Servis Tabanlı ve Tasarım Anında Çalışabilen Etkileşimli Raporlama Mimarisi

Ömer Yanar, Ökkeş Emin Balçiçek
Kuveyt Türk Katılım Bankası A.Ş, Bilgi Teknolojileri Proje Yönetimi ve AR-GE Müdürlüğü, Konya
omer.yanar@kuveytturk.com.tr, emin.balcicek@kuveytturk.com.tr

Özet: Kurumsal Raporlama Çözümleri, farklı ölçeklerdeki birçok işletmede yaygın olarak kullanılmaktadır. Ancak bu çözümler genellikle iki bileşenden oluşur: tasarım modu ve ön izleme modu. Raporlar, tasarım modunda yer imleri ve veri bandları gibi soyut rapor bileşenleri kullanılarak, çalışma anında nasıl görünecekleri bilinmeden tasarlanır. Bu yaklaşım, kullanıcıları sürekli tasarım ve ön izleme modları arasında geçiş yapmaya zorlar. Bunun yanında, son kullanıcılar genellikle raporlar üzerinde çok az kişiselleştirme seçeneklerine sahiptir çünkü tasarım modları ancak üst seviyeli bilgisayar kullanıcıları tarafından kullanılabilecek ölçüde karmaşıktır. Bu yüzden aşağıdaki çalışmada, raporların tasarlanırken aynı anda görülebileceği, "What You See Is What You Get" (WYSIWYG) olarak adlandırılan bir model sunulmuştur. Kullanıcıların son derece aşina olduğu, kelime işlemci benzeri bir ara yüz sağlanarak, raporların kullanıcıların kişisel ihtiyaçlarına göre kolaylıkla düzenlenebildiği bir uygulama geliştirilmiştir. Ayrıca rapor dosyalarının yönetimini ve istemcilere dağıtımını kolaylaştırmak için, istemci bağımsız web servis tabanlı bir raporlama sunucusu sağlanmıştır.

Anahtar Sözcükler: Raporlama Aracı, Raporlama Mimarisi, WYSIWYG Raporlama, Servis Tabanlı Rapor Üretimi, Tek Adımlı Çıktı Üretimi.

Service Based, WYSIWYG Modeled, Interactive Reporting Architecture

Abstract: Business Reporting Solutions have widespread use in most enterprises. However, a great many of these solutions are composed of two parts: a design mode and a preview mode. Reports are designed by using abstract building blocks such as placeholders and data bands in design mode without the knowledge of how they look when rendered. This approach forces users to switch between design and preview modes constantly. Besides, end users generally have no or very limited customization options because design modes are so complicated that only power users can handle. Therefore we are presenting a "What You See Is What You Get" (WYSIWYG) model in which a report can be designed and viewed at the same time. By providing a very familiar word-processor like interface, we enable end users to edit reports according to their custom needs. Moreover, we provide a client independent, web service based reporting server in order to ease report file management and client deployment.

Keywords: Reporting tool, Reporting Architecture; WYSIWYG Reporting, Service Based Reporting, One-Step Output.

1. Giriş

İş zekası ve raporlama araçları, iş dünyasında en sık kullanılan uygulamalar arasında yer almaktadır. Hali hazırda yazılım piyasasında kullanımda olan uygulamalar maliyet, entegrasyon, kullanıcı deneyimi ve sağlanan özellikler açısından değerlendirildiğinde, çok sayıda se-

çenek mevcuttur. Bu nedenle, uygulama seçiminden önce, iş ihtiyaçları doğru belirlenmeli, kullanım kolaylığı da göz önünde bulundurularak bir fayda maliyet analizi yapılmalıdır.

Yazılım piyasasında, ticari veya ücretsiz birçok raporlama uygulaması bulunmaktadır. Örneğin BIRT [1], hem iş zekası hem de raporlama özellikleri barındıran, açık kaynaklı bir raporlama sistemidir ve ücretsiz olduğundan uygun maliyetli olarak değerlendirilebilir. Ayrıca, şu anda büyük ölçüde pazar hakimiyetine sahip olan, Crystal Reports [2] ve Active Reports [3] gibi güçlü ve çok çeşitli özellikler sunan ticari uygulamalar da mevcuttur. Seçilen raporlama aracının, kendi uygulamalarınıza entegre edilmesi de ayrıca düşünülmesi gereken bir konudur. Tüm raporlama araçları belli ölçüde entegrasyon desteği sunmasına rağmen, hiçbir üçüncü parti çözüm, kendi geliştirdiğiniz bir uygulama kadar mevcut uygulamalara kolay entegre edilemez. Ayrıca genellikle üçüncü parti araçlar, son kullanıcıların rapor dokümanlarını kendi isteklerine göre düzenlemelerini engelleyen birçok kısıtlama barındırır. Kendi geliştirdiğiniz bir araçta ise, desteklenen özellikleri ve kısıtlamaları kendiniz belirleyebilirsiniz.

