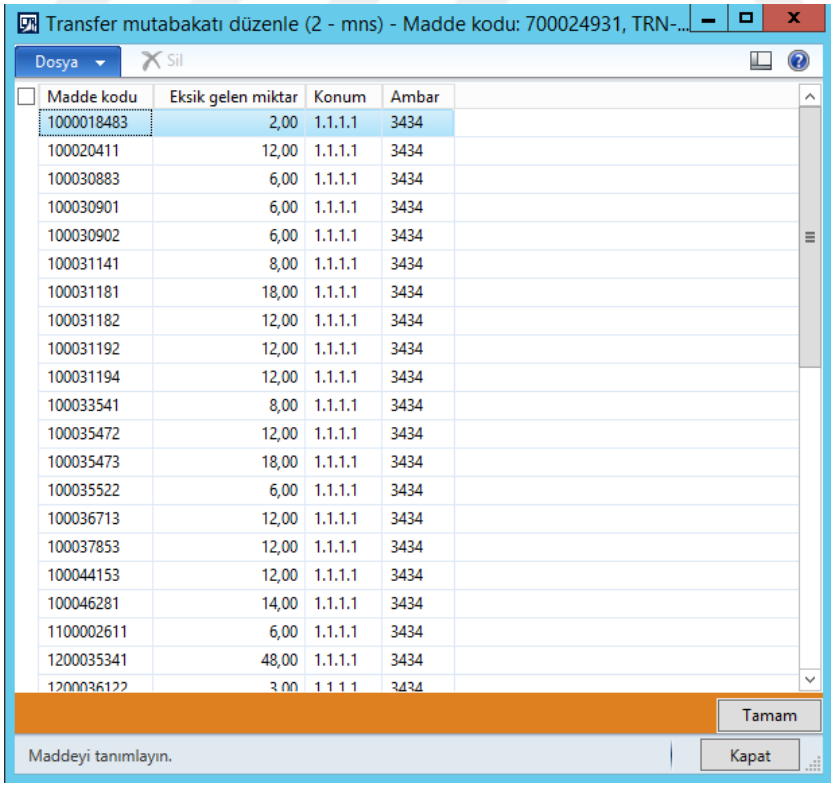
Şekil 7.8. Eksik beyanı geri çek buton fonksiyonu kod bloğu.

Şekil 7.8'de verilen kod bloğu ile kullanıcının ekranına bir messagebox çıkarttırılarak "Kayıtlar işlenecek emin misiniz?" seçeneği ile işleme devam edilip edilmeyeceği bilgisi istenmektedir. Eğer seçimi "Yes" ise işlemlere devam edilmektedir. Şekil 7.7'de belirtilen "Transfer Mutabakatı Düzenle" formunda görüntülenen her bir veri için "transfer mutabakat" günlükleri oluşturulmaktadır. "Transfer Mutabakat" tablosundaki verinin "Geri çekilen miktar" alanı hesaplamalar sonucunda güncellenir. Kaydın durumu "Alıcıda bekliyor" olarak güncellenmektedir. Günlüklerin oluşturulma işlemi tamamlandığında deftere nakil işlemi otomatik olarak gerçekleştirilmektedir.



Şekil 7.9. Eksik beyanı kabul et fonksiyonu.

Gönderici alıcının beyan ettiği eksik miktarları kabul etmesi durumunda eksik beyanı kabul et fonksiyonunu görselde belirtilen buton aracılığıyla tetikleyerek ürünlerin kendi deposunda olduğunu ve gönderimin eksik yapıldığını kabul eder.



Şekil 7.10. Eksik beyanı kabul et butonu ile açılan form.

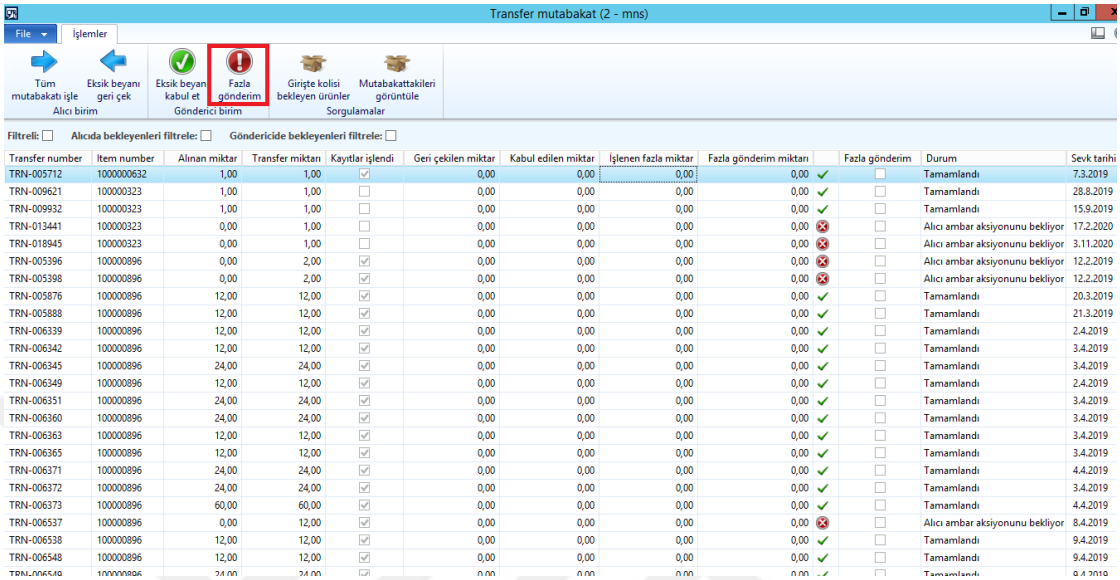
Gönderici beyan edilen eksik miktarlardan düzeltmesinin yapılması gereken ürünleri miktarlarıyla birlikte Şekilde gösterilen form üzerinde belirterek tamam butonuna tıklar. Böylece belirttiği ürünleri kendi deposuna geri almış olur.

```
public void closeOk()
{
    DmrTransferWhsTmp tmp;
    InventJournalTable      journalTable;
    InventParameters        InventParameters;
    InventjournalTableData  journalTableData;
    JournalCheckPost        journalCheckPost;
    DmrTransMutabakat       mutabakat;
    DmrTransMutabakatTmp    dmrTransMutabakatTmp;
    WHSWorkLine             WHSWorkLine;
    DmrCreateTransferJournalMutabakatMail   DmrCreateTransferJournalMutabakatMail;
    str emailBodyForBrachManager;
    boolean     ret;
    tmp.setTmpData(DmrTransferWhsTmp);

    super();
    if(box::yesno('Kayıtlar işlenecek emin misiniz?  ', DialogButton::No) == DialogButton::Yes)
    {
        ttsBegin;
        journalTable.clear();
        journalTable.JournalNameId      = DmrMutabakatTransferJournals::find().UnderReceiveJournalNameId;
        journalTableData                = JournalTableData::newTable(journalTable);
        journalTable.JournalId          = journalTableData.nextJournalId();
        journalTable.Reservation        = ItemReservation::Automatic;
        journalTable.JournalType        = InventjournalType::Transfer;
        journalTable.DmrReferanceId     = transferTable.TransferId;

        journalTableData.initFromJournalName(journalTableData.JournalStatic().findJournalName(journalTable.JournalNameId));
        journalTable.Description            = strfmt ( " %1 nolu transfer mutabakatı", transferTable.TransferId) ;
        journalTable.insert();
        while select tmp
            where tmp.Qty
        {
            ret = element.createTransferjournal(tmp.ItemId,tmp.Qty,journalTable,tmp.InventLocationId, tmp.WMSLocationId);
            mutabakat =  DmrTransMutabakat::find(tmp.InventTransferId,tmp.ItemId,true);
            if (mutabakat && ret)
            {
                mutabakat.QtyApproved = mutabakat.QtyApproved + tmp.Qty;
                mutabakat.ApprovedDate = systemDateGet();
                mutabakat.update();
            }

        }
        journalCheckPost  = InventJournalCheckPost::newPostJournal(journalTable);
        if(journalCheckPost.validate())
        {
           journalCheckPost.run();
        }
        else
        {
            throw error("Deftere nakil sırasında hata oluştu");
        }
        ttsCommit;
```

Şekil 7.11. Eksik beyanı kabul et butonu fonksiyonu kod bloğu.

