TC.
BEYKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
BİLGİSAYAR MÜHENDİSLİĞİ BİLİM DALI

UYGULAMA GELİŞTİRME SÜREÇLERİNDE DOKÜMAN VE GEREKSİNİM YÖNETİM ARACI

Yüksek Lisans Tezi

Tezi Hazırlayan: **Mustafa ŞEYHANOĞULLARI**

İstanbul, 2011

TC.
BEYKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
BİLGİSAYAR MÜHENDİSLİĞİ BİLİM DALI

UYGULAMA GELİŞTİRME SÜREÇLERİNDE DOKÜMAN VE GEREKSİNİM YÖNETİM ARACI

Yüksek Lisans Tezi

Tezi Hazırlayan:

Mustafa ŞEYHANOĞULLARI

Öğrenci No:

090820005

Danışman:

Yrd. Doç. Gökhan Silahtaroğlu

İstanbul, 2011

YEMİN METNİ

Yüksek lisans tezi olarak sunduğum "Uygulama Geliştirme Süreçlerinde Doküman ve Gereksinim Yönetim Aracı Yazılımı" başlıklı bu çalışmanın, bilimsel ahlak ve geleneklere uygun şekilde tarafımdan yazıldığını, yararlandığım eserlerin tamamının kaynaklarda gösterildiğini ve çalışmamın içinde kullanıldıkları heryerde bunlara atıf yapıldığını belirtir ve bunu onurumla doğrularım. 23/06/2011

Aday:Mustafa ŞEYHANOĞULLARI

UYGULAMA GELİŞTİRME SÜREÇLERİNDE DOKÜMAN VE GEREKSİNİM YÖNETİM ARACI

Tezi Hazırlayan: **Mustafa ŞEYHANOĞULLARI**

Özet

Bu tez çalışmasında uygulama geliştirme süreçlerinde gereksinim ve doküman yönetim sistemi aracı nasıl olmalıdır, ne tür fonksiyonaliteler içermelidir ve bu tür bir uygulama nasıl geliştirilmelidir konusu ele alınmaktadır. Buna göre, gereksinim ve doküman yönetiminin beraber sağlandığı bu uygulama (Raqoon), analiz, izlenebilirlik, önceliklendirme, kabul gibi gereksinim mühendiliğinin en önemli dört maddesinde büyük bir kolaylık sağlamaktadır. Raqoon kullanıcılara yalnızca zengin bir fonsiyonalite sunmakla kalmayıp ayrıca kullanıcı dostu oluşu ve sezgisel öğrenmeye açık bir uygulama oluşuylada farkını göstermektedir.

Anahtar Kelimeler: Gereksinim Yönetimi, Doküman Yönetimi, Raqoon

UYGULAMA GELİŞTİRME SÜREÇLERİNDE DOKÜMAN VE GEREKSİNİM YÖNETİM ARACI

Presented by: **Mustafa ŞEYHANOĞULLARI**

Abstract

This research thesis focuses on requirement and document management systems in application development processes. Main topics include how requirement management together with and document management systems should be, which functions are performed and how applications are developed using requirement management tools. The proposed tool, called Raqoon, is a requirement management tool integrated in and document management systems that serves four main capabilities: Analysis, Monitoring, Scalability and User Acceptance. Raqoon offers all these functionalities in a user friendly and intuitive way.

Key Words: Requirement Management,Document Management,Raqoon

İÇİNDEKİLER

Sayfa No.

ŞEKİLLER LİSTESİ

1. GİRİŞ

Gereksinim değişikliklerinin yönetilmesi süreci olarak tanımlanan yazılım gereksinim yönetim süreci, yazılım endüstrisi tarafından hızla kabul gören bir yönetim şeklidir. Yazılım gereksinim yönetimi, sistem gereksinimlerinin etkin bir şekilde kontrol edilmesini sağladığı gibi sistemdeki gereksinimlerin yine aynı sistem çevresinde oluşabilecek degişikliklerle bütünlügünün korunmasını sağlayan bir yönetim şeklidir. Gereksinim yönetiminin temel amacı, projenin müşterinin belirttiği ihtiyaçları tamamıyla karşılayacak bir biçimde oluşturulmasıdır. Bu noktada önemli olan, amacı gerçekleştirirken zaman ve maliyet açısından da en az çabanın sarf edilmesidir. Bu sürecin temel kaygıları gereksinim degişikliklerinin ve gereksinimler arası ilişkilerin yönetilmesi ve de sistem mühendisliği süreçleri dahilinde üretilen gereksinim dökümanı ve diğer dökümanlar arasındaki ilişkilerin kurulmasıdır. Bu süreç 4 başlık altında toplanabilir:

1-Analiz
2-İzlenebilirlik
3-Önceliklendirme
4-Kabul

Gereksinim yönetimi, tartışmalarla, raporlarla, iyi bir müşteri yönetimiyle birleşen ve neyin yazıya dökülüp neyin dökülemeyeceğini belirleyebilmeye olanak tanıyan, gereksinimlerin analizi, değişiklikleri, geçerlilikleri gibi konuları kapsayan bir süreçtir. Bu süreçte en önemli nokta yazılım geliştirici, kullanıcı, müşteri gibi tüm tarafların üzerinde anlaşabildiği, doğru bir yazılım gereksinimleri dökümanının hazırlanmasıdır. Bu gereksinimler dökümanı, yazılım geliştirme çabasının kapsamını, yazılım tasarımının, gerçekleştiriminin ve testinin temelini oluşturur.

Hazırlanan uygulama gereksinimlerin yönetilmesini sağladığı gibi ayrıca dokümaların saklanmasını ve yönetilmesini sağlamak gibi özelliklere sahiptir. Doküman yönetim sistemi, *"kağıtsız ofis"* bakış açısı altında, bir kurum veya organizasyon dahilinde oluşturulan ve farklı kullanıcılar tarafından kullanılan değişik

tür ve kategorideki tüm dokümanların hayat döngüleri boyunca sistematik olarak elektronik ortamda saklanması ve yönetilmesini sağlar.

Akla gelebilecek tüm iş ortamlarında bilgi, değişik tür dokümanlarda saklanır. Bu bilgiler yapılandırılmış ve yapılandırılmamış olmak üzere farklılaşan bilgi kümeleri dahilinde birikir.

Yapılandırılmış bilgi kümeleri, Kurumsal Kaynak Planlama-ERP, Müşteri İlişkileri Yönetimi-CRM, Tedarik Zinciri Yönetimi-SCM veya finansal uygulamalar tipi bütünleşik sistemler dahilinde yaratılan, işlenen ve saklanan kurumsal bilgileri içerir. Bu tarz bilgi kümeleri, veritabanları gibi formatlı yapılarda tutulur ve arayüz programları tarafından ayrıştırılarak başka sistemlere aktarılabilirler.

Yapılandırılmamış bilgi kümeleri ise E-Posta mesajları, notlar, dökümanlar gibi ortak çalışma gruplarınca yaratılan, işlenen ve saklanan bilgi kümelerini içerir. Bu tip bilgi kümelerinin bir kısmı elektronik olarak bir kısmı ise kağıt üzerinde saklanırlar.

Özellikle yapılandırılmamış bilgi kümelerinde yer alan bilgiler, dağınık ve genellikle kendini tekrarlar biçimde birikir. Bununla birlikte bunların büyük bir bölümü geçerliliğini yitirmiş, güncel olarak kullanılmayan bilgilerden oluşur. Çalışanların aradıkları bir bilgiye erişimleri güçtür ve bu durum iş süreçleri için gereken sentezlerin kısa sürede ve kolaylıkla yapılmasının önünde bir engeldir.

Döküman yönetimi ve takibinin amacı, öncelikle yapılandırılmamış bilgilerin ve dökümanların çoklu kullanıma imkan veren elektronik bir ortamda, tek noktadan, kolay erişilebilir bir biçimde kullanılmasını sağlamaktır. Birikim yönetimi felsefesi altında öncelik, şirket içi iletişim ortamında bilginin en etkin şekilde kullanımına imkan vermek, verimli bir ofis yaratmaktır. Amaç ortak çalışmalarda kullanılan tüm dökümanların kolay bulunur, kolaylıkla güncellenebilir, hızla erişilerek paylaşılabilir halde bulunmasıdır. Bununla birlikte birçok kurum basılı kağıdı en yaygın döküman aracı olarak kullanmaktadır. Bu noktada verimlilik, dökümanların kağıda basılması, kopyalanması, dosyalanarak arşivlenmesi işlemlerinde harcanan emek, zaman ve masrafların en aza indirgenmesi ile elde edilmiş olmaktadır. Verimliliği ve etkinliği arttırmanın bir sonraki aşaması, döküman yönetiminin ana iş uygulamaları ile entegrasyonudur. Yapılandırılmış bilgi kümelerinin devreye gireceği bu aşamada,

artık bilgi statik bir biçimde birikmenin ötesine geçerek, akan süreçler halinde sisteme girecektir.

2. Yazılım Mühendisliği Temel Bilgiler Ve Literatür Taraması

Günümüzde yazılım sistemleri, bankacılıktan otomotiv sanayisine, sağlık bilgi sistemlerinden şirket yönetimine, telekomünikasyon sistemlerinden hava taşımacılığına, çok geniş alanlarda kullanılan bilgisayar sistemlerinin çok önemli ve kritik bir parçasını oluşturuyor. Yazılım Mühendisliği 1968 yılında NATO tarafından gerçekleştirilen bir konferans esnasında ortaya çıkan yeni bir kavram ve yeni bir mühendislik alanı olup, yazılım sistemlerinin mühendislik prensipleri çerçevesinde tasarımı, üretimi ve işletilmesini hedeflemektedir. Bilgisayar sistemleri artık günlük hayatın her alanında yoğun ve etkin bir şekilde kullanılmakta olduğundan, Yazılım Mühendisliği tüm disiplinlerde uygulamaları olan bir alandır.