Sıfırdan bir raporlama aracı geliştirmek başlangıçta çok zor ve karmaşık bir iş olarak görünebilir. Piyasada birçok hazır çözüm olduğu da göz önüne alınırsa, böyle bir maliyete katlanmanın anlamsız olacağı değerlendirmesi de yapılabilir. Ancak basit bir mimari ve doğru bileşenlere sahipseniz, zor görünen bu iş basitleşmeye başlar. Buna ek olarak, kişisel bir raporlama aracına sahip olmak üretkenliği ve müşteri memnuniyetini büyük ölçüde arttırabilir. Dolayısıyla, ilerleyen satırlarda basit ama güçlü bir raporlama mimarisi sunulmuş ve bu mimariyle örnek bir uygulama geliştirilmiştir. Kurum içi geliştirilen böyle bir uygulamayla, esnek ve uygun maliyetli bir raporlama aracına sahip olmak mümkün kılınmıştır.

Bildirinin 2. bölümünde, yapılan benzer çalışmalar özetlenmiş, 3. bölümde sunulan mimariye değinilmiş, 4. bölümde ise geliştirilen örnek uygulama anlatılmıştır. 5. bölüm çalışmanın sonuçlarını ve gelecekte yapılması planlanan çalışmaları göstermektedir.

2. İlgili Çalışmalar

Raporlama araçları üzerine uzun yıllardır birçok çalışma yapılmıştır. Örneğin Shuai Hu [4] tekrar kullanılabilen rapor elemanlarının birleştirilmesi ile tasarlanabilen bir model önermiştir. Bir ara-format kullanılarak çalışma zamanında 2 aşamalı çıktı üretimi hedeflenmiştir. Ancak çalışma dahilinde geliştirilen uygulama hiyerarşik raporları desteklememektedir.

Chan [5] geleneksel şema güdümlü yaklaşım yerine, belge güdümlü bir yaklaşım benimsemiştir. Rapor modelleri SGML tabanlı bir biçim dili kullanılarak tanımlanmıştır. Veri tabanı bağımsızlığına ulaşmak amacıyla bir dönüşüm dili ve kavramsal bir mimari tasarlanmıştır.

Guillén [6] veri tabanı şemalarından otomatik web raporları üreten GARP isimli bir araç geliştirmiştir. XML ve XSL teknolojilerine dayanan bu araç, rapor üretimi için gerekli olan tüm bileşenleri barındıran JSP dosyalarından oluşmaktadır.

Tarassenko [7] son kullanıcıların SQL öğrenmelerine gerek duymadan karmaşık raporlar geliştirebileceği esnek bir model önermiştir. Ancak çalışmada da bahsedildiği üzere, bu model kullanılarak geliştirilebilecek rapor türleri sınırlıdır ve yalnızca küçük kurumlar hedeflenmiştir.

Bu bildiri ise, aynı yazarlar tarafından kaleme alınan ve daha çok sistem mimarisine değinen bir önceki bildirinin [8] devamı niteliğinde olup, çalışmanın geniş bir özetini ve dağıtım (deployment) sürecini kolaylaştırmak için mimarinin servis tabanlı bir yapıya büründürülmesini anlatmaktadır.

3. Sistem Mimarisi

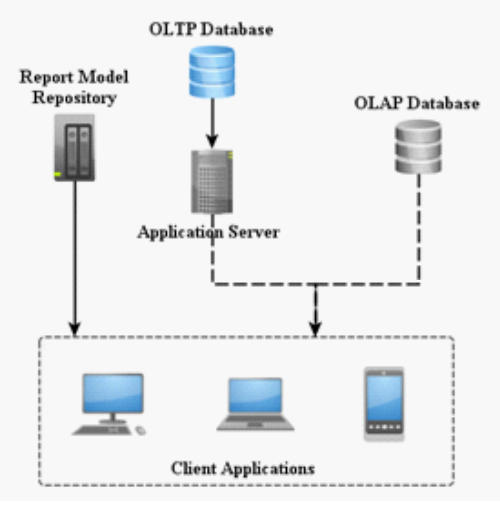
Rapor üretim süreci iki ana aşamadan oluşur: rapor düzeninin tanımlanması ve rapor içeriğinin

verilerle beslenmesi. Geleneksel yaklaşımda, öncelikle rapor tasarımı modellenir ve çalışma zamanında gerçek veriye bağlanır. Rapor tasarımı anında yalnızca kavramsal veri bulunduğundan dolayı, sonuçlar ancak rapor çalıştırılıp gerçek verilerle doldurulduktan sonra görülebilir.

Geleneksel yaklaşımın aksine, önerdiğimiz raporlama modeli gerçek verilerle tasarlanmaktadır. Bu yüzden, tasarıma başlamadan önce verilerin hazır olması gerekmektedir. Dolayısıyla öncelikle veri erişimine değinilecektir.

3.1. Veri Erişimi

Veri erişimi iki yolla sağlanmaktadır: SQL sorguları kullanılarak veya sistem ara yüzü (API) aracılığıyla doğrudan sağlanarak. Rapor tasarımı aşaması ise, verinin hangi yolla elde edildiğinden tamamen bağımsız olarak ilerleyebilmektedir.