Şekil 7.11'de verilen kod bloğu ile kullanıcının ekranına bir messagebox çıkarttırılarak "Kayıtlar işlenecek emin misiniz?" seçeneği ile işleme devam edilip edilmeyeceği bilgisi istenmektedir. Eğer seçimi "Yes" ise işlemlere devam edilmektedir. Şekil 7.10'da belirtilen "Transfer Mutabakatı Düzenle" formunda görüntülenen her bir veri için transfer mutabakat günlükleri oluşturulmaktadır. "Transfer Mutabakat" tablosundaki verinin "Kabul edilen miktar" alanı hesaplamalar

sonucunda güncellenir. Günlüklerin oluşturulma işlemi tamamlandığında deftere nakil işlemi otomatik olarak gerçekleştirilmektedir.



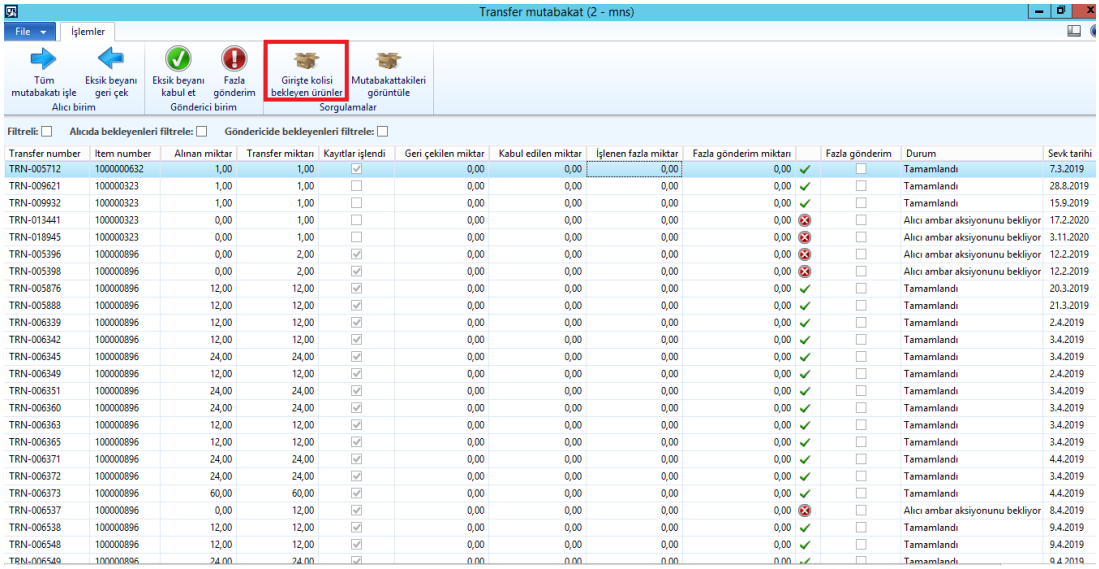
Şekil 7.12. Fazla gönderim fonksiyonu.

Alıcı kendisine gelen irsaliyeden daha fazla ürün gönderildiğini tespit ederse bunu fazla gönderim fonksiyonunu görselde belirtilen buton aracılığıyla tetikleyerek bildirir. Bu durumda göndericiye fazla gönderim yapıldığına dair mail otomatik olarak iletilir. Gönderici buna istinaden fazla gönderilen ürünler için stoklarını günceller.

```
public void closeOk()
{
    InventJournalTrans                      inventJournalTrans;
    JournalCheckPost                        journalCheckPost;
    InventJournalTable                      journalTable;
    InventJournalTableData                  journalTableData;
    int                                     i=1;
    InventDim                               FromInventDim,ToInventDim;
    FormRun                                 formRun;
    Object                                  callerDataSource;
    DmrCreateTransferJournalMutabakatMail   DmrCreateTransferJournalMutabakatMail;
    DmrTransMutabakatTmp                    dmrTransMutabakatTmp;
    WHSWorkLine                             WHSWorkLine;
    DmrTransferMutabakatHistory             dmrTransferMutabakatHistory;

    super();

    ttsBegin;

    journalTable.clear();
    journalTable.JournalNameId  = DmrMutabakatTransferJournals::find().OverReceiveJournalNameId;
    journalTableData            = JournalTableData::newTable(journalTable);
    journalTable.JournalId      = journalTableData.nextJournalId();
    journalTable.Reservation    = ItemReservation::Automatic;
    journalTable.JournalType    = InventJournalType::Transfer;
    journalTableData.initFromJournalName(journalTableData.JournalStatic().findJournalName(journalTable.journalNameId));
    journalTable.Description    = strFmt("%1 %2 fazla gönderim",dmrTransMutabakat.InventTransferId,dmrTransMutabakat.ItemId);
    journalTable.DmrReferanceId = dmrTransMutabakat.InventTransferId;
    journalTable.insert();

    inventJournalTrans.clear();
    inventJournalTrans.JournalId      = journalTable.JournalId;
    inventJournalTrans.JournalType    = InventJournalType::Transfer;
    inventJournalTrans.initFromInventJournalTable(journalTable);
    inventJournalTrans.TransDate      = systemdateget();
    inventJournalTrans.ItemId         = dmrTransMutabakat.ItemId;

    inventJournalTrans.Qty            = FromInventoryDimensionGroup_Qty.realValue() * -1;
    inventJournalTrans.LineNum = i;
    inventJournalTrans.initFromInventTable(InventTable::find(inventJournalTrans.ItemId), False, False);

    if(inventJournalTrans.validateWrite())
        inventJournalTrans.insert();
    i++;

    journalCheckPost = InventJournalCheckPost::newPostJournal(journalTable);
    if(journalCheckPost.validate())
    {
        journalCheckPost.run();
        journalTable.reread();
        if(journalTable.Posted)
        {
            dmrTransMutabakat.selectForUpdate(true);
            dmrTransMutabakat.SentExtra = NoYes::Yes;
            dmrTransMutabakat.ApprovedDate = systemDateGet();
            dmrTransMutabakat.TransferMutabakatStatus = DmrTransferMutabakatStatus::Closed;
            dmrTransMutabakat.PostedOverQty = dmrTransMutabakat.PostedOverQty + FromInventoryDimensionGroup_Qty.realValue();
            dmrTransMutabakat.update();
        }
        else
        {
            throw error("@SYS69433");
        }

    }

    ttsCommit;
```

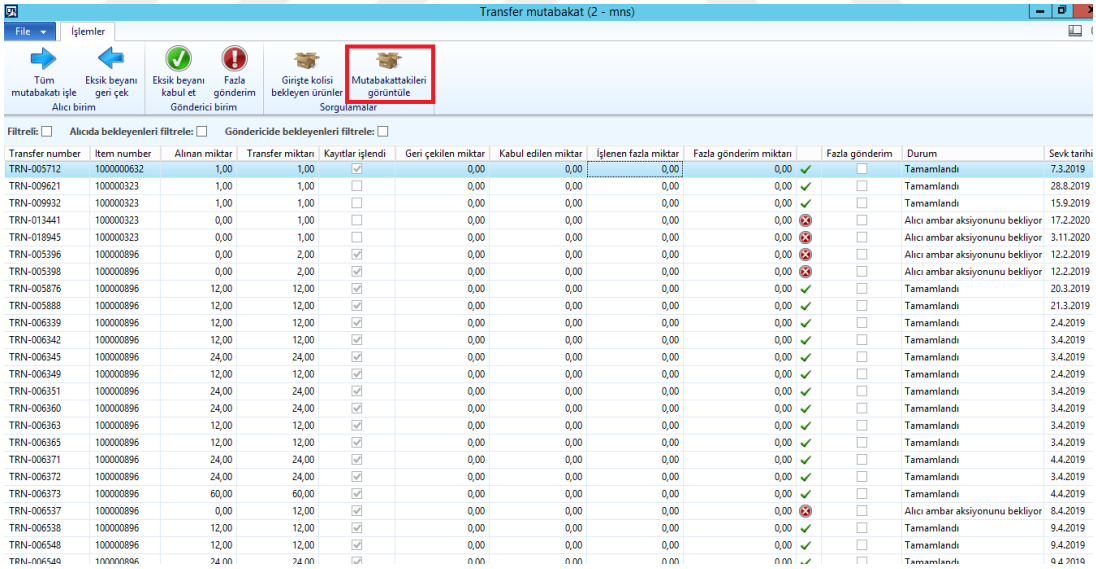
Şekil 7.13. Fazla gönderim fonksiyonu kod bloğu.