Yazılımın günümüzde hızla artan önemi, tüm dünyada yazılım mühendisliği disiplinindeki çalışmaların yoğunlaşmasına neden olmuştur. Yazılım mühendisliği disiplini 1968 yıllarından bu yana oldukça gelişme kaydetmiş; yazılım geliştirme metodolojileri, programlama paradigmaları, programlama dilleri ve çeşitli araçların geliştirilmesiyle hayli ilerleme sarfetmiştir. Bu gelişime ,**IEEE-**IEEE Computer Society ve **ACM-**Association for Computing Machinery gibi mesleki kuruluşların önemli etkisi olmuştur. Ayrıca bu kuruluşlar son yıllarda, yazılım mühendisliği çekirdek bilgisinin tanımlanması ve bu bilgilerle uyumlu yazılım mühendisliği eğitim programlarının geliştirilmesine yönelik çalışmalar da yapmaktadır. Bu bağlamda, diğer mühendislik dallarında olduğu gibi yazılım mühendisliği için de ayrı eğitim programlarının oluşturulması gündeme gelmiştir. Yazılım mühendisliği disiplinin olgunlaşma sürecinde yazılım mühendisliği eğitimi özel bir önem kazanmıştır. Özel bir önem kazanan bu eğitim programı için özel projelere başlanmıştır.

Genelde kısaltılmış adlarıyla karşımıza çıkabilecek olan bu projeler;

SWEBOK - Software Engineering Body of Knowledge : Yazılım mühendisliği çekirdek bilgisinin tanımlanması

SWCEPP - Software Engineering Code of Ethics and Professional Practice : Yazılım mühendisliği etiklerinin tanımlanması.

SWEEP - Software Engineering Education Project : SWEBOK ile uyumlu olarak örnek bir eğitim programı tanımlanması, olarak sıralanmaktadır.

Bu projeler sıralandıktan sonra birlikte çalışan IEEE ve ACM yazılım mühendisliği eğitimlerinin mufredatını tanımlamaya yönelik bir dizi çalımalar yapmış ve bu konu için araştırmalar yapması, gereklilikleri belirlemesi için bir kurul kurmuştur. Bu kurulun yaptığı araştırmalar ve ulaştığı bulgular neticesinde bir yazılım mühendisliği eğitminin müfredat açısından amaçları dolayısıyla bir bireye kazandırması amaçlanan yetenekler;

-Kullanıcı ihtiyaçlarını analiz ederek, uygun çözümler tasarlayabilmek,

-Kullanıcının belirlediği fakat genelde sürtüşmelere yol açan zaman, maliyet, kullanılabilirlik noktalarında uzlaşma sağlayabilmek,

-Mühendislik yaklaşımlarını kullanarken etik, sosyal, yasal ve ekonomik ilgileri bütünleştirecek uygun çözümler tasarlamak,

-Yazılım tasarımı, geliştirilmesi, gerçekleştirimi ve doğrulanması için bir temel sağlayan mevcut teorileri, modelleri ve teknikleri anlamak ve uygulayabilmek,

-Yazılım geliştirme ortamında etkin olarak çalışmak, gerekli olduğunda liderlik yapabilmek ve kullanıcılarla iyi iletişim kurabilmek bir diğer deyişle proje yönetimi becerisi edinmek,

-İlgili alanlardaki gelişmeleri takip ederek uygulayabilmek, şeklinde sıralanmıştır. Sonuç olarak, amaçlardan da anlaşılacağı üzere Yazılım Mühendisliği eğitiminde teknik bilgi ve beceriler yanında hukuki kavramlar, etik değerler, takım çalışması, proje yönetimi gibi soyut fakat önemli olan kavramların da kişiye kazandırılması amaçlanmıştır.

Kurulun belirlediği amaçlar doğrultusunda belirlenen Yazılım Mühendisliği Eğitimi Bilgi Alanları aşağıdaki ana başlıklar çerçevesinde detaylandırılmıştır:

Temel Bilgiler

Yazılım mühendisliğinin temelleri, yazılım mühendisliğinin ürettiği ürünlerin niteliklerini anlatan teorik ve bilimsel, matematiksel temellerden ve öngörülebilir sonuçlar üreten ana ilkelerden oluşur. Buradaki ana nokta, kaynakları belirlenmiş bir amaca dönüştürmek için mühendislik tasarımı ve mühendislik biliminin uygulanarak en uygun modellemenin yapılabilmesidir.

Profesyonel Uygulama

Teknik beceri gelişiminden çok düşünce gelişimini hedefleyen profesyonel uygulama, yazılım mühendislerinin, profesyonel ve etiğe uygun olarak uygulamam yapabilmeleri için sahip olmaları gereken bilgi, beceri ve davranışlarla ilgilidir.

İhtiyaçları Belirleme

Kullanıcı ihtiyaçlarını mevcut teknolojilerle en uyumlu biçimde bağdaştırarak çözüm tasarlayabilme.

Yazılım Tasarım

Adından da anlaşılacağı üzere teknikler, stratejiler, gösterimler ve desenlerle ilgilidir. Tasarım, kaynaklar, performans, güvenilirlik ve güvenlik gibi kısıtlamalar gözönüne alınarak işlevsel gereksinimlere uygun olmalıdır. Ayrıca, yazılım bileşenleri arasındaki içsel arayüzler, mimari tasarım, veri tasarımı, kullanıcı arayüzü tasarımı, tasarım araçları ve tasarımın değerlendirilmesi de bu alanın kapsamındadır.

Yazılım Oluşturma

Tasarımı belirlenmiş yazılımın dökümantasyonu aşamasıdır. İşleyiş, teknik vs. açısından bilgileri içerir.

Yazılım Gelişimi

Yazılımın kullanıma başlanmasından sonra yazılımın desteklenmesi sürecini kapsar. Yazılımın eksiklerinin giderilmesi, test edilmesi,iyileştirilmesi gibi aşamaları içeren bu aşama maliyet gerektiren önemli bir aşamadır.

Yazılım Kalitesi

Yazılımın kalite nitelikleri, kullanılabilirlik, güvenilebilirlik, güvenlik, bakıma uygunluk, esneklik, etkinlik ve performans gibi kriterleri kapsamaktadır.

Yazılım Yönetimi

Yazılımın kullanımından sonra etkin bir şekilde devam edebilmesi, varlığını sürdürebilmesi için uygulama alanındaki tüm aşamaların izlenmesi ve kontrol edilmesini kapsar. Yazılım geliştirme projelerinin başarısı, farklı kollardaki işlerin koordinasyonu, yazılım versiyonlarının bakımı, kaynakların gerekli oldukları zaman var olabilmesi, projedeki işlerin uygun olarak bölünebilmesi, iletişimin kolaylaşması için kritik önemdedir.

Yazılım mühendisliğinin gerektirdiği nitelikler olarak belirtebileceğimiz bu tanımlar ülkemizde de yer alan lisans ve yüksek lisans yazılım mühendisliği eğitimlerinde müfredatın belirlenmesinde önemli bir rol oynamaktadır.

2.1. Endüstri Uygulamaları

2.1.1. Relational Doors

IBM® Rational® DOORS® yazılımı, gereksinimlerin duyurulmasını, işbirliğini ve doğrulamayı iyileştirmenize olanak sağlayarak maliyetleri düşmesine, verimliliği artmasına ve kalitenin yükselmesine yardımcı olan lider bir gereksinim yönetimi uygulamasıdır.

- Kapsamlı bir gereksinim yönetimi ortamı sağlar.
- Gereksinim veritabanına web tarayıcı erişimi ve gereksinim tanımlama yetenekleriyle bütünleştirme sağlayarak işbirliğine dayalı gereksinim sürecine tüm paydaşları etkin şekilde dahil eder.
- Rational' ürününün değişiklik yönetimi çözümleriyle bütünleştirir.
- Sağlayıcıların ve geliştirme ortaklarının doğrudan geliştirme sürecine katılmasını sağlar.
- Kolay ve güçlü izlenebilirlik için, gereksinimler ile tasarım öğeleri, test planları, test örnekleri ve diğer gereksinimler arasında bağlantı kurar.
- Değişen gereksinim yönetimi gereksinimlerine göre ölçeklenir.
- DOORS Discussions ile resmi olmayan gereksinim tartışmalarına olanak sağlar.
- Test yapanların, gereksinimler ile test örnekleri arasında bağlantı kurması için el ile gerçekleştirilen test ortamlarına yönelik Test İzleme Araç Seti içerir.

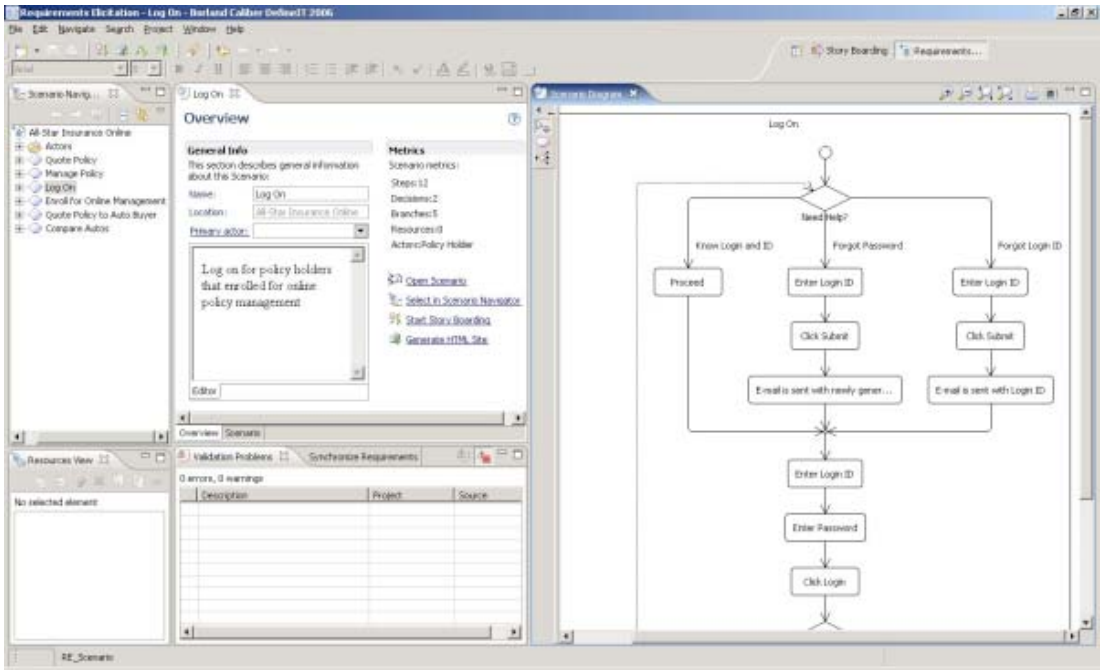
- Sistemleri ve yazılım yaşam çevrimi araçlarını bütünleştirmeye yönelik genel ve açık bir yol sağlayan, gereksinim yönetimi, değişiklik yönetimi ve kalite yönetimi için Open Services for Lifecycle Collaboration (OSLC) belirtimlerini destekler.