Şekil 1. Veri Erişim Metotları

Şekil 1 veri erişim metotlarını kavramsal olarak göstermektedir. Report Model Repository, bir raporun oluşturulabilmesi için gerekli tüm bileşenleri içeren rapor dosyalarından oluşmaktadır. Sistemin çalışmakta olduğu kurumsal mimari, sunucu-istemci modeline dayanmakta ise rapor dosyaları veri erişimi için SQL sorguları, raporu şekillendirmek için ise veri tanımlayıcıları (üst veri - meta data) içerir. Eğer çok katmanlı bir mimari kullanılıyorsa rapor dosyaları sadece veri şemasını ve veri tanımlayıcılarını içerir çünkü gerçek veriler bir uygulama sunucusu tarafından sağlanacaktır.

Birinci yöntem, tüm kullanıcılarının şirket veri tabanına erişimine izin veren kurumlar tarafından kullanılabilir. Kullanıcılar, başka bir servise ihtiyaç duymayan bir istemci uygulama aracılığıyla rapor dosyalarını açar ve çalıştırırlar. Dolayısıyla raporlama süreci, kurumda yer alan başka uygulamalardan bağımsızdır. Öte yandan ikinci yöntemde kurumlar rapor istemcilerini diğer kurum içi uygulamalarına entegre edebilirler. Bu durumda istemci uygulama, rapor verisini sağlamakla yükümlü başka bir uygulama içinden çalıştırılır. Sağlanan bu veri ilgili şirkete özgü iş nesnelerinden oluşabileceği için, raporlama uygulamasının mevcut kurumsal uygulamalara entegrasyonu daha kolay ve sorunsuz olacaktır.

Yeri gelmişken, mimaride yer alan Online Transaction Processing (OLTP) ve Online Analytical Processing (OLAP) kavramlarının açıklanmasında fayda vardır. Birinci kavram genellikle canlı veri giriş çıkışının olduğu işlem (transaction) tabanlı sistemler için kullanılır. Öte yandan ikinci kavram çoğunlukla veri analizi için kullanılan veri ambarlarını ifade eder. Dolayısıyla önerilen mimarideki birinci veri erişim metodu, çok katmanlı bir mimari olsa bile OLAP için kullanılabilir çünkü kurumlar genellikle ayrı bir veri tabanı sunucusunu veri ambarı olarak kullanırlar. Bu durumda ikinci metot ise, canlı verilerin, yani OLTP'nin raporlanmasında kullanılır.

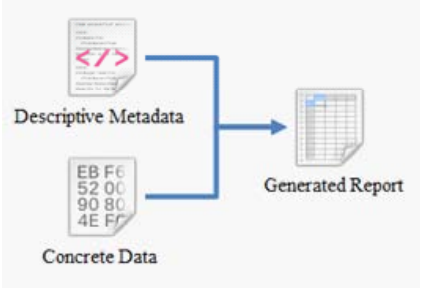
İstemci uygulama, rapor dosyası ismini sağlayarak raporlama sürecini başlatır. Eğer rapor ismiyle birlikte veri de sağlanırsa, rapor içeriği bu verilerle oluşturulur. Eğer veri sağlanmamışsa, rapor dosyası içerisindeki SQL sorguları kullanılır. Veri erişiminin başarılı olması halinde, yine rapor dosyası içerisinde yer alan veri tanımlayıcılarının rapora uygulanması süreci başlar.

3.2. Tasarım ve Düzen

Raporlama çözümleri, rapor tasarımı için genellikle kendi sistemlerine özgü bir biçimleme (markup) dili kullanırlar. 2. bölümde de belirtildiği gibi, Chan [5] rapor düzeninin tanımlanması için SGML tabanlı bir dil kullanırken Guillén [6] ise aynı amaç için XML ve XSL teknolojilerinden faydalanmıştır.

Rapor yapısı ve içerdiği veriler, öncelikle bir ara formata dönüştürülür ve daha sonra görüntüleme veya yazıcıdan çıktı almak üzere işlenirler. Dolayısıyla, raporlama süreci en az iki aşamadan oluşur: rapor modelinin ara formata dönüştürülmesi ve bu ara formatın görüntülenmek üzere okunabilir bir formata çevrilmesi. Bu yüzden Shuai Hu [4] çalışmasını Reporting Tool with Two-phase Outputs olarak adlandırmıştır.

Öte yandan bizim önerdiğimiz rapor modelinde, tek adımlı çıktı elde edebilmek için ara format kullanılmamıştır. Rapor düzeni ve içeriği, doğrudan okunabilir bir formata dönüştürülür. Örneğin geliştirdiğimiz örnek uygulamada bu amaçla Rich Text Format (RTF) kullanılmıştır. Böylece geleneksel yöntemdeki ikinci aşama, yani ara formatın okunabilir formata dönüştürülmesi işlemi saf dışı bırakılmıştır.