Üzerinde bulunduğu kaydın değerlerini alarak “Fazla gönderim” günlüklerini oluşturmaktadır. Oluşan günlüklerin deftere nakil işlemi otomatik olarak yapılmaktadır. “Transfer Mutabakat” tablosundaki verinin “İşlenen fazla miktar” alanı hesaplamalar sonucunda güncellenir. Durum alanı “Tamamlandı” ve Fazla gönderim alanı “Yes” yapılmaktadır.



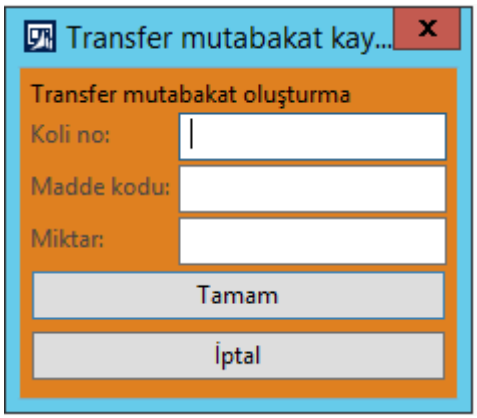
Şekil 7.14. Girişte kolisi bekleyen ürünler fonksiyonu.

Alıcı kendisine gelen ancak el terminali ile mal kabulünü yapmadığı transferleri sorgulamak için girişte kolisi bekleyen ürünler butonunu kullanır.



Şekil 7.15. Mutabakattakileri görüntüle fonksiyonu.

Gönderici ve alıcı eksik beyanların raporlamasını yapabilmek amacıyla mutabakattakileri görüntüle butonunu kullanır. Bu buton ile alıcı veya göndericinin aksiyonunu gerektiren transferler listelenir.



Şekil 7.16. Transfer mutabakatı el terminali ekranı.

Gönderici Ürünleri standart el terminali ekranıyla topladıktan sonra alıcı, firmaya özel geliştirilen transfer mutabakatı ekranını kullanarak koli,ürün ve miktar bilgilerini girer. Şekil 7.16'da belirtilen transfer mutabakat formunu kullanarak ilgili kişilere bildirim maillerinin atılmasına ve farkların mutabakat boyutunda ayrıştırılmasını sağlar.

El terminali geliştirmesi tamamen X++ sınıflar üzerinden çalışır. Yaptığımız her yeni menu item için bir sınıf yapılması gerekmektedir. Tüm menu item sınıfları "WMSWorkExecuteDisplay" ile başlamalıdır. Eğer eklediğimiz menu item bir "work" ise "WHSWorkExecuteMode" enum içine de eklenmesi gerekmektedir.

Yapılan sınıf aşağıdaki gibi tanımlanmaktadır.

Class WHSWorkExecuteDisplay DmrTransMutabakat extends WHSWorkExecuteDisplay

"WHSWorkExecuteDisplay" sınıfının "construct" metodundaki "switch/case" bloğuna aşağıdaki kod parçası eklenir.

case WHSWorkExecuteMode::DmrTransferMutabakat : return WHSWorkExecuteDisplayDmrTransMutabakat::construct();

```
container displayForm(container _con, str _buttonClicked = '')
{
    container               ret;
    container               con = _con;
    int                     hasError = 0;
    container               tmpFieldValues;
    WHSRFPassthrough        fieldValues;
    boolean                 complete;
    Str 100                 errorImput = "";
    DmrTransMutabakat       dmrTransMutabakat;
    InventTransferLine      transferLine;
    InventTransferTable     inventTransferTable;
    ;

    mode = WHSWorkExecuteMode::DmrTransferMutabakat;
    pass = WHSRFPassthrough::create(conPeek(_con, 2));

    hasError = this.hasError(_con);

    if(pass.exists(#InventLocationId) ) InventLocationId = pass.lookup(#InventLocationId);
    if(pass.exists(#InventSiteId) )     InventSiteId     = pass.lookup(#InventSiteId);
    if(pass.exists(#UserId) )           whsUserId        = pass.lookup(#UserId);

    DmrNum1 = WHSRFControlData::dmrGetControlData(con, #DmrNum1);
    if(DmrNum1 != '')
    {
        pass.insert(#DmrNum1, DmrNum1);
        InventTransferId = WHSWorkTable::DmrFindByLicensePlateForTransfer(pass.lookup(#DmrNum1)).OrderNum;
    }

    DmrNum2 = WHSRFControlData::dmrGetControlData(con, #DmrNum2);
    pass.insert(#DmrNum2, DmrNum2);
    ItemId = DmrNum2;

    DmrNum3 = WHSRFControlData::dmrGetControlData(con, #DmrNum3);
    pass.insert(#DmrNum3, DmrNum3);
    qty = str2num(DmrNum3);
,

   switch (step)
   {
       case 0:
           ret = this.buildFirstPage(ret , "Transfer mutabakat oluşturma");
           step = 1;
           break;

       case 1:
           try
           {
               [con, tmpFieldValues] = WHSRFControlData::processData(_con, pass);
               fieldValues = WHSRFPassthrough::create(tmpFieldValues);
               pass        = this.combineMaps(pass, fieldValues);
               hasError    = this.hasError(con);
               ret         = con;

               if (_buttonClicked == #RFOK)
               {
                   if(this.validateFields() && ItemId && InventTransferId )
                   {

                       select  forUpdate dmrTransMutabakat
                       where  dmrTransMutabakat.ItemId            == ItemId
                       &&     dmrTransMutabakat.InventTransferId  == InventTransferId;
                       if(dmrTransMutabakat)
                       {
                           ttsBegin;
                           dmrTransMutabakat.Qty += qty;
                           dmrTransMutabakat.SysUserId = WHSWorkUser::find(pass.lookup(#UserId)).DmrUserId;
                           dmrTransMutabakat.TransDate = today();

                           if(dmrTransMutabakat.Qty < dmrTransMutabakat.TransferQty)
                           {
                               dmrTransMutabakat.TransferMutabakatStatus = DmrTransferMutabakatStatus::WaitingToReceiver;
                           }


                          else if(dmrTransMutabakat.Qty > dmrTransMutabakat.TransferQty)
                           {
                               dmrTransMutabakat.TransferMutabakatStatus = DmrTransferMutabakatStatus::waitingToSender;
                           }
                           else if(dmrTransMutabakat.Qty == dmrTransMutabakat.TransferQty)
                           {
                               dmrTransMutabakat.TransferMutabakatStatus = DmrTransferMutabakatStatus::Closed;
                           }
                           dmrTransMutabakat.update();
                           ttsCommit;
                       }
                       else
                       {
                           inventTransferTable = InventTransferTable::find(InventTransferId);
                           dmrTransMutabakat.Qty += qty;
                           dmrTransMutabakat.SysUserId = WHSWorkUser::find(pass.lookup(#UserId)).DmrUserId;
                           dmrTransMutabakat.TransDate = today();
                           dmrTransMutabakat.InventTransferId = InventTransferId;
                           dmrTransMutabakat.ItemId  = ItemId;
                           dmrTransMutabakat.FilterCode        = inventTransferTable.InventLocationIdFrom +","+ inventTransferTable.InventLocationIdTo;
                           dmrTransMutabakat.insert();

                       }
                       pass.insert(#DmrNum2,"");
                       itemId = "";
                       ret = this.buildFirstPage(ret , "Kayıt oluşturuldu." , 0,InventTransferId );
                   }
                   else
                   {
                       ret = this.buildFirstPage(ret , errorTxt , 0,InventTransferId );
                   }

                   step = 1;
               }

           }

            catch
            {
                ret = conNull();
                ret = this.buildFirstPage(ret , "Bu kayıt zaten işlendi" , errorIndex);
                step = 1;
            }
            break;
   }

   ret = this.updateModeStepPass(ret, mode, step, pass);

   return ret;
}
```