2.1.2. Borland

Tam entegre, maliyet etkin bir Yazılım Gereksinimleri Tanımlama ve Yönetim Sistemi olarak hem dünyada hem de ülkemizde pek çok önemli yazılım projesinde tercih edilen bir üründür. Borland Caliber ile elde edilebilecek avantajlar şu şekilde açıklanmaktadır:

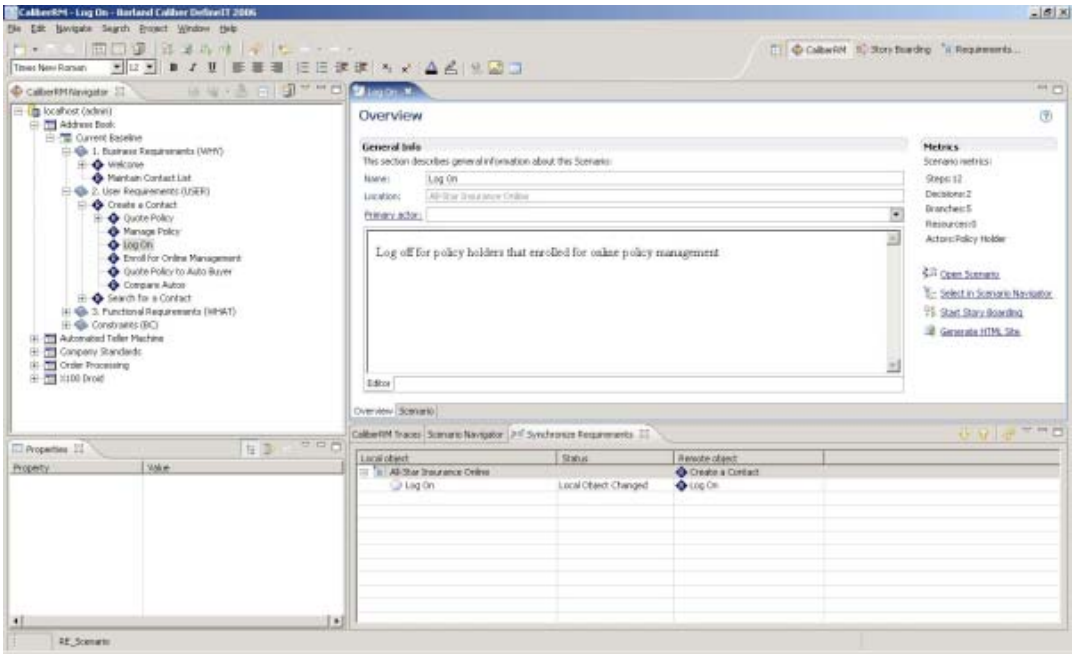
- Ortak İş Senaryosu yöntemiyle ekip iletişimini etkinleştirerek gereksinimlerin en başta doğru tanımlamasının sağlanması ve böylece yanlış anlamalardan ve iletişim sorunlarından kaynaklanan hatalı gereksinimlerin gereksiz iş tekrarına neden olmasının önlenmesi,
- Gereksinimlere ilişkin toplanan bilgi, tablo, ekran çıktıları ve gerekli olan diğer kaynakların iş senaryosuna dahil edilebilmesi,
- Tanımlanan gereksinimlerin kusursuz gereksinim entegrasyonuyla yazılım mimarisi/ tasarımı, yazılım test yönetimi, kalite yönetimi ve iş yönetimi gibi diğer iş alanlarına yönlendirilebilmesi ve böylece gereksinimlerin ilgili bütün süreçlerde izlenebilmesi,
- Etkin değişiklik yönetimi ile planlamanın iyileştirilebilmesi,
- Gereksinimlerle kodlar ve testler arasında izlenebilirlik sağlanması,
- Proje aşamalarındaki ilerlemelerin kolayca izlenebilmesi,
- Test elemanlarının tanımlı gereksinimlerden yola çıkarak test senaryolarını kolay ve hızlı bir şekilde hazırlayabilmesi, ve böylece etkin test yönetimi için pratik ve güvenli bir gereksinim altyapısının hazırlanması,
- Diğer Borland ürünleri veya üçüncü parti ürünlerle entegre çalışılabilmesi,
- Projenin herhangi bir anında gereksinimlerle ilgili rapor ve metriklerin kolayca elde edilebilmesi,
- Gereksinim yönetim sürecine hız ve çeviklik kazandırılması,
- Baştan sona etki analizinin yani görsel izlenebilirlik yöntemleri ile kullanıcıların tek bir değişikliğin yol açacağı tüm etkiyi analiz edebilmelerinin sağlanması,
- Gereksinim değişikliklerini yönetme kabiliyetinin artması,
- Müşterinin artan ve değişen taleplerine cevap verilebilirliğin artması,

Firma; projelerde müşteri gereksinimlerinin doğru şekilde anlaşılabilmesi, müşterilerin projelerde istedikleri şeyleri yapılıp yapılmadığının tespiti, müşteriler ile teknik çalışanların aynı dilden konuşabilmesi, gereksinimlerin kullanıcıların da anlayabileceği şekilde görsel olarak tanımlanabilmesi, senaryoların basit akışlar şeklinde gösterilebilmesi ve böylece müşteriler ve yazılımcıların ne yapılacağı konusunda daha kolay bir şekilde anlaşabilmeleri gibi durumları bu program aracılığıyla sağlamaya çalışmışlardır.



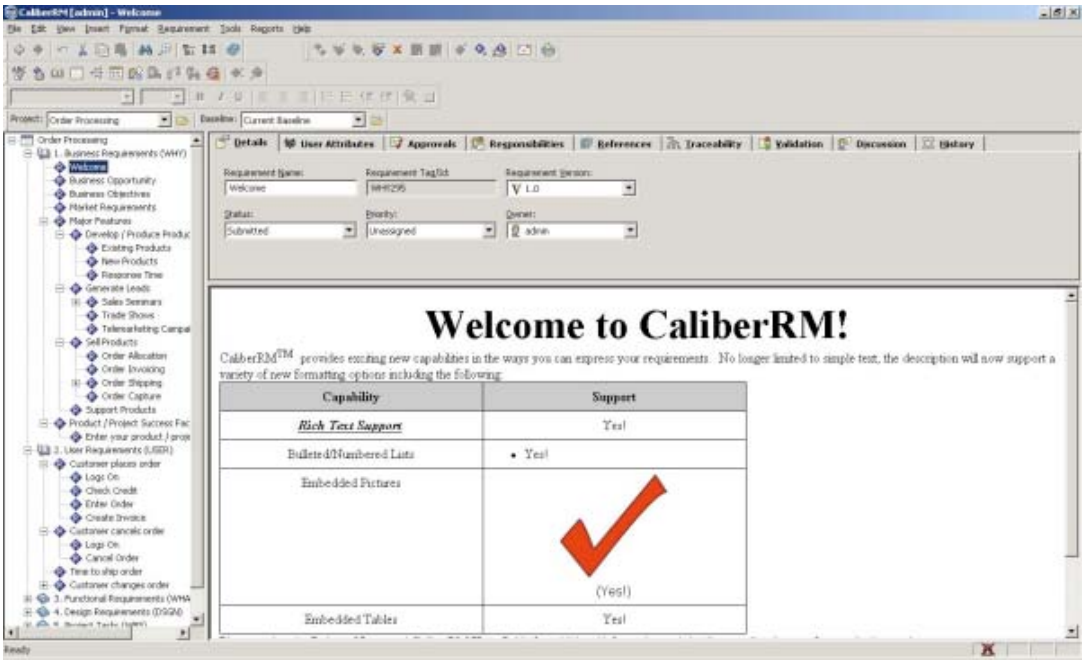
Şekil. 1. Borland Requirements Elotation Örnek Ekran Görüntüsü

Firma aynı zamanda, kullanıcı senaryolarının tanımlanması, saklanması, ekip ile paylaşılması, kullanıcı senaryolarının görsel olarak oluşturulabilmesi, müşterilerden onay alındıktan sonra Borland CaliberRM ile tüm çalışanlara paylaştırılabildiğini ve merkezi olarak saklanabildiğini iletmektedir. Borland Caliber DefineIT'in sahip olduğu Eclipse arayüzünde yer alan Borland CaliberRM ile tüm bu işlevlerin kolayca yapılabildiği görülmektedir.



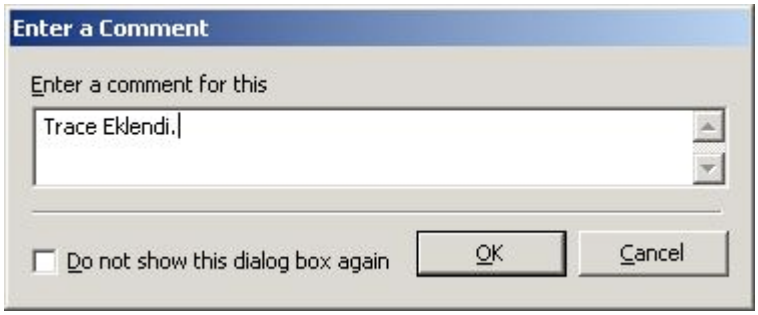
Şekil. 2. Borland CaliberRM Örnek Ekran Görüntüsü

Gereksinim analizlerinin hazırlanması ve gereksinimlerdeki değişikliklerin takip edilmesi, gereksinim dokümanlarının son sürümlerinin kolayca takip edilebilmesi ve gereksinimlerin doküman karmaşasına girmeden sade bir arayüz üzerinden tüm ekip ile paylaşılabilmesi sağlanmıştır.