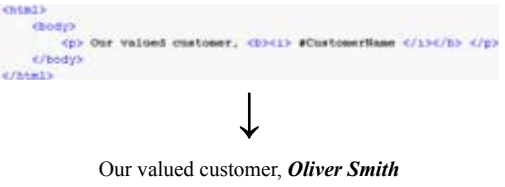
Şekil 2. Tek Adımlı Çıktı Modeli

Rapor üretim süreci, Şekil 2'de resmedilmiştir. Ara bir formatın bulunmaması, tek adımlı çıktı modelini vurgulamaktadır. Şekildeki Tanımsal Üst Veri ve Rapor Çıktısı dosyaları aslında tek bir dosyayı temsil etmektedir ve aralarında yalnızca bir fark vardır: birincisi veri yer imlerinden (data placeholder) oluşurken ikincisi gerçek veriyi içermektedir. Üst veri bilgisi son kullanıcılar açısından görünmez niteliktedir çünkü veri erişimi sağlandıktan hemen sonra yer imleri gerçek verilerle değiştirilerek rapor çıktısı elde edilir.

Rapor çıktısı, son kullanıcılar tarafından yalnızca düzenlenebilir bir metin dokümanı değil, aynı zamanda hala üretimde kullanılan veriden haberdardır. Örneğin rapor üretildikten sonra bile sıralama, gruplama, filtreleme, özet bilgi oluşturma gibi veri şekillendirme işlemleri yapılabilir. Kullanıcılar menü, buton gibi grafiksel ara yüz elemanları ile sistemle sürekli etkileşim halindedirler ve yaptıkları her işlem rapor dokümanının şekillenmesini sağlar. Bu modelle, raporların kullanıcıya göre kişiselleştirilmesi ve kullanıcılara kendi raporlarını oluşturabilme imkanı verilerek üretkenlik maksimize edilmeye çalışılmıştır. Böylece yazılım geliştiriciler, iş kuralları üzerine odaklanabilecek ve raporlama işlerini son kullanıcıya bırakabilecektir.

3.3. Veri Birleşimi

Raporlama motoru, rapor dosyası içerisinde bir yer imi ile karşılaştığında, hemen gerçek veri ile birleştirir. Yer imine uygulanan tüm biçimlendirme işlemleri, birleştirilen veriye de uygulanır. Bu işlemin nasıl gerçekleştiği Şekil 3'te gösterilmiştir. Şekildeki yer imi, ayrıştırılabilmesi amacıyla kare (#) işareti ile başlamaktadır ve bir müşteri ismine karşılık gelmektedir. Veri ile birleştirildiğinde ise, yer iminin yerinde müşteri ismi görünmektedir.



Şekil 3. Veri Birleşimi

Bu metot, Microsoft Word [9] veya Open Office Writer [10] gibi bazı kelime işlemcilerde kullanılan Mail Merge özelliğine benzetilebilir. Kul-

lanıcılar yer imleri veya parametreleri görmezler, yalnızca bir tablo sütununu tıklarlar veya bir alanı sürükleyip bırakırlar. Kelime işlemci imlecin bulunduğu yere bir yer imi yerleştirir ve veri ile doldurur. Ancak doküman kaydedildiğinde, yalnızca yer imleri dosyaya yazılır. Böylece, dokümanlar bir şablon gibi kullanılabilir ve daha sonra başka bir veri ile doldurulabilir. Ayrıca kaydedilen dokümanlar gerçek veri içermediğinden boyutları da daha küçüktür.

3.4. Farklı Kaydetme ve Yazdırma

Hangi biçimlendirme teknolojisi kullanılırsa kullanılsın, her raporlama çözümü, raporların yazdırılabileceği veya PDF gibi başka formatlarda saklanabileceği bir özellik sunmalıdır.

Önerdiğimiz raporlama mimarisi, bu konuda ciddi bir avantaja sahiptir. Çünkü okunabilir bir format kullanmaktadır ve bu format kolaylıkla diğer standart formatlara dönüştürülebilir. Örneğin, geliştirdiğimiz örnek uygulamada RTF formatı kullanılmıştır ve bu formatı PDF, HTML gibi diğer yaygın kullanılan formatlara dönüştüren birçok ticari veya açık kaynak kütüphane bulmak mümkündür.

Okunabilir bir format kullanılması, yazdırma işlemini kolaylaştırmakla kalmaz, aynı zamanda hızlandırır. Çünkü bu formatlar tasarlanırken yazdırma özelliklerinin olacağı göz önünde bulundurulmuş ve buna göre optimize edilmiştir. Dolayısıyla, dönüştürme işlemini yapacak olan kütüphanelerin yükü nispeten az olmaktadır.

3.5. Dağıtım

Kurumsal uygulamalarda karşılaşılan en büyük güçlüklerden biri de, uygulamaların son kullanıcılara dağıtım (client deployment) sürecidir. Yalnızca masaüstü platformu için geliştirilen uygulamaların dağıtımı, ancak tüm kullanıcı bilgisayarlarına kopyalanması suretiyle yapılabilir. Bu işlem hem zaman alıcıdır, hem de kurumsal ağda yüksek bant genişliği tüketimine sebep olur. Üstelik bu işlem, uygulamanın her yeni sürümünde tekrarlanacaktır. Web tabanlı uygulamalar ise, platformun neden olduğu bir takım kısıtlar nedeniyle performans ve kullanıcı deneyimi açısından masaüstü uygulamalara kıyasla daha zayıf kalmaktadır. Ayrıca web tabanlı uygulamaların geliştirilme süreci de nispeten daha uzundur.