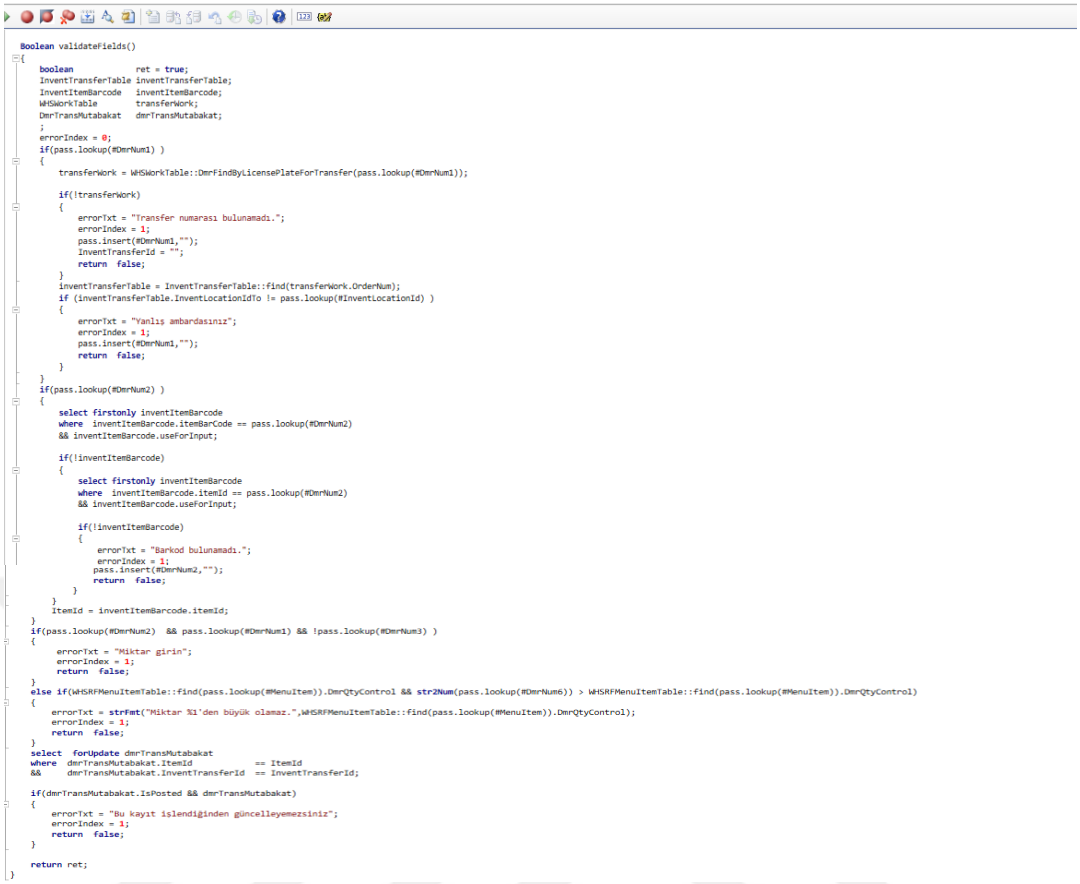
Şekil 7.17. Displayform() metot el terminali işlemleri.

“DisplayForm” metot el terminali tetiklendiğinde ilk çalışan metottur. “DisplayForm” metodu aşamalar şeklinde çalışır. Bunu step isimli değişkenle takip eder. Bizim örneğimizde step 0’da Şekil 7.16’da verilen ekran görüntüsünün kullanıcıya gelmesi için “BuildFirstPage()” metotu tetiklenmektedir. Kullanıcı el terminali ekranından gerekli bilgileri doldurduktan sonra bu metot tekrar çalışarak step 1 olarak işleme devam eder. Step 1 olduğunda “validateFields()” metotunu tetikleyerek alan kontrollerini gerçekleştirir. Girilen değerler kontrollerden doğru bir şekilde dönerse mutabakat kabul işlemleri gerçekleştirilmiş olunmaktadır. Kayıt oluşturulduğun da kullanıcıya “Kayıt oluşturuldu” bilgisi verilmektedir. Sonrasında step 1 yapılarak yeni kayıt girişi yapılması sağlanmaktadır. Aynı kayıt ikinci kez işlenmeye çalışırsa kullanıcıya “Bu kayıt zaten işlendi” bilgisi verilmektedir.

```
public container buildFirstPage( container            _con ,
                                 str 100              _error             = "",
                                 int                  _errorIndex        = 0,
                                 str 10               _TransferId       ="")
{
  container          ret  = _con ;
  ;

  ret =[this.buildControl(#RFLabel,'TransferMutabakatKaydıOlusturma',_error != "" ? _error:"Transfer mutabakat kaydı olusturma",1,'',#WHSRFUndefinedDataType,'', 0)];
  ret += [this.buildControl(#RFText, #DmrNum1, "Koli no" , 1, pass.lookupStr(#DmrNum1), extendedTypeNum(WHSLicensePlateId), "", 0, !pass.lookupStr(#DmrNum1))];
  ret += [this.buildControl(#RFText, #DmrNum2, "Madde kodu" , 1,  pass.lookupStr(#DmrNum2), extendedTypeNum(ItemId),"",0, !pass.lookupStr(#DmrNum2))];
  ret += [this.buildControl(#RFText, #DmrNum3, "Miktar", 1, "", extendedTypeNum(qty), _errorIndex == 1 ? _error : "", 0, true)];

  ret += [this.buildControl(#RFButton, #RFOK, "@SYS5473", 1, '', #WHSRFUndefinedDataType, '', 1)];
  ret = this.addCancelButton(ret, 1, true);

  return ret;
}
```

Şekil 7.18. BuildFirstPage() metotu el teminali veri giriş ekranı alanları.

“BuildFirstPage()” metotu ile Şekil 7.16’de verilen ekran görüntüsü elde edilmektedir. Örnek metotlarda DmrNum1 ile Koli no,DmrNum2 ile Madde kodu,DmrNum3 ile Miktar alanlarını ifade etmektedir.



```
Boolean validateFields()
{
    boolean             ret = true;
    InventTransferTable inventTransferTable;
    InventItemBarcode   inventItemBarcode;
    WHSWorkTable        transferWork;
    DmrTransMutabakat   dmrTransMutabakat;
    ;
    errorIndex = 0;
    if(pass.lookup(#DmrNum1) )
    {
        transferWork = WHSWorkTable::DmrFindByLicensePlateForTransfer(pass.lookup(#DmrNum1));

        if(!transferWork)
        {
            errorTxt = "Transfer numarası bulunamadı.";
            errorIndex = 1;
            pass.insert(#DmrNum1,"");
            InventTransferId = "";
            return  false;
        }
        inventTransferTable = InventTransferTable::find(transferWork.OrderNum);
        if (inventTransferTable.InventLocationIdTo != pass.lookup(#InventLocationId) )
        {
            errorTxt = "Yanlış ambardasınız";
            errorIndex = 1;
            pass.insert(#DmrNum1,"");
            return  false;
        }
    }
    if(pass.lookup(#DmrNum2) )
    {
        select firstonly inventItemBarcode
        where  inventItemBarcode.itemBarCode == pass.lookup(#DmrNum2)
        && inventItemBarcode.useForInput;

        if(!inventItemBarcode)
        {
            select firstonly inventItemBarcode
            where  inventItemBarcode.itemId == pass.lookup(#DmrNum2)
            && inventItemBarcode.useForInput;

            if(!inventItemBarcode)
            {
                errorTxt = "Barkod bulunamadı.";
                errorIndex = 1;
                pass.insert(#DmrNum2,"");
                return  false;
            }
        }
        ItemId = inventItemBarcode.itemId;
    }
    if(pass.lookup(#DmrNum2)  && pass.lookup(#DmrNum1) && !pass.lookup(#DmrNum3) )
    {
         errorTxt = "Miktar girin";
         errorIndex = 1;
         return  false;
    }
    else if(WHSRFMenuItemTable::find(pass.lookup(#MenuItem)).DmrQtyControl && str2Num(pass.lookup(#DmrNum6)) > WHSRFMenuItemTable::find(pass.lookup(#MenuItem)).DmrQtyControl)
    {
        errorTxt = strFmt("Miktar %1'den büyük olamaz.",WHSRFMenuItemTable::find(pass.lookup(#MenuItem)).DmrQtyControl);
        errorIndex = 1;
        return  false;
    }
    select  forUpdate dmrTransMutabakat
    where  dmrTransMutabakat.ItemId             == ItemId
    &&     dmrTransMutabakat.InventTransferId   == InventTransferId;

    if(dmrTransMutabakat.IsPosted && dmrTransMutabakat)
    {
        errorTxt = "Bu kayıt işlendiğinden güncelleyemezsiniz";
        errorIndex = 1;
        return  false;
    }

    return ret;
}
```

Şekil 7.19. El terminali ekranında girilen değerlerin validasyonu.