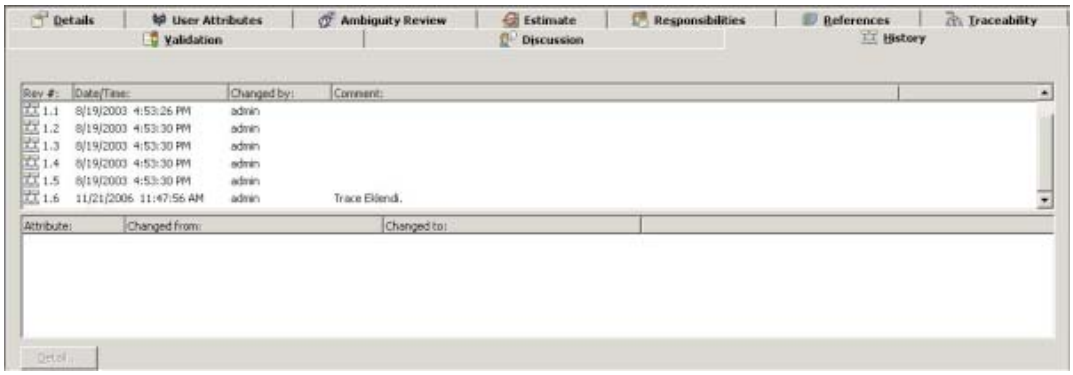
Şekil. 3. Borland CaliberRM Örnek Ekran Görüntüsü

Borland CaliberRM ile gereksinimler üzerinde yapılan tüm değişikliklerin kim tarafından, ne zaman ve neden yapıldığı gereksinimler ile birlikte saklanabilmekte ve gerektiğinde bu bilgilere hızla ulaşılabilinmektedir.



Şekil. 4. Yorum Ekleme Ekranı

Gereksinimlerin analiz süreci içerisindeki değişiminin yönetimi, projelerin kritik anlarında gereksinimlerin dondurulabilmesi ve gereksinim havuzu içerisindeki her bir gereksinimin tarihçesi görüntülenebilmektedir.



Şekil. 5. Tarihçe Ekran Görüntüsü

Gereksinimlerin yazılım sürecinde ilerleyen adımlarla ilişkilendirilmesi, testlerin hangi gereksinimleri kapsadığı, açıkta kalan gereksinimlerin, gereksiz yere karşılanan işlevselliklerin belirlenebilmesi, kaynak kod, UML model, değişiklik isteği, doküman, yazılım test tanımı gibi yazılım süreçlerinin diğer ürünleri arasındaki izlenebilirlik ilişkilerinin kurulabilmesi, süreçlerin görülebilir ve izlenebilir olması sağlanmıştır.

3. Yapılan Çalışma

3.1. Amaç

Uygulamamız doküman ve gereksinim yönetimi uygulamalarını birleştirerek yazılım geliştirme süreçlerlerinde kullanıcılara ve firmalar kolaylık sağlamak amacıyla geliştirilmiştir. Yapılan uygulamada bir yazılım projesi için oluşturulacak tüm dokümanlar hazırlanabilir. Bu dokümanlar kapsam, analiz, test senaryosu, bulgu formu ve kullanıcı kabul dokümanlarıdır. Tabi hazırlamış olduğumuz uygulama yanlızca bu dokümanları hazırlamak için yapılmadı. Ayrıca bir kurumun veya firmanın isteyeceği tüm dokümanları hazırlama kabiliyetine sahiptir.

Uygulama geliştirme süreçlerinde firmaların en önemli ihtiyaçlarından biri süreç dokümalarının kolay hazırlanması, izlenebilmesi ve değiştirilebilmesidir. Bu dokümanlar hazılanırkende kuruma özel bir doküman kütüphanesinin oluşmasının sağlanmasıdır. Bu kütüphane günümüz firmaları için vazgeçilmez bir kaynaktır. Bunun en önemli nedenlerinden biri kurum içi sirkülasyonlarda yeni gelecek insan kaynakları için kolaylıkla erişebilecekleri bir ortamın olması ve bu sayade kurum bilgi birikiminin yeni gelen kaynalara kolaylıkla aktarılmasıdır.

3.2. Kısıtlar Ve Ortam

Yapılan uygulamada geliştirme ortamı olarak Microsoft Visual Studio .Net kullanılmıştır. Uygulama tamamiyle c# dilinde geliştirilmiş veri tabanı olarakta Oracle 10g veritabanı kullanılmıştır. Uygulama geliştirilirken kulanılan dil ve araçlar nedeniyle çalışma ortamı olarak yalnızca Microsoft Windows işletim sistemi kullanılmalıdır.Veri tabanı seviyesinde bu tür bir kısıt yoktur. Piyasada bulunan tüm işletim sistemlerinde çalışacak bir versiyonu bulunmaktadır.

3.3. Uygulama

Yapılan uygulama, uygulama geliştirme süreçlerinde ihtiyaç duyulan tüm doküman ve gereksinimlerin oluşturulması ve yönetilmesini sağlamak amacıyla kullanılır. Detaylarını aşağaıda anlatacağım gibi uygulamamız oluşturulacak tüm dokümanların öncelikle şablonlarının oluşturulmasını ve daha sonra oluşturulan şablona disiplininden çıklımadan dokümanların oluşturulmasını sağlar. Oluşturulan bu dokümanlara gereksinimler kolaylıkla tanımlanabilir ve gerekirse başka kullanıcılar veya süreç çersinde bulunan diğer paydaşlar tarfından rahatlıkla değiştirilebiir ve yönetilebilir. Tabiki bu yönetim ve değişikliği yapabilmek için kullanıcıların yeterli yetkiye sahip olmaları gerekmektedir. Yine uygulamammızın en önemli özelliklerinden biri web servisler yardımıyla başka uygulamalar ile rahatlıkla entegra olabilmektedir.

3.4. Uygulama Detayları

3.4.1. Uygulama Yazılım Mimarisi

Ugulama çok katmanlı bir mimari yapıda hazırlanmıştır. Ekran katmanı uygulama sunucusu katmanı ile .net remoting teknolojisi kullanarak haberleşmektedir. Uygulama sunucusu katmanında assemblyler iis (internet information services) tarafından host edilmektedir.

Internet Information Services web sayfalarının yayınlanmasını ve web uygulamalarının çalışmasını sağlayan, istemcilerden HTTP ve FTP üzerinden gelen talepleri Microsoft Windows sunucu tabanlı işletim sistemlerinde karşılayan birim IIS'tir.

Windows Sunucu işletim sistemlerinin en önemli parçalarından birisi olan IIS, HTTP ve FTP protokollerini başarılı bir şekilde kullanarak önemli bir görevi yerine getiren bir sistemdir.

Bir ağ mühendisinin gözüyle IIS, OSI katmanının bir üstünde yer alarak herhangi bir bilgisayar ile bir Windows bilgisayar arasında oturumsuz bir protokol sağlar. Buradaki oturumsuz kelimesi ile anlatmak istediğim, Telnet gibi protokollerde olduğu gibi iki sistem arasında devam eden bir konuşma bulunmamaktadır. İstemci – Sunucu mimarisi doğrultusunda, istemci HTTP

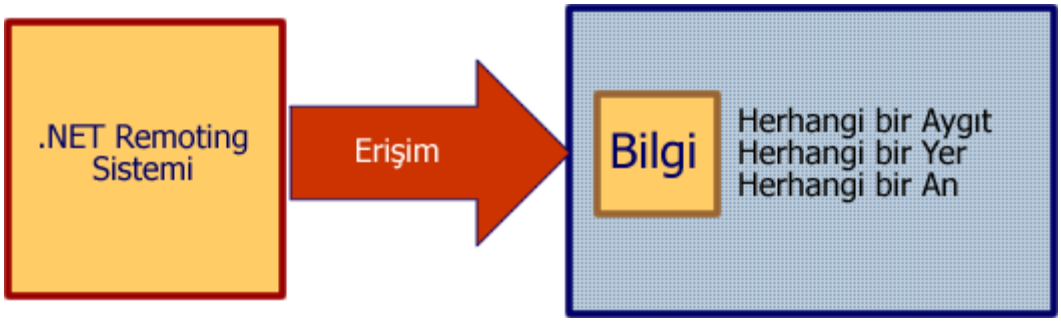
protokolü üzerinden sunucuya talepte bulunduğunda, ki burada sunucu tarafında istemciyi IIS karşılamaktadır, IIS ya cevap döner ya da dönmez. IIS aynı zamanda FTP Server olarak da kullanılabilir.

3.4.2. .Net Remoting

Remoting sistemi, kısaca, farklı platformlarda çalışan uygulamalar arasında veri alışverişine imkan sağlayan bir sistemdir. Bu tanımda söz konusu olan platformlar farklı işletim sistemlerinin yer aldığı farklı ve birbirlerinden habersiz proseslerde çalışan uygulamaları içerebilir. Olayın en kilit noktasıda, farklı sistemlerin veri alışverişinde bulunabilmelerinin sağlanmasıdır.

Konuyu daha iyi irdeleyebilmek amacıyla, Microsoft tarafından belirtilen şu örneği göz önüne alabiliriz. Bir pocket pc aygıtında, Windows CE işletim sistem üzerinde, C# dili ile yazılmış bir uygulamamız olduğunu düşünelim. Bu uygulama herhangibir parçası olmadığı bir ağda yer alan bir uygulamanın ilgili metodlarını çağırıyor olsun. Uzak network üzerinde kurulu olan bu uygulama, Windows 2000 işletim sisteminde çalışan VB ile yazılmış ve başka bir sql sunucusunda yer alan bir takım bilgileri tedarik eden bir yapıya sahip olabilir. İşte .net remoting ile bu iki farklı uzak nesne arasında kolayca iletişim kurabilir ve veri alışverişini sağlayabiliriz.

Remoting sisteminin bize verdiği aslında, herhangibir zamanda, yerde ve aygıt üzerinden bilgi erişimine izin verilebilmesidir. Bu aşağıdaki şekilde daha anlaşılır bir biçimde ifade edilmeye çalışılmıştır.