İster masaüstü ister web tabanlı olsun, geliştirilen bir uygulamanın diğer kurum uygulamalarına entegrasyonu ayrıca ele alınması gereken bir süreçtir. Yazılım teknolojilerinde yaşanan sürekli gelişimler nedeniyle, uzun süredir çalışmakta olan kurumsal uygulamalar ile yeni geliştirilen uygulamaların birbirine entegrasyonu giderek daha da güçleşmektedir. Öte yandan kurumlar genellikle farklı ihtiyaçları için farklı uygulamalar kullanmakta, bu uygulamaların kullandıkları teknolojiler de birbirinden çok farklı olabilmektedir. Raporlama ise tüm uygulamaların ihtiyaç duyduğu ortak bir gereksinimdir. Bu nedenle geliştirilen raporlama uygulamasının farklı platformlarla kolay entegre olabilmesi gerekli ve önemli bir husustur.

Daha önceki çalışmamızda [8] raporların hem tasarlandığı hem de görüntülendiği bir uygulama geliştirilmişti. Bu model küçük ve orta ölçekli kurumlar için kullanımı ve yönetimi son derece kolay olmakla birlikte çok kullanıcılı büyük işletmeler için uygulamanın tüm kullanıcılara gönderilmesi gerekliliği nedeniyle sorun olmaktadır. Ayrıca Microsoft .NET platformunda geliştirilen bu uygulamanın, başka teknolojilerle geliştirilen diğer kurumsal uygulamalar tarafından çağırılması problemlere neden olmaktadır. Tüm bunlara ek olarak, çok kullanıcılı işletmelerde görevler ayrılığı ilkesi nedeniyle raporları tasarlayanlar ve inceleyenler genellikle farklı kullanıcılar olmaktadır. Bu nedenle yeni çalışmada, raporların sadece görüntülenebildiği web tabanlı bir rapor görüntüleyicisi geliştirilmiştir. Rapor tasarımını da sağlayan önceki uygulama, yüksek etkileşimli bir ara yüz içermesi ve rapor tasarımı yapan kullanıcıların görüntüleyenlere oranla çok daha az sayıda olması nedeniyle yine yalnızca masaüstü platformuna hizmet vermektedir.

Raporlama ihtiyacı bulunan diğer kurumsal uygulamalarla entegrasyonu kolaylaştırmak için, rapor üretim sürecinin çekirdeğini oluşturan raporlama motoru, kullanıcı ara yüzünden tamamen ayrıştırılarak web servis tabanlı bir yapıya büründürülmüştür. Böylece web servis çağrısı yapabilen ve çok farklı platformlarda (masaüstü, web, mobil vs.) çalışmakta olan herhangi bir uygulama, raporlama uygulamasından hizmet alabilmektedir.

4. Örnek Uygulama

Önerilen mimarinin uygulamaya dönüştürülmesine başlamadan önce, bir doküman formatı seçilmelidir. Bu mimarinin ana hedeflerinde biri, tasarlanan raporların etkileşimli olması, yani rapor üretildikten sonra bile düzenlenebilir olmasıdır. Bu nedenle, PDF veya PS gibi salt okunur formatlar yerine, RTF veya HTML gibi düzenlenebilir bir format tercih edilmelidir.

4.1. Doküman Formatı Seçimi

Geliştirilen örnek uygulamada RTF formatının kullanılması kararlaştırılmıştır çünkü bu format birçok kelime işlemci tarafından yaygın olarak desteklenmektedir ve farklı işletim sistemlerinde çalışabilmesi nedeniyle taşınabilir olma özelliğine sahiptir. Bunların yanında, tablo, liste, resim gibi bir rapor dokümanında bulunması gereken birçok özelliği desteklemektedir. Microsoft tarafından geliştirilen ve firmaya özgü bir format olmasına rağmen dokümantasyonu [11] tüm dünyaya açılmıştır.

```
{\rtf1\ansi{\fonttbl\f0\fswiss Times New Roman;} \f0\pard
Our valued customer, {\b \i Oliver Smith}
}
```

↓

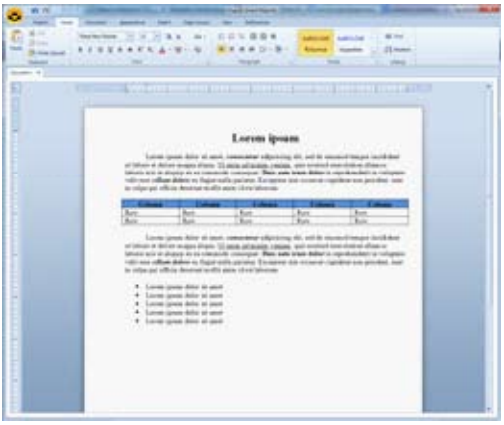
Şekil 4. Rich Text Format

Örnek bir RTF dokümanı Şekil 4'te gösterilmiştir. RTF dokümanları \rtf kontrol koduyla başlar. Biçimlendirme için birçok kontrol kodu vardır, örneğin şekilde görülen \b kodu, metnin kalın puntolarla yazılmasını sağlar. Kontrol kodları haricinde, RTF herhangi bir metin editörüyle okumaya son derece uygun bir formattır. Standart RTF dokümanları ASCII karakter kodları içerirler ancak ilgili kontrol kodları ile ASCII haricindeki karakterleri de kodlayabilirler [12].