Şekilde verilen örnek kodda el terminaline girilen veriler için gerekli validasyonlar yazılmıştır. “pass.lookup(#DmrNum1)” değeri ile “Koli no” alanına girilen değer elde edilmiş olunur. Girilen değer gerekli kontrollere sokulup sonucunda hata mesajı döndürülebilir. Eğer bir hata varsa “pass.insert(#DmrNum1,"")” kod satırı ile bu alan sıfırlanır ve yeni değer girişi yapılması beklenir. Hata mesajı “errorTxt” değişkeni ile kullanıcıya iletilmektedir. “errorIndex” değişkenine 1 değeri atanarak “BuildFirstPage()” metotuna iletilir ve burada miktar alanının boşaltılması sağlanır.

BÖLÜM 8

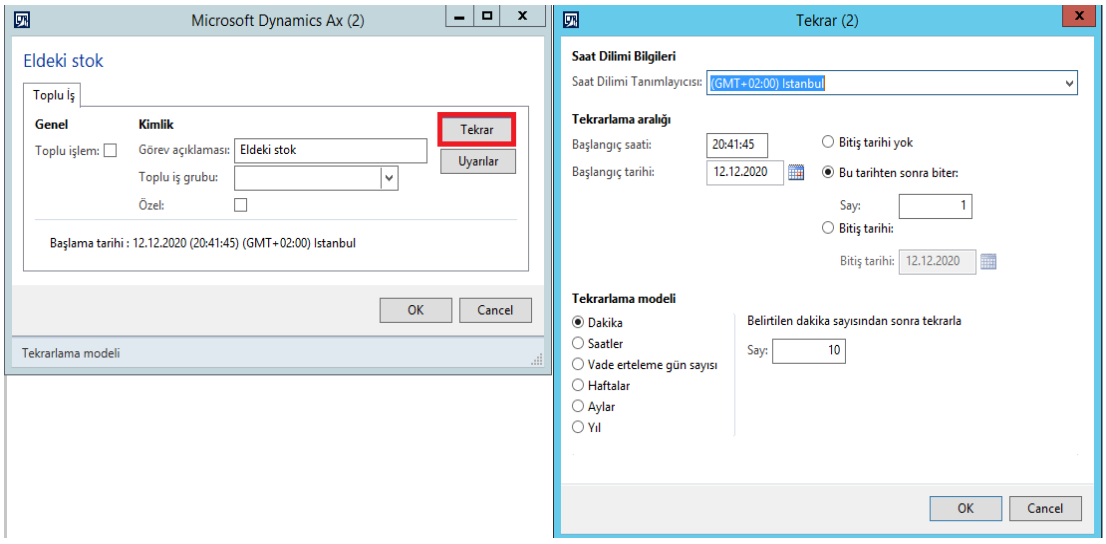
MICROSOFT DYNAMICS AX ÜZERİNDE SFTP KULLANIMI

8.1. SFTP İle Eldeki Stok Bilgisinin Gönderme Geliştirmesi

Bir lojistik şirketinin müşterisi kendisine ait ürünlerin günlük olarak lojistik deposundaki eldeki stok takibini yapmak istemektedir. Bunun için her gün bir "Toplu iş" çalıştırılarak ilgili müşteriye ait eldeki stoklar kontrol edilmekte ve bir .txt dosyası oluşturularak sftp ile ortak bir klasöre ilgili .txt bırakılmaktadır.

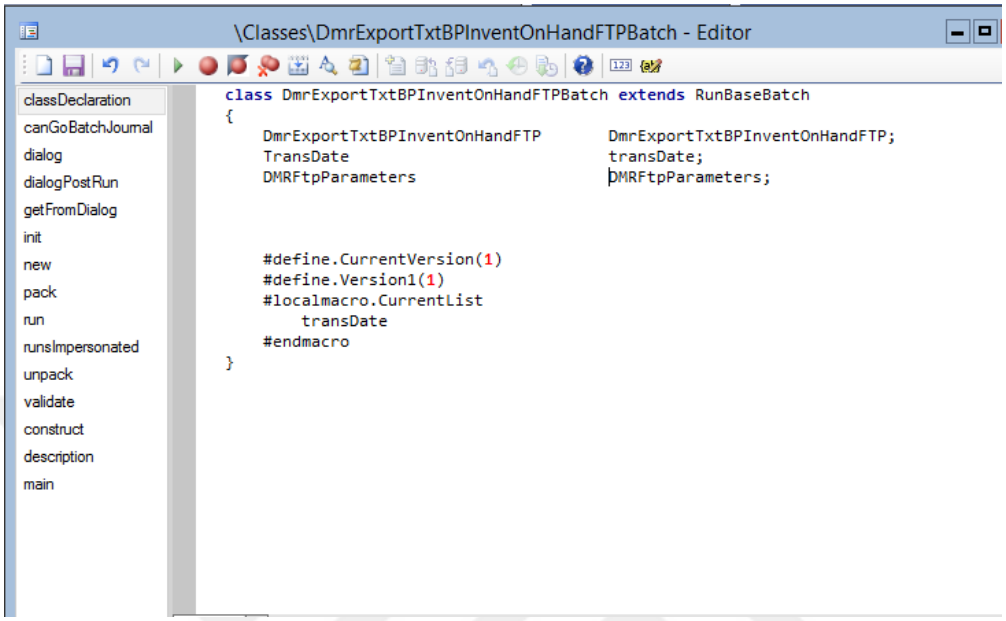
8.2. Toplu İş Yapısı

Yapılması gereken işlemleri belirli dönemlere periyodik bağlayarak sistemin otomatik gerçekleşmesini sağlamak için oluşturulan sınıflara toplu iş sınıfı denmektedir.



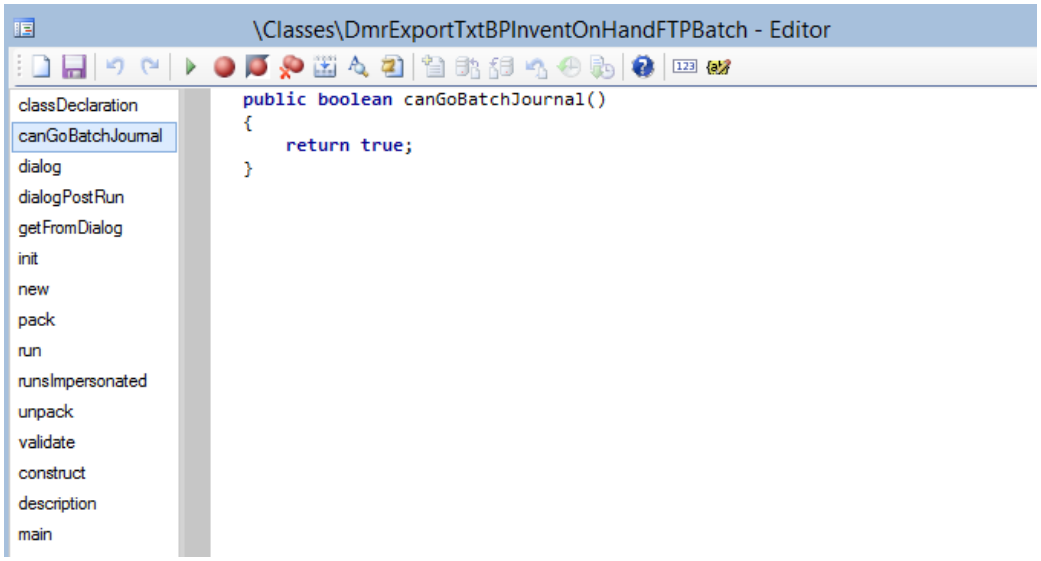
Şekil 8.1. Toplu iş sınıfı görünümü.