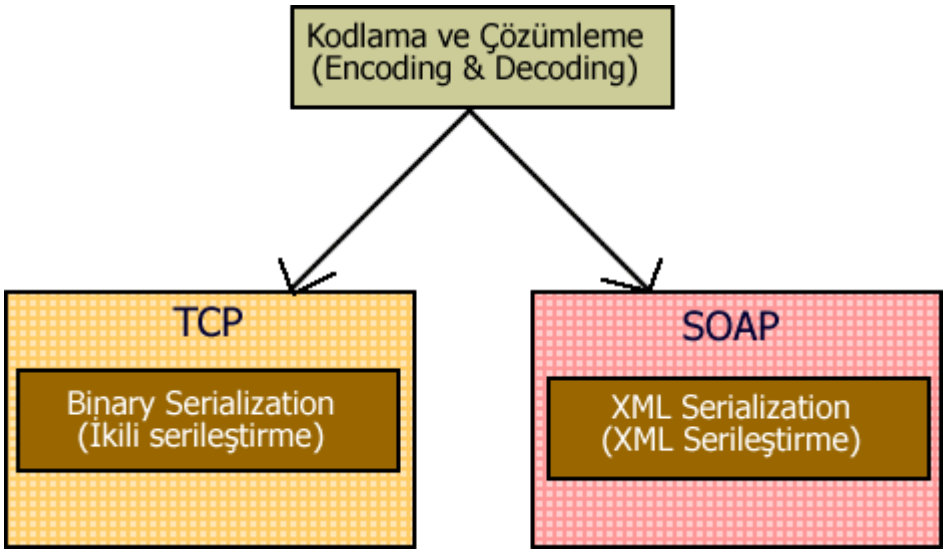


Şekil. 6. . NET Remoting Sisteminde Temel Amaç

.Net remoting sisteminin temel olarak nasıl çalıştığına ve nelere ihtiyaç duyduğuna değinecek olursak, öncelikli olarak sistemde uzak sınırlarda yer alan nesnelerin olduğunu söyleyebiliriz. Bu nesneler arasında meydana getirilecek iletişim, .net remoting alt yapısında yer alan bir takım üyeler ile sağlanacak. Herşeyden önce, uzak nesneler arasında hareket edicek mesajların taşınması işlemi ile, iletişim kanallarının (Communication Channels) ilgilendiğini söyleyebiliriz. Uzak nesneler arasındaki tüm mesajlar bu kanal nesneleri yardımıyla taşınacak, kodlanacak (encoding) ve çözülecektir (decoding) . Kodlamaktan kastımız, uzak nesneler arasında taşınacak mesajların, kullanılan iletişim protokolünün tipine göre belli bir formatta serileştirilmesi (serialization)dir. Diğer yandan kodlanan bu mesajın aynı kurallar çevrçesinde diğer uzak nesne tarafından çözümlenmesi (decoding) gerekecektir. Zaten aradaki mesajların yaygın iletişim protokolleri üzerinden gerçekleştirilmesi, remoting'in en temel özelliklerinden birisidir.

Bir iletişim kanalı nesnesi yardımıyla taşınan mesajlar, doğal .net serleştirici formatları (binary, soap) kullanılarak, kodlanır ve çözülürler. Bir uzak nesne, başka bir uzak nesneye mesaj gönderdiğinde, bu mesaj, kullanılan kanal nesnelerinin esas aldığı protokoller çerçevesinde serileştirilerek binary yada xml formatına dönüştürülür. Mesajı alan uzak nesne ise, serileştirilmiş bu mesajı uygun protokol

tabanlı .net serileştirme formatlarını kullanarak açar ve kullanır. Serileştirme amacıyla kullanılan iki kodlama çeşidi vardır.



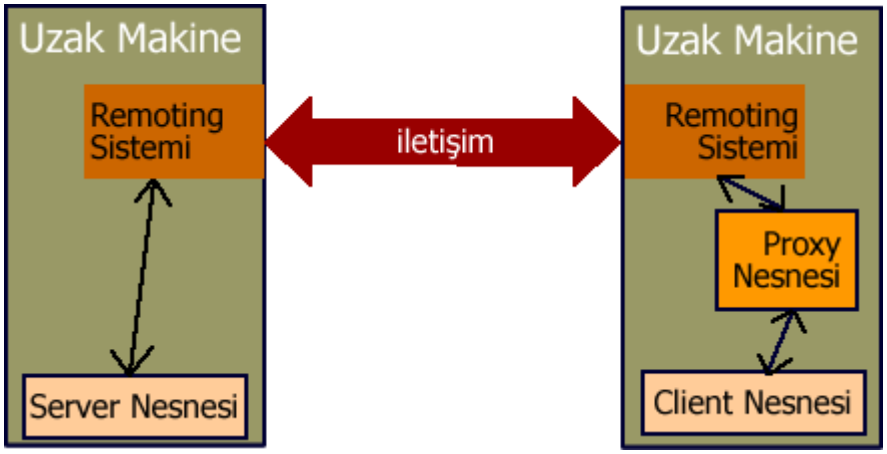
Şekil. 7. Serileştirme Seçenekleri

Uzak nesneler nasıl kullanılacak olunacağına değinecek olursak, özellikle server (Sunucu) nitelikli merkezi nesnelere, client (istemci) uygulamalardan nasıl erişilebilecek. Bunun için uzak nesnelere ait referanslara ihtiyacımız olucaktır. Ancak burada karşımıza adresleme sorunu çıkacaktır. Nitekim, herhangibir proses içinde çalışan bir nesne için, uzak nesnelerin çalıştığı prosesler anlamlı değildir. Bu sorunun aşılmasında kullanılabilecek ilk yol sunucu nesnesinin, istemcilere kopyalanmasıdır. Bu durumda istemci uygulama, sunucu nesnesinin bir örneğine ihtiyaç duyduğunda bunu local olarak kullanabilecektir.

Ancak kopylama yönteminin bir takım dezavantajları vardır. Herşeyden önce çok sayıda metod içeren büyük boyutlu nesnelerin kopyalanması verimlilik ve performans açısından negatif bir etki yaratacaktır. Bununla birlikte, istemci uygulamalar, server nesnelerinin sadece belirli metodlarını kullanıyor olabilirler. Buda nesnenin tamamımın kopyalanmasının anlamsız olmasına neden olan bir diğer durumdur. Bir diğer dezavantaj ise, server nesnesinin bulunduğu sistemdeki fiziki adreslere referanslar içerebiliecek olmasıdır. Örneğin dosya sistemine başvuruda bulunan nesneleri barındıran server nesneleri söz konusu olabilir. Böyle bir durumda elbetteki kopyalama sonucu bu referanslar yitirilecek ve istemci tarafından

kullanılamayacaktır. Son olarak, server nesnesinin kopyalanmasının, istemci kaynaklarını harcadığını söyleyebiliriz.

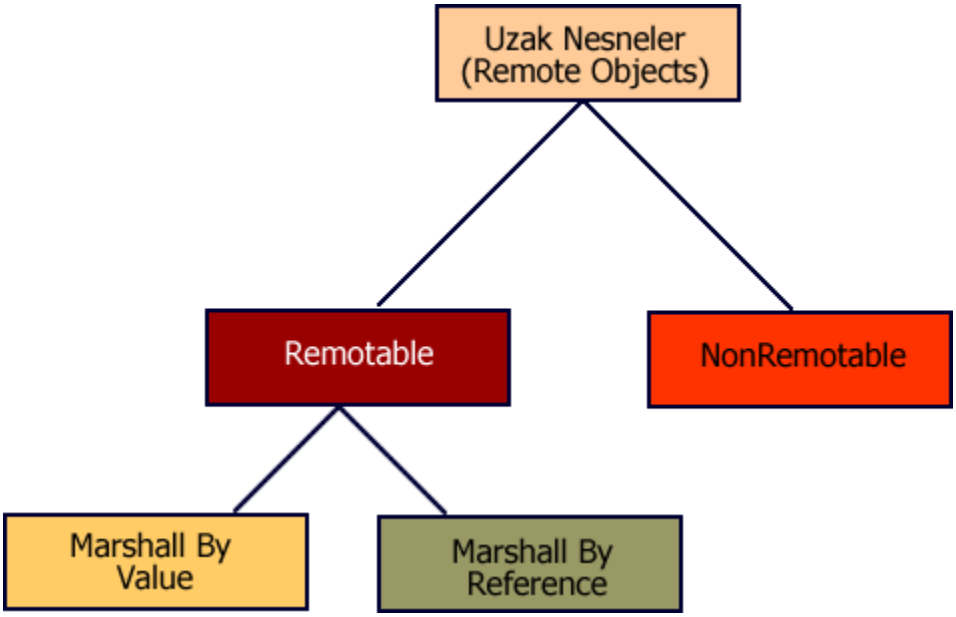
Bu dezavantajlar göz önüne alındığında bize başka bir yöntem gerekmektedir. Bu teknikte, server nesnelerini kullanmak için, bu nesnelere referansta bulunan proxy nesneleri kullanılır. İstemci uygulama, bir uzak nesneye ihtiyaç duyduğunda, bu talebini proxy nesnesine iletecektir. Proxy nesnesi ise, .net remoting'in sağlamış olduğu imkanlar dahilinde, ilgili uzak nesnesin ilgili üyesinin referansına başvuracak, bu metodu çalıştıracak ve dönen sonuçları yine proxy nesnesi aracılığıyla, istemci uygulamaya ulaştıracaktır. Bu konuyu aşağıdaki şekil ile daha net bir biçimde zihnimizde canlandırabiliriz.



Şekil. 8. Proxy Nesnesinin Kullanılması

Remoting sisteminde kullanılan nesnelerin kopyalanabilmesi veya refarans olarak değerlendirilmesi, remoting nesnelerinin iki temel kategoriye ayrılmalarına neden olmuştur. Remotable nesneler ve NonRemotable nesneler. Remotable nesneler, Başka uygulamalarca kopyalanma veya referans tekniği ile erişilebilen nesnelerdir. NonRemotable nesneler ise, diğer uygulamalar tarafından erişilemiyen nesnelerdir. Çoğunlukla büyük boyutlu, çok sayıda metod içeren veya local olarak fiziki adres referanslarına ihtiyaç duyan uzak nesnelerin, nonRemotable olarak belirtilmesi sık görülen bir tekniktir. Peki bu durumda bu nesneleri kullanmak isteyen uzak uygulamalar nasıl bir yol izleyebilir ? İşte bu noktada, nonRemotable nesnelerin, istemcilere açılabilecek olan bölümleri için remotable nesneler kullanılır.

Diğer yandan remotable nesneler, kopyalama veya referans taşımaya imkan veren uzak nesnelerdir. Burada ise karşımıza iki çeşit remotable nesne oluşumu çıkar. Marshall By Value tipi remotable nesneler ve Marshall by Reference tipi remotable nesneler. Bu kategorilendirme şekilsel olarak aşağıdaki gibidir.