4.2. Doküman Düzeni

Kullanılacak doküman formatına karar verildikten sonra, bu formatı işleyebilen bir metin editörü bileşenine ihtiyaç vardır. Böyle bir bileşeni kendi kaynaklarımızla geliştirmemiz mümkün olduğu halde, zaman alıcı bir işlem olduğu ve araştırmamızın kapsamı dışında kaldığı için üçüncü parti bir bileşenin kullanılmasına karar verilmiştir.

Örnek uygulamamız Microsoft .NET 4.0 platformunda geliştirilmiştir. Standart .NET metin editör bileşeni olan RichTextBox [13], temel metin işleme özellikleri sunmaktadır ancak RTF formatının sunduğu birçok özellikten yoksundur. Öte yandan çalışmada kullanılan üçüncü parti bileşen, RTF formatının geçerli son sürümünü (1.9.1) desteklemekle birlikte komut tabanlı bir uygulama geliştirme ara yüzüne sahiptir.



Şekil 5. Uygulama Ana Penceresi

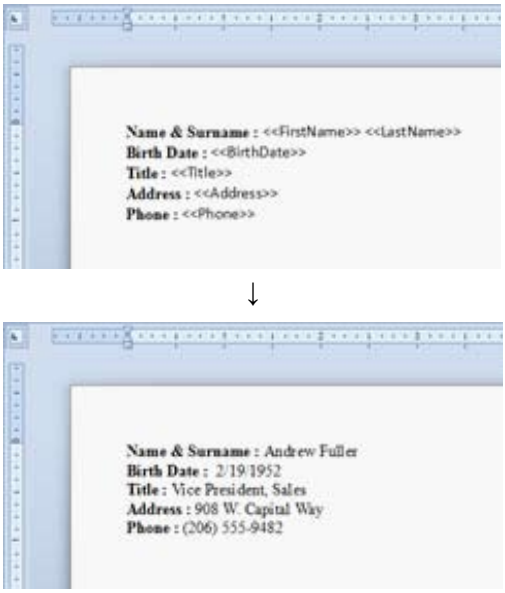
Şekil 5 uygulamanın ana penceresini göstermektedir. Şekilden de görülebileceği gibi, uygulama bir raporlama aracından çok bir kelime işlemciye benzemektedir. Uygulamanın bu şekilde tasarlanması kasıtlı olarak yapılmış ve böylece geleneksel raporlama araçlarının ne-

den olduğu uzun ve zor öğrenme süreci aşılmaya çalışılmıştır. Böylece kullanıcılar, alışık oldukları bir ara yüz sayesinde karmaşık rapor tasarımlarını zorlanmadan yapabilecektir.

Seçilen doküman formatı ve geliştirme platformu, kullanım kolaylığı ve kullanıcı deneyimini son derece etkilemesine rağmen istenilen özelliklere göre değişebilir. Bu çalışmada seçilen format ve platformun amacı mümkün olduğunca çok özellik destekleyen kapsamlı bir raporlama aracı geliştirilmesidir. Ancak gereksinimler daha dar kapsamlıysa, işlemesi daha kolay ve daha az karmaşık bir format seçilebilir.

4.3. Doküman Alanları

Şu ana kadar gelinen noktada, sıradan bir kelime işlemci görünümünde olan uygulamamız, bu bölümde eklenecek özellikler sayesinde bir rapor tasarımcısına dönüşecektir. Daha önce de belirtildiği gibi mimarimiz, önce tasarlayıp sonra çalıştırılan geleneksel yaklaşımı kullanmamaktadır. Bunun yerine, rapor tasarımı devam ederken sonuçların aynı anda görülebileceği bir yapıya ihtiyaç vardır. İşte doküman alanları da tam olarak bu amaca hizmet etmektedir.



Şekil 6. Doküman Alanları

Doküman alanları, verilere karşılık gelen yer imleri olarak düşünülebilir ve bu yer imleri rapora eklendikten sonra karşılığı oldukları veri ile değiştirilir. Temel olarak, metin editöre yapması gereken işlemi tanımlayan kontrol kodlarından oluşurlar.

Şekil 6'da bu değiştirme işlemi özetlenmiştir. Doküman alanlarının sağladığı özellikler, bu şekilde görülen basit metin değiştirme işlemleri ile sınırlı değildir. Tablolar ve listeler gibi daha karmaşık rapor elemanlarının oluşturulması için temel yapı taşlarını temsil ederler. Bir sonraki bölümde, böylesine basit bir metin değiştirme işleminden nasıl tablolu görünümler elde edileceği açıklanmıştır.