Kullanıcı açılan arayüz üzerinden “Tekrar” butonuna tıklayarak açılan ekrandan sınıfın çalışma periyodunu belirlemektedir.



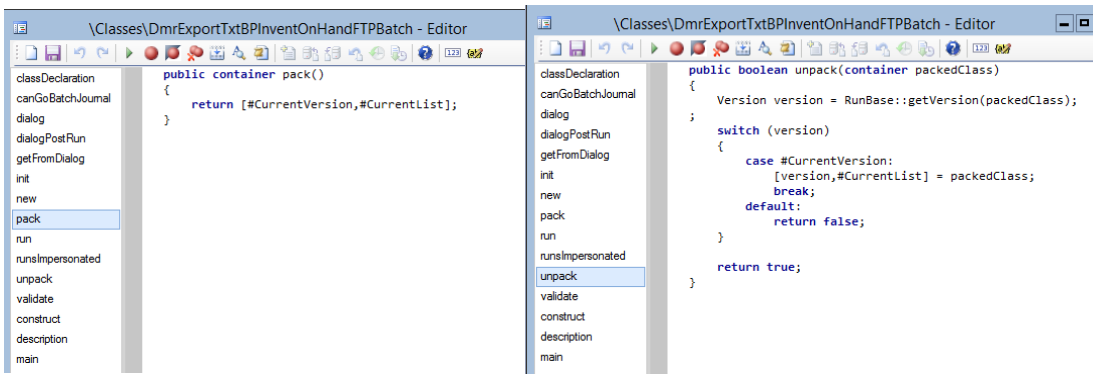
Şekil 8.2. DmrExportTxtBPInventOnHandFTPBatch Classı ClassDeclaration örneği.

Bir sınıfı toplu iş olarak çalıştırabilmek için RunBaseBatch abstract sınıfından “extend” edilmelidir.



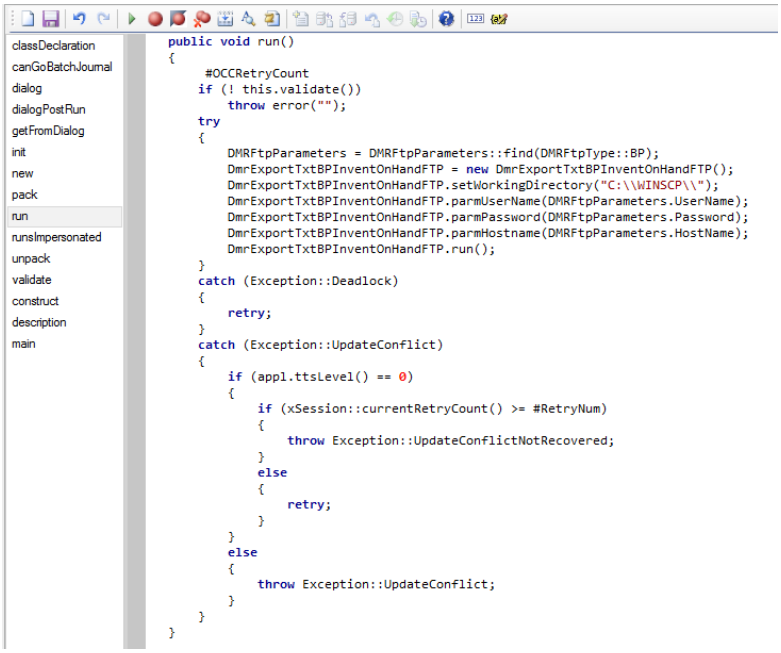
Şekil 8.3. DmrExportTxtBPInventOnHandFTPBatch Classı CanGoBatchJournal() metot.

Sınıfın Toplu Görev formunda görünüp görünmeyeceğini belirler



Şekil 8.4. DmrExportTxtBPInventOnHandFTPBatch Classı Pack() ve unpack() metot.

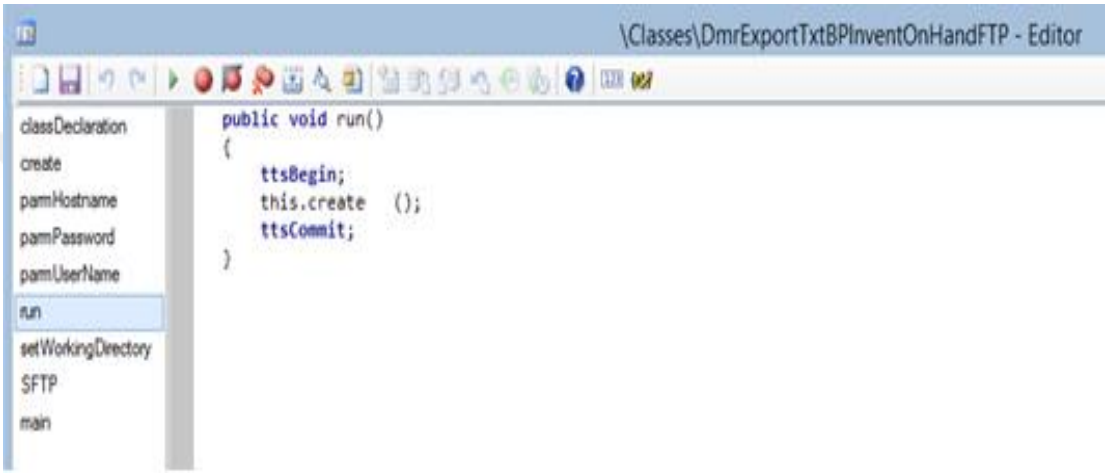
Sınıfın serileştirilmesini sağlamak için paket ve paket açma yöntemlerini de uygulaması gerekir. Bir toplu görev oluşturulduğunda, üye değişkenleri paket yöntemi kullanılarak serileştirilir ve toplu iş tablosunda saklanır. Daha sonra, toplu iş sunucusu yürütme görevi aldığında, paket açma yöntemini kullanarak sınıf üyesi değişkenlerinin serisini kaldırır. Bu nedenle, sınıf yürütme için gerekli olan değişkenlerin doğru listesini sağlamak önemlidir. Herhangi bir üye değişkeni paketlenebilir değilse, sınıf serileştirilemez ve aynı duruma kaldırılamaz.



Şekil 8.5: DmrExportTxtBPInventOnHandFTPBatch Classı Run() metot.

Toplu iş sınıfının temel mantığı **“**run()” metotunda uygulanmaktadır. Çalıştırma yöntemi, içinde tanımlanan görevi yürütmek için toplu iş çerçevesi tarafından çağrılır. X ++ kodunun çoğunu bu yöntemde çalıştırabilirsiniz. Şekil 8.5’de verilen run() metotu içerisinden sftp işleminin gerçekleşeceği sınıf çağrılmış olup çağrılan sınıfa gerekli parametreler gönderilmiştir.

8.3. WINSCP Kullanarak SFTP İşlemi



Şekil 8.6. DmrExportTxtBPInventOnHandFTP Class run() metot.