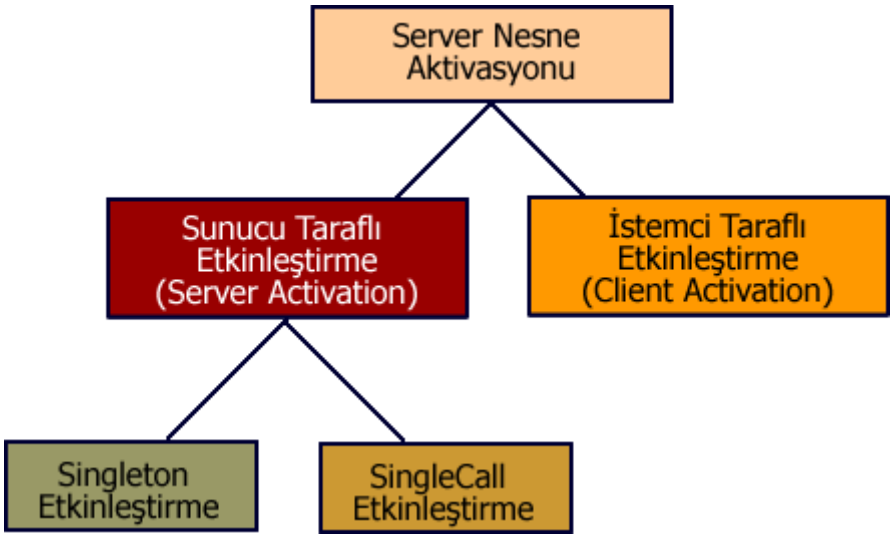


Şekil. 9. Uzak Nesnelerin Temel Kategorilendirilmesi

Özellikle remotable nesneler için yapılan ayrım kullanım açısından çok önemlidir. Marshall By Value olarak belirlenmiş remotable nesneler, bir istemci tarafından talep edildiklerinde, özellikle bu nesnenin herhangibir metodu istemci tarafından çağırıldığında, .net remoting sistemi bu nesnenin bir kopyasını oluşturur ve bu kopyayı iletşim kanal nesneleri yardımıyla serileştirerek, çağrıyı yapan istemciye gönderir. İstemci tarafında yer alan, .net remoting sistemi ise bu kopyayı çözümler ve bir nesnenin bir kopyasını istemci makinede oluşturur. Nesnenin, istemciye kopyalanabilmesi için serileştirme işlemi uygulanır. Burada önemli olan nokta, serileştirmenin otomatik olarak gerçekleştirilebilmesi için, nesneye ISerializable arayüzünün uygulanmasıdır. Nesnelerin, istemcilere bu yolla taşınması özellikle istemciler açısından işlem zamanını kısaltıcı bir etken olarak karşımıza çıkmaktadır. Nitekim istemcinin talep ettiği metodlara, artık istemcideki uzak nesnenin kopyası üzerinden erişilmektedir.

Marshall By Reference nesneleri ise, istemci uygulamalar için bu sistemlerde birer proxy nesnesi şeklinde oluşturulur. Bu proxy nesnesi, asıl uzak nesneye ilişkin referansları içeren metadata bilgilerine sahiptir. Uzak nesnenin herhangibir metodu çağırıldığında, proxy nesnesi ilgili metodun referansı ile hareket eder ve bir takım bilgileri sunucuya gönderir. Sunucu gelen bilgi paketini çözümledikten sonra ilgili parametreleri değerlendirerek talep edilen metodu çalıştırır ve bunun sonucu olarak geri dönüş değerlerini istemci makinedeki proxy nesnesine gönderir. Sonuçlar istemciye bu proxy nesnesi yardımıyla gönderilecektir.

Bir uzak nesnenin herhangibir metodunun çağırılmasında veya uzak nesneye ait bir örneğin istemcide new anahtar sözcüğü ile oluşturulmaya çalışılmasında, uzak nesnenin davranış biçimi ve aktifleştirilmesi önem kazanır. Nitekim uzak nesnenin aktif hale gelmesi iki teknik ile gerçekleştirilebilmektedir.



Şekil. 10. Nesne Etkinleştirme

Öncelikle sunucu taraflı etkinleştirmeden bahsedelim. Bu etkinleştirme tekniğinde, istemci tarafından sunucu nesnesinin bir metodu çağrıldığında, sunucu tarafından bu nesneye ait bir örnek oluşturulur. Daha sonrada bu nesne örneğinin proxy nesnesi, istemci tarafında oluşturulur. Burada dikkat çekici nokta nesne örneğinin, new anahtar sözcüğü ile değil, sunucu nesneye ait bir metod çağırıldığında oluşturuluyor olmasıdır.

Bu etkinleştirme tekniğine örnek olarak şunu gösterebiliriz. Devlet dairelerindeki sunuculara bağlı bürolar olduğunu düşünelim. Sunucu nesnemiz, bu bürolara, vergi numarasına göre kontrol ve kimlik bilgisi değerlerini gönderiyor olsun. Bu örnekte büro istemcilerinin, sunucuya sürekli bağlı olduklarını düşünelim. İstemci, bir vergi numarasını kontrol etmek istediğinde, sunucu nesnesindeki ilgili metodu çağıracaktır. İşte bu noktada sunucu taraflı etkinleştirme devereye girerek, metod çağırımına karşılık sunucu nesnesinin örneğini oluşturur.

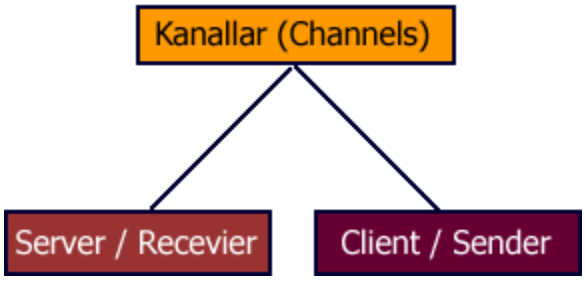
Diğer yandan sunucu taraflı etkinleştirmede, Singleton ve SingleCall teknikleri kullanımaktadır. Singleton tekniğine göre etkinleştirmede, sunucu nesneyi kullanan kaç istemci olursa olsun, her bir istemcinin metod çağırımları sunucu tarafındaki tek bir nesne örneği tarafından yönetilmektedir. SingleCall tekniğinde ise, istemci tarafından yapılan her metod çağırısına karşılık birer sunucu nesnesi örneği oluşturulacaktır.

İstemci taraflı etkinleştirmeye gelince. Bu kez sunucu taraflı etkinleştirmenin aksine, new anahtar sözcüğü ile bir sunucu nesne örneği istemci uygulamada oluşturulduğunda, sunucu üzerinde sunucu nesnesinin örneği oluşturulacaktır. Bu tip kullanıma örnek olarak chat uygulamalarını gösterebiliriz.

Remoting sisteminde uzak nesneler dışında en önemli unsur mesajları taşıyan kanal nesneleridir. (Channels) Kanal nesneleri uzak bilgisayarlarda yer alan uzak nesneler arasındaki haberleşmede rol oynayan kilit nesnelerdir. Aradaki iletişimi sağlamak için kullanılan kanal nesneleri, bu iletişim üzerinden mesajların gönderilmesi, taşınması ve alınması gibi işlemlerden sorumludurlar. Bildiğiniz gibi kanal nesnelerinin taşıdığı mesajlar HTTP veya TCP protokolleri çerçevesinde hareket ederler. Bir kanal nesnesi yardımıyla, uzak kanallara mesaj gönderebilir yada uzak kanallardan gelen çağrıları dinleyebiliriz.

Bir uzak nesne, başka bir uzak nesneye mesaj gönderdiğinde, kanal nesneleri devreye girerek bu mesajı binary veya xml formatında serileştirir. Mesajı yani çağrıyı alan uzak nesne ise, bu mesajın içeriğini, buradaki kanalın mesajı binary veya xml formattan çözümlemesi ile okuyabilir. Remoting sisteminde kullanılan kanallara ilişkin sınıflar ve arayüzler System.Runtime.Remoting.Channels isim alanında yer almaktadır.

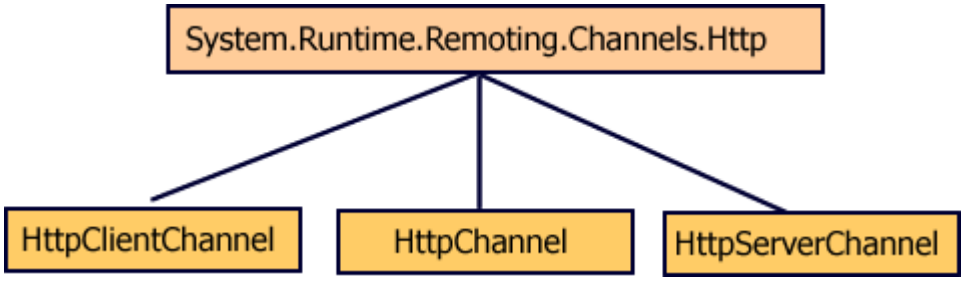
Temelde bütün kanal nesneleri IChannels arayüzünü uygulamaktadır. Kanal nesneleri iletişimde oynayacakları role göre kategorilendirilirler. Eğer kanal nesnesi çağrıları dinlemek amacıyla kullanılacaksa, alıcı (receiver) veya server (sunucu) olarak adlandırılırlar. Bununla birlikte, mesaj göndericek olan kanal nesneleri sender (gönderici) veya client (istemci) olarak adlandırılırlar.



Şekil. 11. Kanallar

Receiver kanallar, IChannelSender arayüzünü uygulayan kanallardır. Kullandıkları protokollere göre HttpClientChannel ve TcpClientChannel sınıflarından oluşturulurlar. Diğer yandan Sender kanallar, IChannelReceiver arayüzünü uygulayan TcpServerChannel ve HttpServerChannel nesneleridir. Diğer önemli iki kanal nesnesi ise HttpChannel ve TcpChannel nesneleridir.

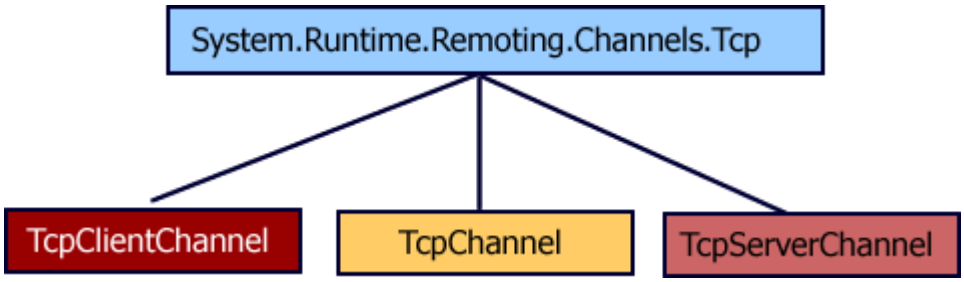
Şimdi dilerseniz bu kanalların .net remoting sisteminde oynadıkları rolleri kısaca incelemeye çalışalım. Burada önemli olan, mesajlaşmanın gerçekleştirileceği protokoldür. Nitekim HTTP protokolünü kullanacaksak buna uygun kanal nesnelerini kullanmalı, TCP protokolünü kullanacaksakta buna uygun kanal nesnelerini kullanmalıyız. Eğer HTTP protokolü kullanılacaksa, System.Runtime.Remoting.Channels.Http isim alanında yer alan kanal sınıflarını kullanırız.