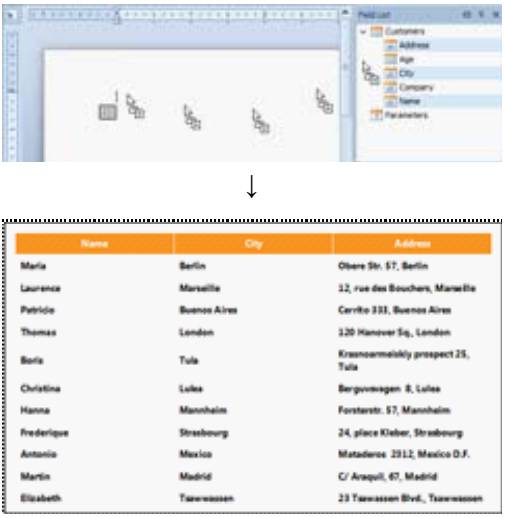
4.4. Tablo İçeren Rapor Tasarımı

Veri kümelerinin satır ve sütunlardan oluşan tablolar aracılığıyla gösterimi, birçok uygulama içerisinde son derece yaygındır. İlişkisel veri tabanı sistemlerinin (RDBMS) verileri yine 2 boyutlu tablolarda tuttuğu göz önünde bulundurulursa, bu verilerin tablo formatında gösterimi, veri analizi açısından anlaşılması kolay bir metot sunmaktadır. Bu nedenle, tablo içeren tasarımlar bir raporlama aracından beklenen en temel özelliklerdendir.

Bir önceki bölümde işlenen doküman alanları kavramı, tablolu yapılar için de bir temel model sunmaktadır. Ancak bu kez raporlama motoru, bir doküman alanıyla karşılaştığında sonraki satır ve sütunları da işlemeye devam etmelidir. Bunun yanında, tablo gösterimiyle alakalı RTF kodlarını da karşılık gelen alanlara eklemelidir.

Öncelikle, tabloda yer alması istenen alanlar sürüklenerek doküman üzerinde istenen yere bırakılır. Bu işlem, sürüklenen alan sayısıyla eşit miktarda sütun içeren bir tablo meydana getirir. Bu işlemin hemen ardından raporlama motoru, birinci satırın sütunlarına karşılık gelen değerleri işlemeye başlar, daha sonra ikinci satırın sütunları işlenir ve bu işlem başka satır kalmayıncaya kadar devam eder. Bir başka deyişle

tablo hücreleri önce sütun sütun, sonra satır satır işlenir. Bu süreç, Şekil 7'de gösterilmiştir.



Şekil 7. Tablo İçeren Raporlar

Örnekte yer alan veri kaynağının aşağıdaki şemadan oluştuğu varsayılmıştır:

Customers(Name, Company, City, Addres, Age)

Ayrıca kullanıcının koyu renkle işaretlenen alanları (Name, City, Address) sırasıyla seçtiği varsayılmıştır. Sonuç olarak dokümana üç sütundan oluşan bir tablo eklenmiş ve veriyle doldurulmuştur.

4.5. Dağıtım

3. bölümde belirtildiği üzere, raporlama uygulamasının hem dağıtımını kolaylaştırmak, hem farklı görevler üstlenen kullanıcılara farklı ara yüzlerde hizmet sunabilmek, hem de diğer kurumsal uygulamalarla entegrasyonu kolaylaştırmak için uygulama çeşitli katmanlara ayrılmıştır. Bu katmanlar ve üstlendikleri görevler Şekil 10'da gösterilmiştir.

FSR.exe: Bildiride anlatılan raporlama mimarisinin tüm bileşenlerini içeren, başka bir servis veya uygulamaya ihtiyaç duymadan çalışabilen, raporların hem tasarlanıp hem görüntülendiği masaüstü uygulamasıdır. Büyük boyutlu olması nedeniyle, çok kullanıcılı sistemler için dağıtımı zordur.

Şekil 10. Uygulama Katmanları

FSR.Viewer.exe: Raporların yalnızca görüntülenebildiği, bunun öncesinde raporun üretimi için web servise ihtiyaç duyan masaüstü uygulamasıdır. Rapor tasarımcısına göre daha kısıtlı özellikler barındırmasına rağmen boyutunun çok daha küçük olması nedeniyle dağıtımı daha kolaydır. Kendi web sunucusu olmayan kurumlar tarafından tercih edilebilir.

WebViewer.aspx: Bünyesinde kendi web sunucularını bulunduran kurumlar için ideal olan bu uygulama, masaüstü rapor görüntüleyici uygulamanın web tabanlı sürümüdür. Masaüstü karşılığı ile aynı özellikleri barındırmakla birlikte bir web tarayıcısına sahip olan her platformda çalışabilmesi ve manuel kopyalama işlemine ihtiyaç duymaması nedeniyle dağıtımı en kolay olan seçenektir.