Run metot ile .txt’nin oluşturulma işlemi olan “create()” metotu tetiklenmektedir.

```
public void createCSV()
{
    TextIo                          textIOWrite;
    Filename                        scriptFileName;

    filePath = @"C:\WINSCP\BPInventOnHand\";

    try
    {
        While select inventTable
        Where inventTable.DmrFtpType == DmrFtpType::BP
        && inventTable.ItemId        == "01-01-01-06"
        {
            physicalInvent = 0;
            ReservPhysical = 0;
            AvailPhysical  = 0;

            while select dmrCustumerItemWHSControl
            group by ItemId,InventLocationId
            where  dmrCustumerItemWHSControl.ItemId       == inventTable.ItemId
            Exists join  smmBusRelSalesDistrictGroup
            where smmBusRelSalesDistrictGroup.DMRFtpType  == DmrFtpType::BP
            &&     dmrCustumerItemWHSControl.CustGroupId  == smmBusRelSalesDistrictGroup.SalesDistrictId
            {
                inventSum.clear();
                invDim.InventLocationId = dmrCustumerItemWHSControl.InventLocationId;
                invDim = InventDim::findOrCreate(invDim);
                invDimParm.initFromInventDim(InventDim::find(invDim.inventDimId));

                inventSum = InventSum::findSum(inventTable.ItemId,invDim,invDimParm);
                inventOnhand = InventOnhand::newParameters(inventTable.ItemId, invDim, invDimParm);
                physicalInvent =  physicalInvent + inventOnhand.physicalInvent();
                ReservPhysical =  ReservPhysical + inventOnhand.reservPhysical();
                AvailPhysical  =  AvailPhysical + inventOnhand.availPhysical();

               select firstOnly Amount from priceDiscTablePurch
               where priceDiscTablePurch.AccountRelation == inventTable.PrimaryVendorId
               &&    priceDiscTablePurch.ItemRelation    == inventTable.ItemId
               &&    priceDiscTablePurch.relation        == PriceType::PricePurch
               &&    priceDiscTablePurch.fromDate        <  today()
               &&    priceDiscTablePurch.ToDate          >  today();

               select firstonly priceDiscGroup
               where priceDiscGroup.DMRFtpType == DMRFtpType::BP;

               select firstOnly Amount from priceDiscTableSales
               where priceDiscTableSales.AccountRelation == priceDiscGroup.GroupId
               &&    priceDiscTableSales.ItemRelation    == inventTable.ItemId
               &&    priceDiscTableSales.relation        == PriceType::PriceSales
               &&    priceDiscTableSales.fromDate        <  today()
               &&    priceDiscTableSales.ToDate          >  today();

               select firstOnly Factor from  UnitOfMeasureConversionKoli
               where UnitOfMeasureConversionKoli.Product           == inventTable.Product
               &&    UnitOfMeasureConversionKoli.FromUnitOfMeasure == 5637145327
               &&    UnitOfMeasureConversionKoli.ToUnitOfMeasure   == 5637144624;

               flag = false;

               if(!flag)
               {
                   fileName = "ProductOnHand"+strFmt("-%1",today());
                   scriptFileName = (filePath + fileName + format);

                   if (System.IO.File::Exists(scriptFileName))
                   {
                       System.IO.File::Delete(scriptFileName);
                   }
                   flag = true;
               }
           }
```

```
            if(flag)
            {
                fileLine = strFmt("%1;%2;%3;%4;%5;%6;%7;%8;%9;%10",
                Today()-1,
                inventTable.ItemId,
                EcoResProductTranslation::findByProductLanguage(EcoResProduct::find(inventTable.Product).RecId,"en-au").name,
                physicalInvent,
                ReservPhysical,
                AvailPhysical,
                priceDiscTablePurch.Amount,
                priceDiscTableSales.Amount,
                TaxOnItem::findTaxItemGroup_IN(inventTable.salesTaxItemGroupId()).showTaxValue() ,
                UnitOfMeasureConversionKoli.Factor);

                if(!fileLineCreate)
                    fileLineCreate = fileLine;
                else
                    fileLineCreate = fileLineCreate + "\n" + fileLine ;
            }
        }
        if(fileLineCreate)
        {
            FileIOPermission = new FileIOPermission(scriptFileName, "RW");
            FileIOPermission.assert();

            textIOWrite = new TextIo(scriptFileName , "W", 20127);
            textIOWrite.write(fileLineCreate);

            CodeAccessPermission::revertAssert();
            this.SFTP(fileName);
        }

    }
    catch (Exception::CLRError)
    {
        ex = ClrInterop::getLastException();
        if (ex != null)
        {
            ex = ex.get_InnerException();
            if (ex != null)
            {
                error(ex.ToString());
            }
        }
    }
}
```

Şekil 8.7. DmrExportTxtBPInventOnHandFTP Class create() metotu.

Şekil 8.5'deki kod bloğu ile .txt de istenilen bilgilerin elde edilmesi için gerekli tablolarda işlemler yapılmaktadır. "flag" değişkeni kullanılarak "(System.IO.File::Exists(scriptFileName))" fonksiyonu ile dosya yolunda aynı isimde dosyanın mevcudiyeti kontrol edilmektedir. Eğer dosya yolunda aynı isimde dosya mevcut ise "System.IO.File::Delete(scriptFileName)" fonksiyonu ile dosya silinmektedir. "fileline" değişkeni içerisine istenilen bilgiler sırasıyla eklenmektedir.

```
if(fileLineCreate)
{
    FileIOPermission = new FileIOPermission(scriptFileName, "RW");
    FileIOPermission.assert();

    textIOWrite = new TextIo(scriptFileName , "W", 20127);

    // textIOWrite.write(headerFileLine);
    textIOWrite.write(fileLineCreate);


    CodeAccessPermission::revertAssert();
    this.SFTP(fileName);
}
```

Şekil 8.8. .txt dosyasının verilerinin yazdırılması.

Şekil 8.8'de verilen kod bloğu ile "fileLineCreate" değişkeni içerisine doldurulan verilerin istenilen dosya yolunda .txt dosyasına yazdırılması sağlanmıştır. "SFTP(fileName)" metotu ile gönderimin yapılacağı fonksiyonu tetiklemektedir.

```
public void SFTP(str _RemoteFileName)
{
    TextIo                  textIoFile;
    Set                     permissionSet;
    FileName                batchFileName, scriptFileName;
    InteropPermission       interopPerm;
    FileIoPermission        filepermission;
    str                     FTPInventOnHand;

    System.Diagnostics.Process              process;
    System.Diagnostics.ProcessStartInfo     processStartInfo;
    System.Exception ex;
    ;

    batchFileName =  _workingDirectory + 'winscp.com';
    scriptFileName = _workingDirectory + 'readfileScript.txt';

    try
    {
        FTPInventOnHand = DMRFtpParameters::find(DMRFtpType::BP).FTPInventOnHand;

        //Creating SCRIPT File
        permissionSet = new Set(Types::Class);
        permissionSet.add(new FileIoPermission(scriptFileName, "W"));
        CodeAccessPermission::assertMultiple(permissionSet);
        textIoFile = new TextIo(scriptFileName , "W", 0);
        CodeAccessPermission::revertAssert();
        textIofile.write(strFmt('open sftp://%1:%2@%3' ,userName , password,Hostname));
        textIoFile.write(strFmt('put -delete BPInventOnHand\\%1.txt /%2',_RemoteFileName,FTPInventOnHand));

        textIoFile.write('exit');
        textIoFile.finalize();

        //EXECUTE SFTP
        new InteropPermission(InteropKind::ClrInterop).assert();
        process = new System.Diagnostics.Process();
        processStartInfo = new System.Diagnostics.ProcessStartInfo();
        processStartInfo.set_FileName(batchFileName);

      processStartInfo.set_FileName(batchFileName);
      processStartInfo.set_WorkingDirectory(_workingDirectory);
      processStartInfo.set_Arguments(strFmt("/script=%1",scriptFileName));
      process.set_StartInfo(processStartInfo);
      process.Start();//this method call the SFTP
      process.WaitForExit(); //dosyayı alıyor

      CodeAccessPermission::revertAssert();

     // process.Close();
      interopPerm = new InteropPermission(InteropKind::ClrInterop);
      interopPerm.assert();

      System.IO.File::Delete(scriptFileName);

      CodeAccessPermission::revertAssert();

      files = System.IO.Directory::GetFiles(_workingDirectory, "*.txt", System.IO.SearchOption::TopDirectoryOnly);

   }
   catch (Exception::CLRError)
   {
          ex = ClrInterop::getLastException();
          if (ex != null)
          {
          ex = ex.get_InnerException();
          if (ex != null)
          {
              error(ex.ToString());
          }
      }
   }

}
```

Şekil 8.9. Winscp üzerinden oluşan .txt dosyasının gönderimi.