Şekil. 12. Http Kanalları

İstemciden, uzak nesnelere mesaj göndermek için HttpClientChannel sınıfına ait nesne örnekleri kullanılır. Diğer taraftan, istemcilerden gelicek çağrıları dinleyecek kanal nesneleri ise, HttpServerChannel sınıfından örneklenir. HttpClient sınıfına ait nesne örnekleri ise mesaj almak ve mesaj göndermek için kullanılırlar. HttpClientChannel nesne örnekleri oluşturulurken, bu kanalın kullanacağı bir port numarasını özellikle belirtmemiz gerekmez. Remoting sistemi o an için kullanılabilen serbest portlardan birisini bu kanal nesnesi için tahsis edecektir. Diğer yandan çağrıları dinlemek amacıyla tanımlanan bir HttpServerChannel kanal nesnesinin belirli bir port numarası üzerinden oluşturulması gerekmektedir. Şayet o an için kullanımda olan bir port belirlenirse çalışma zamanında istisna alırız. Bununla birlikte HttpServerChannel nesnesinin yapıcı metoduna 0 değerini göndererek, port seçme insiyatifini otomatik olarak .net remoting sistemine bırakabiliriz.

Http kanalları SoapFormatter sınıfını kullanarak, mesajları xml formatında serileştirirler. Bununka birlikte, TCP protokolünü kullanan kanal nesneleri ise, serileştirme işlemini binary olarak yapar ve bunun içinde BinaryFormatter sınıfını kullanırlar. TCP protokolünü taban alan kanal nesneleri, System.Runtime.Remoting.Channels.Tcp isim alanında yer alan sınıflardan örneklenirler.



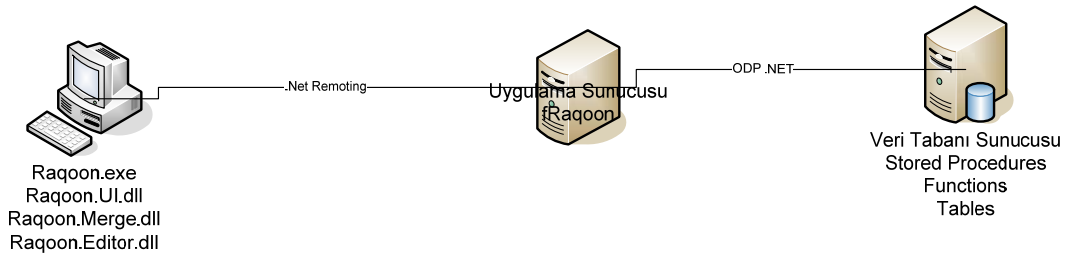
Şekil. 13. Tcp Kanalları

Tcp isim alanında yer alan sınıflar, Http'dekiler ile aynı işlevlere sahiptirler. Tek fark, kullanılan serileştirme işleminin farklı oluşudur. Her iki isim alanı içinde, oluşturulan kanal nesne örneklerinin ChannelServices sınıfında yer alan static RegisterChannel metodu ile sisteme kayıt edilmeleri gerekmektedir.(Registiration)

Kanallaların kullanımında dikkat edilmesi gereken en önemli nokta, uzak nesnelerin aynı protokolü destekleyen kanalları kullanmalarının gerekli olduğudur. Örneğin, istemcilerden gelen çağrıları dinlemek amacıyla Http protokolünü taban alan kanal nesneleri kullanılıyorsa, istemcilerdede mesaj göndermek için Http protokolünü taban alan kanal nesneleri kullanılmalıdır. Nitekim farklı protokol tabanlı kanalların kullanılmasında istisnalar alırız. Bunun sebebi, farklı protokol kullanımının sonucu olarak kodlama ve çözümleme işlemlerinin farklı serileştirme teknikleri içerisinde yapılıyor olmasıdır. Http ve Tcp kanalları arasındaki farkları şu şekilde özetleyebiliriz.

- Http kanal nesneleri Http protokolünü, Tcp kanal nesneleri ise Tcp protokolünü kullanır.
- Serileştirme işleminde, Http kanal nesneleri SoapFormatter sınıfını kullanırken, Tcp kanal nesneleri BinaryFormatter sınıfını kullanır.
- Http kanal nesneleri için serileştirme xml tabanlı yapılırken, Tcp kanal nesneleri için serileştirme binary formatta yapılır.

3.4.3. Yazılım Topolojisi



Şekil. 14. Yazılım Topolojisi

Uygulama çok katmanlı mimari yapısında gerçekleştirilmiştir. Uygulamanınveri tabanıyla erişimi yanlızca uygulama sunucusu üzerinde yapılmaktadır. Client tarafında windows formlar ve kütüphaneler çalışmaktadır.

Bunlar Raqoo.exe,Raqoon.UI.dll,Raqoon.Merge.dll,Raqoon.Editor.dll'dir. Uygulama sunucusu katmanında fRaqoon.dll çalışmaktadır. Veri tabanı katmanında stored procedure'ler, fonksiyonlar ve tablolar bulunmaktadır. Uygulamada kod içersinde herhangibir veri tabanı script'i bulunmaz, tüm veri tabanı işlermleri stored procedureler ve fonksiyonlar tarafından yapılmaktadır. Çok katmanlı mimaride çalışmanın bize sağladığı bir takım avantajlar vardır. Bunlar;

- Zamandan tasarruf.
- Daha kolay adapte olma.
- Daha kolay anlaşılabilir kod parçaları.
- Hata riskinin en aza inmesi.
- Yapılacak değişiklerde sadece belli noktaları değiştirerek köklü değişiklikler.
- Modülerlik, Yazdığımız class'ları birçok yerde veya projede kullanabilme.

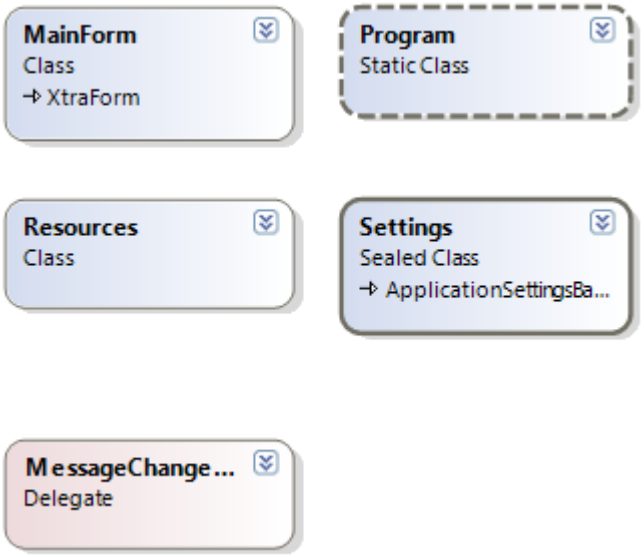
Üç katmanlı mimarı projelerimiz içerisinde kullandığımız veri tabanı bağlantısı, veri tabanı işlemleri (insert, update, delete, select, vb.) ve kullanıcı ara yüzünü birbirinden ayıran bir yapıyı bizlere sunar.

Çok katmanlı mimari yapısının kullanım sebeplerinden birisi ve bence en önemlisi tekrar eden satır sayısını ortadan kaldırılmasıdır. Yani yazdığımız satırlar mümkün olduğunca tekrar etmemelidir. Tekrar etmeyen satırlar sayesinde projemize daha fazla hükmederiz buda bize ve bizden sonra proje ile ilgilenecek kişilere (yazılımcı arkadaşlara) daha kolay bir adapte olma seçeneği sunar.

Bir tek yerde yazılan kod parçacıkları birçok yerde kullanılarak merkez kod parçacıkları oluşturmak temel amaçtır.

3.4.3.1. Raqoon Main

Uygulamanın çalışabilir (Executable) katmanıdır. Bir windows exe olarak disk üzerinde durmaktadır. Temel olarak tek bir kullanıcı arayüzünden ve ona bağlı bir sınıftan oluşmaktadır. Buna ait kod örneği EK-1'de yer almaktadır.