ReportService.svc: Başta raporlama motoru olmak üzere mimarinin kullanıcı ara yüzü haricindeki tüm bileşenlerini içeren, masaüstü ve web tabanlı rapor görüntüleyicilere hizmet vermekle birlikte, web servis çağrısı yapabilen tüm uygulamalara servis sağlayan katmandır. SOAP protokolüne [13] uygun olarak geliştirilen bu uygulama, birçok farklı formatta çıktı (DOC,

DOCX, RTF, HTML, MHT vs.) üretebilmesinin yanında en sık kullanılan formatlardan biri olan PDF dokümanları için şifreleme, dokümanın tamamının veya bir kısmının kopyalanmasını ve yazdırılmasını engellemek gibi özellikler de barındırmaktadır. SOAP protokolünün en yaygın standartlardan biri olduğu göz önüne alınırsa, başka uygulamaların rapor servisiyle entegrasyonu son derece kolaydır ve mobil istemcilere de hizmet verebilmesi itibariyle geleceğe dönük kazanımlar ihtiva etmektedir.

API: Bir .NET kütüphanesi olarak geliştirilen bu uygulama rapor tasarlayıcısı, rapor görüntüleyicileri ve web servisin çağrılmasını kolaylaştıran temel fonksiyonları içermektedir. Böylece aynı platformu kullanan diğer uygulamaların rapor web servisini çağırmasına gerek yoktur, doğrudan bu API üzerinden hizmet alabilirler.

5. Sonuçlar

Bu bildiride, WYSIWYG (What You See Is What You Get) olarak adlandırılan bir raporlama mimarisi, bu mimarinin gerçeklenmesi için gerekli bileşenler ve örnek bir uygulama sunulmuştur. Benzer sistemlerle karşılaştırıldığında, sunulan modelde tasarım ve çalışma zamanlarını tek bir adımda birleştiren farklı bir yol izlenmiştir. Böylece, raporun tasarlandığı anda çalışabilmesi sağlanarak rapor hazırlama süresinin önemli ölçüde kısaltılması hedeflenmiştir. Bunlara ek olarak okunabilir doküman formatları kullanılarak sistem mimarisi basitleştirilmiştir. Dolayısıyla, raporların alışılmış kelime işlemci benzeri ara yüzlerle tasarlanması sağlanarak kullanıcıların aracı kısa sürede öğrenmeleri mümkün kılınmıştır.

Temel sistem mimarisi daha önceki bir çalışmada tasarlanan ve web platformuna taşınması gelecek çalışmalara bırakılan bir önceki bildirinin [8] devamı niteliğinde olan bu çalışmada yalnızca web tabanlı bir rapor görüntüleyicisi geliştirmekle yetinilmemiş, sistemin çekirdek bileşenleri tümüyle ayrıştırılarak web servis tabanlı bir mimariye geçilmiştir. Böylelikle raporlama ihtiyacı bulunan tüm uygulamalar web servis çağrıları yaparak ortak bir raporlama sistemine entegre olabileceklerdir.

6. Kaynaklar

[1] BIRT Technical Reference, (2013) http://www.eclipse.org/birt/phoenix/ref/

[2] SAP Crystal Reports, (2013) http://www.crystalreports.com/

[3] Active Reports, (2013) http://www.componentone.com/SuperProducts/ActiveReports/

[4] Hu, S., Zhang, J. and Li, J., "Pattern-directed Reporting Tool With Two-phase Outputs", IEEE International Conference on Software Engineering and Service Sciences (ICSESS), 23-28 (2010).

[5] Chan, D. K. C., "A Document-driven Approach to Database Report Generation", Proceedings of the Ninth International Workshop on Database and Expert Systems Applications, 925-930 (1998).

[6] Guillén, M., Vázquez , M. D. R., Sosa, V. J. and Hernández, H., "GARP: A Tool for Creating Dynamic Web Reports Using XSL and XML Technologies", Proceedings of the Fourth Mexican International Conference on Computer Science, 54-59 (2003).

[7] Tarassenko, P. F. and Bukharova, M. S., "System For Database Reports Generating", Proceedings of the Fifth Russian-Korean International Symposium on Science and Technology, Vol. 1, 84-88 (2001).

[8] Yanar, Ö. and Balçiçek, Ö.E., "Interactive Reporting Architecture: A WYSIWYG Approach to Enterprise Reporting", IEEE International Conference on Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE), 387-393 (2013).

[9] Microsoft Office, (2013) http://office.microsoft.com/en-us/word-help/use-mail-merge-to-create-and-print-letters-and-other-documents-HA101857701.aspx?CTT=1

[10] Apache OpenOffice, (2013) http://wiki.openoffice.org/wiki/Documentation/OOoAuthors_User_Manual/Writer_Guide/Using_Mail_Merge

[11] Rich Text Format Specification, (2013) http://www.microsoft.com/download/details.aspx?id=10725

[12] Microsoft Developer Network, (2013) http://msdn.microsoft.com/library/system.windows.controls.richtextbox.aspx

[13] Simple Object Access Protocol (SOAP), (2013) http://www.w3.org/TR/soap/