Toplu iş sınıfı üzerinden gelen bağlantı bilgileri ile “Winscp” programı kullanılarak dosyanın oluşturulması için kod düzenlemeleri yapılmıştır. “put -delete” fonksiyonu ile dosyayı istenilen dosya yoluna konulması ve sonrasında oluştuğu yerden silinmesi sağlanmaktadır.

BÖLÜM 9

SONUÇ

Teknolojinin hızlı gelişmesi basit anlatımı ile şirketlerin bilgiye hızlı erişmesine olanak sağlarken bir yandan rekabet ortamını ciddi derecede arttırmıştır. Aynı bilgiye aynı anda çoğu şirket reaksiyon gösterebilecek ve bulunan fırsata cevap verebilecek niteliğe erişmiştir. Bu rekabet ortamlarında şirketlerin ayakta kalabilmesini sağlayan etkilerden birisi de Yönetimsel, Stratejik ve Operasyonel kararların kısa ve uzun vadeli planlar çerçevesinde doğru ve tutarlı veriler üzerinden yapabilmesi ile mümkün olacaktır. Orta ve büyük ölçekli şirketlerin ellerindeki verileri saklayabilmesi ve okunabilirliği olan yapılar üzerinden işlemesi kaçınılmaz bir duruma gelmiştir.Bu ihtiyaç da bize ilişkisel veritabanı modeline sahip kompakt bir sistem halinde satılan ERP programlarını karşımıza çıkarmıştır. ERP programları ayrıca günümüz teknolojisine ve hızına ayak uydurabilecek yapıdaki sistemler bütününü de ifade etmektedir. ERP programlarının faydası yalnızca veri güvenliği ve erişilebilirliğin kolaylığı değildir. Ek olarak firmaların çalışanlarının da bizzat kullandığı canlı ve yaşayan sistemler olarak düşünmek de gerekmektedir.

Bu sistemler kullanıcısına vadettiği rekabet gücü ile beraber doğru ve güvenilir yapılar olarak da adlandırılmışlardır. Bir şirketin yaptığı işlerin ve yönettiği operasyonların nicelik ve nitelik değerleri artmaya başladığında çoğu zaman yönetimsel anlamda sorunları ortaya çıkmaktadır.

ERP uygulama süreci belirsizliklerin çok olduğu, zamanında ve başarılı bir şekilde tamamlanması güç bir süreçtir. Uygulama aşamaları birbirinden kesin bir biçimde ayrılmamakla birlikte bütünsel birer süreç oldukları kadar farklı şartların da geçerli olduğu niteliklere sahiptir. ERP sistemleri özünde oldukça iyi sistemler olmasına karşın, ERP sistemlerine geçmek ve bu sistemleri uygulamak pek kolay değildir. ERP

gibi bir yazılımı işletmeye kurmak, işletme için büyük bir değişimin habercisidir. Bu avantajlara karşın, başarısız bir uygulama, işletmenin verimliliğini tehlikeye sokabilir. Dolayısıyla, ERP sistemleri doğru bir şekilde yönetilmesi ve uygulanması gereken işletme için çok kritik kilometre taşlarıdır.

ERP projelerinde yazılım geliştiricileri hem destek hem de sistemin yönetilmesi konusunda aktif olarak görev alırlar. Danışman ve yazılım geliştiricisi çoğu zaman ortaklaşa çalışarak işleri tamamlar ve kullanıcılara teslim ederek talepleri yerine getirirler. Yapılan ve talep edilen işlerin bir termin tarihleri vardır ve planlar çerçevesinde bu işler tamamlanarak süreç aksatılmadan ilerletilmeye çalışılır. Şirketlerin zaman, maliyet ve kalite gibi birincil öneme sahip kavramlar çerçevesinde optimum verimlilikte işler çıkartılarak günün sonunda birbirleri ile entegre, bütünleşik ve yapı olarak güçlü sistemler ortaya konulmaktadır.

Bu çalışmamızda; şirketlerin ihtiyaç duyduğu yapıların ERP sistemlerinde bir bütün olarak sunulduğu ve istenildiği gibi özelleştirilebilir yapılar olarak karşımıza çıktığı anlatılmıştır. Şirketin büyüklüğüne, bulunduğu sektöre ve operasyon çeşitliliğine göre ERP sistemlerinin de bu uygunluklara göre seçiminden, gelişimlerinden ve sağlayacağı faydalardan söz edilmiştir. Bir yazılım geliştiricinin gözünden ERP projesinde davranılması ve yapılması gereken görevlerden, best practice olarak adlandırılan performans ve verilerin incelenmesinden, metotlar ve yapıların kullanıcı dostu olabilecek ekranlar üzerinden tasarlanması gerektiğinden bahsedilmiştir.

Son olarak; ERP Projelerinde göz ardı edilmemesi ve tartışılması gereken konulardan birisi de yöneticilerin projeye olan davranışlarıdır. Çoğu yönetici ERP projelerinin sorumluluğunu projenin başarılı veya başarısız olmasına bakmaksızın bütün yükünü Bilişim Teknolojisi departmanına bırakmaktadır. ERP projelerinde unutulmaması gereken konulardan birisi süreçleri ilgilendiren birçok konunun iş birimlerinin de katkısı ile başarılabileceği olgusudur. Üst yönetimin ve proje yöneticilerinin bu konudaki tutumları ve yönlendirmeleri projenin başarılı olmasına bir adım daha yaklaştıran öngörümler neticesinde gerçekleşecektir. Başarılı bir projenin ortaya çıkabilmesi de ancak tüm ekip üyelerinin azmi, emeği ve özverisi sonucunda gerçekleşebileceği de unutulmamalıdır.

KAYNAKLAR

Arslan, E., (2015). *Kurumsal Kaynak Planlaması (ERP) Microsoft Dynamics Ax Yazılımının Şirketlere ve Kamu Kuruluşlarına Uyarlanması*. (Yüksek Lisans Tezi). İstanbul Aydın Üniversitesi / Fen Bilimleri Enstitüsü, İstanbul.

Bayraktar, E. & Mehmet E. F. E. (2006). Kurumsal Kaynak Planlaması (Erp) Ve Yazılım Seçim Süreci. *Selçuk Üniversitesi Sosyal Bilimler Enstitüsü Dergisi*, *15*, 689-709.

Boztaş, M. (2012). *Kurumsal Kaynak Planlaması Programı Microsoft Dynamics Ax Programının Crm Modülünün Hizmet Şirketinde Uyarlanması*. (Yüksek Lisans Tezi). İstanbul Ticaret Üniversitesi / Fen Bilimleri Enstitüsü, İstanbul.

endustrimuhendisligim.com (2015). 10 Şubat 2020 tarihinde https://www.endustrimuhendisligim.com adresinden erişildi.

Sönmeztürk, G. İ. (2008). *Kurumsal Kaynak Planlamasında Başarı Faktörleri*. (Yüksek Lisans Tezi). İ.T.Ü. / Fen Bilimleri Enstitüsü, İstanbul.

Talu, Ş. (2004). Sorularla Kurumsal Kaynak Planlama (EntERPrise Resource Planning), İstanbul Ticaret Odası - İşletme Yönetiminde Yeni Eğilimler Dizisi, Yayın No: 2004-27, İstanbul.

Wikipedia, (2010). 'Microsoft Dynamics AX' http://tr.wikipedia.org/wiki/Axapta (18.05.2020)

workcube. (2018). 12 Ocak 2020 tarihinde https://www.workcube.com adresinden erişildi.

ÖZGEÇMİŞ

Ortaöğrenimini Muharrem Hasbi Anadolu Lisesinde 2012 yılında tamamladı. Lisans öğrenimini Trakya Üniversitesi Bilgisayar Mühendisliği bölümünde 2016 yılında tamamladı. Trakya Üniversitesi Bilgisayar Mühendisliği bölümünde 2017 yılında başladığı yüksek lisans öğrenimine halen devam etmektedir. 2017 yılından itibaren ERP programı (Microsoft Dynamics Ax) yazılımı geliştiricisi olarak çalışmaktadır. MB6-890 Microsoft Dynamics AX Development Introduction ve MB6-894 Development, Extensions and Deployment for Microsoft Dynamics 365 for Finance and Operations serfitikalarına sahiptir. 2020 Haziran ayında Enis STRİKO ile evlenmiştir.