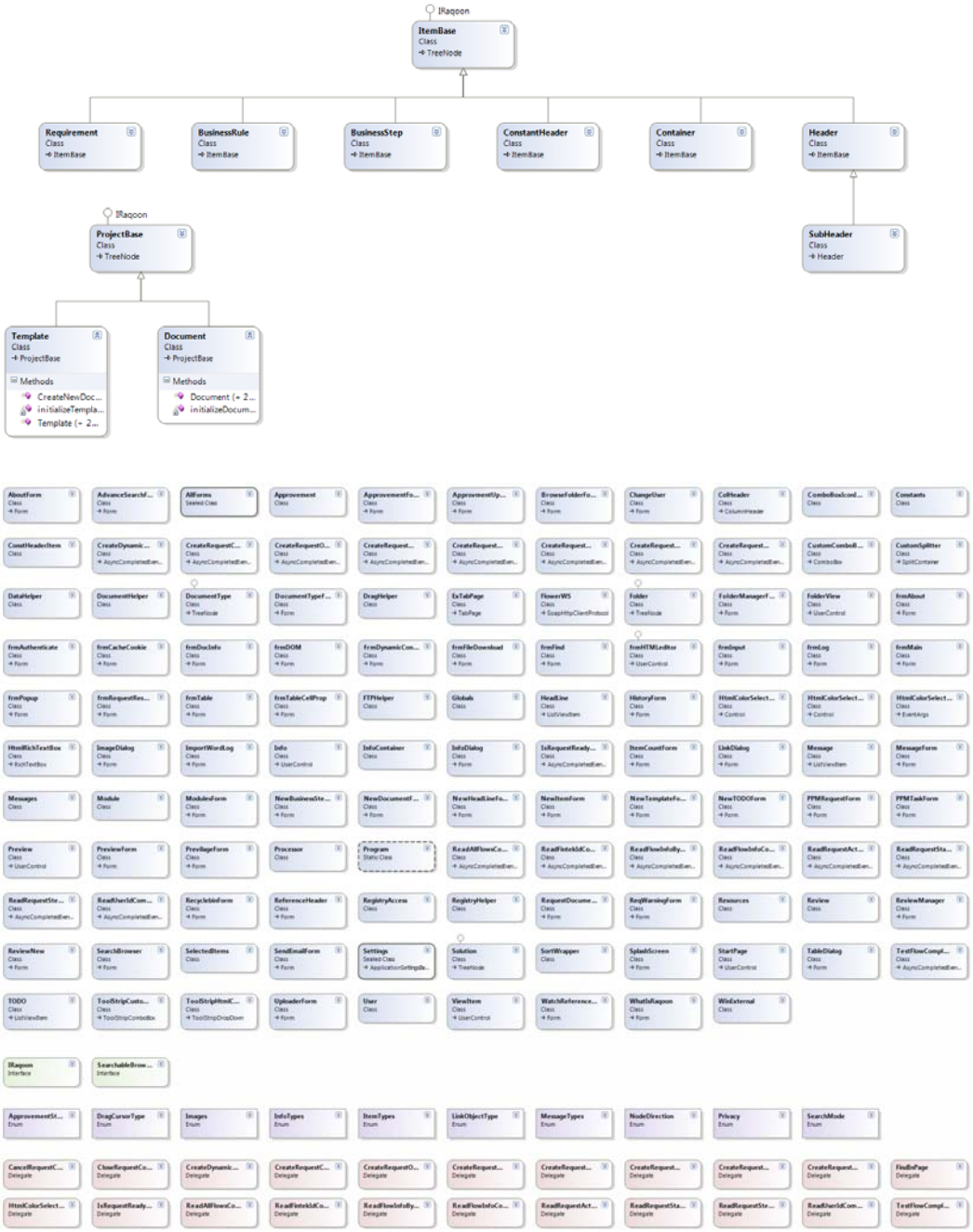


Şekil. 15. RaqoonMain Sınıf Diagramı

3.4.3.2. Raqoon UI

Uygulamanın client tarafında bulunan katmanıdır. Uygulamanın en önemli parçacığıdır. Tüm iş mantığı bu katmanda bulunmaktadır. Temel olarak iki kısımdan oluşmaktadır. Bunlar ItemBase ve ProjectBase adını verdiğimiz base sınıflardır. ItemBase sınıfı doküman başlıklarını oluşturmak için kullanılmaktadır. Yani Raqoon'unu kodlarken oluşturulması gereken tüm başlık sınıfları ItemBase sınıfından türemek zorundadır. Aynı şekilde ProjectBase sınıfıda doküman ve şablonları hazırlamak için kullanılır. Yani uygulama geliştirici tarafından

oluşturulacak doküman ve şablon sınıfları ProjectBase sınıfından türemelidir. Aynı zamanda projectbase ve itembase sınıfları *IRaqoon interface'inden türetilmektedir. Diğer sınıflar tamamiyle doküman ve başlıklara yardımcı sınıflardır. Örneğin FTPHelper sınıfı FTP sunucusu ile bağlantı kurulmasını sağlayan bir sınıftır. Aynı şekilde DocumentHelper sınıfıda uygulamanın MS word ile bağlantısını sağlayan bir sınıftır. Raqoon.UI'a ait kod örneği EK-2'de yer almaktadır.



Şekil. 16. Raqoon.UI Sınıf Diagramı

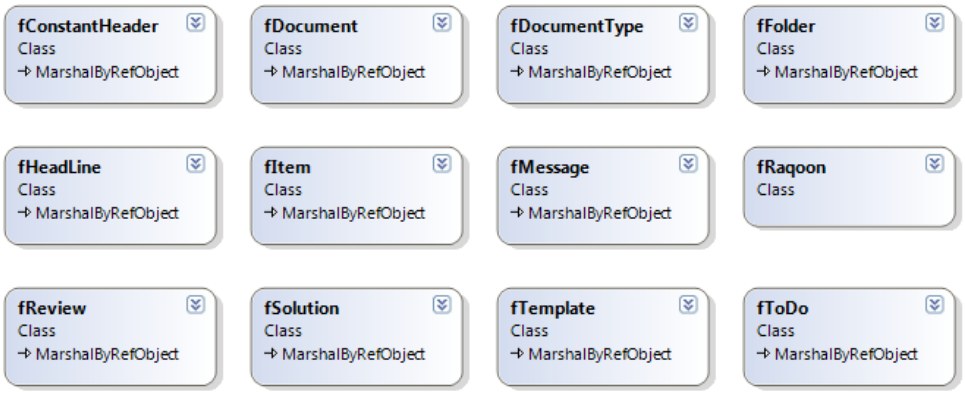
3.4.3.3. Raqoon Editor

Yine client tarafında çalışan ve uygulmadaki editörün oluşturulduğu kütüphanedir. Hazırlanan editör uygulaması html editör olarak tasarlanmıştır. Yani bu editörle yazılan tüm yazılar arka tarafta html olarak tutulmaktadır. .Buna ait kod örneği EK-3'de yer almaktadır.

Şekil. 17. Raqoon.Editor Sınıf Diagramı

3.4.3.4. fRaqoon

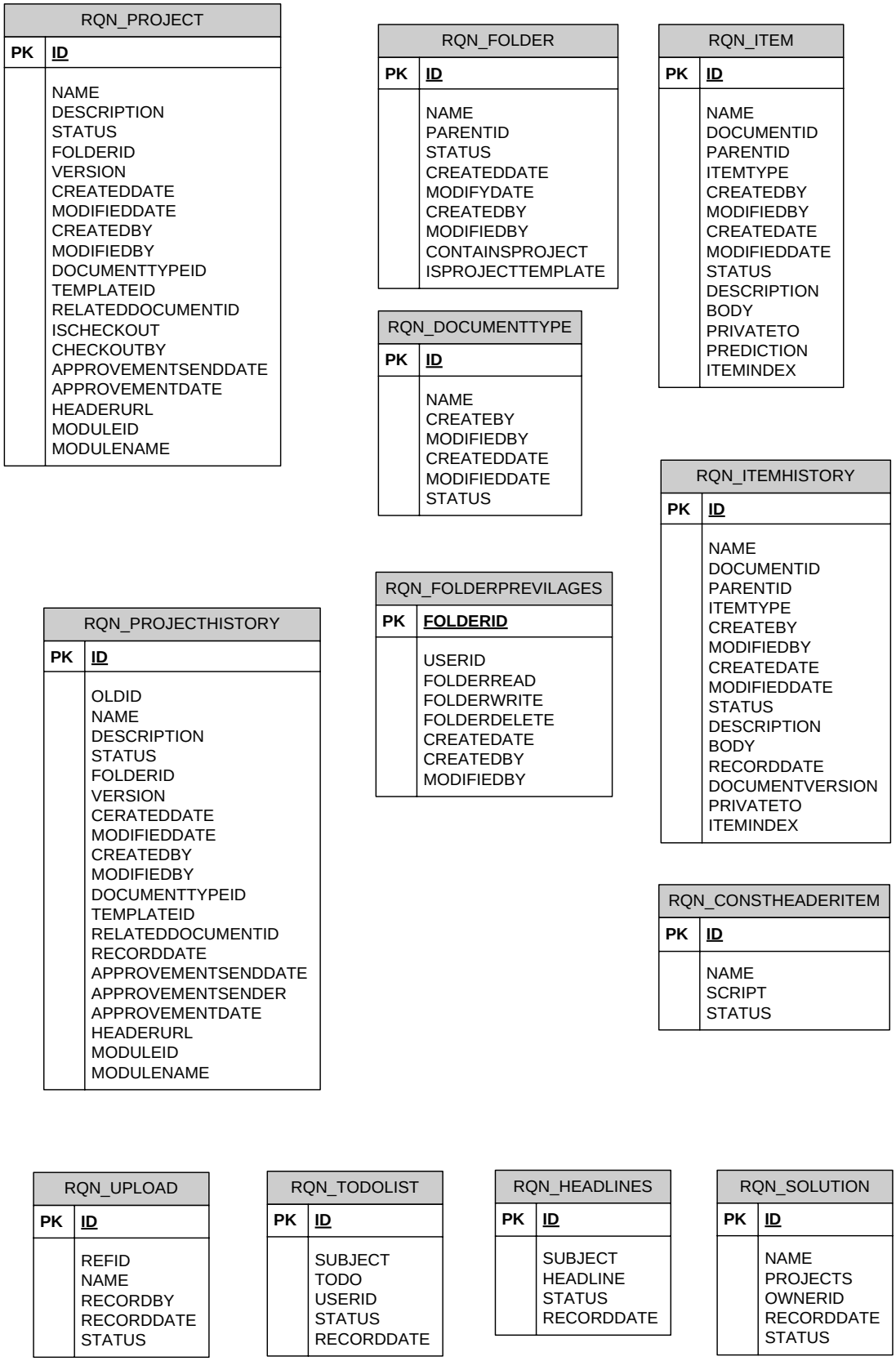
Uygulama sunucusu katmanında çalışan ve uygulamanın veritabanı ile alışverişleri yapmasını sağlayan katmandır. Daha önceki tasarımımızda belirttiğimiz gibi istemci tarafı uygulama sunucusundaki fRaqoon ile konuşmak için .Net Remoting teknolojisini kullanmaktadır. Yine fRaqoon ile ilgili söylenebilecek en önemli konu, istemci ile konuşan tüm sınıflar MarshalByRefObject sınıfından türemektedir. Tüm sınıfların bu sınıftan türemesinin sebebi tamamen serleştirme yapabilmek amacıylandır. Serileştirmeyi en yalın hali ile istemci ve uygulama sunucusunun aynı dili konuşmalarını sağlayan bir işlem olarak tarif edebiliriz. fRaqoon'a ait kod örneği EK-4'te yer almaktadır.



Şekil. 18. fRaqoon Sınıf Diagramı

3.4.4. Veri Tabanı

Uygulama veri tabanı olarak Oracle 10g veri tabanı kullanmaktadır. Uygulamada veri tabanı geliştirme ortamı olarak pl/sql ugulaması kullanılmaktadır. Uygulamada veri tabanı tablolarına erişim yazlızca stored prosedürler ve fonksiyonlar yardımıyla olmaktadır. Bu sayede uygulama veri tabanı seviyesinde hem güvenli hemde kolay yönetilebilir bir hal almaktadır. Veri tabanı tablolarında performans kaygısından dolayı dış anahtarlar özellikle kullanılmamıştır.



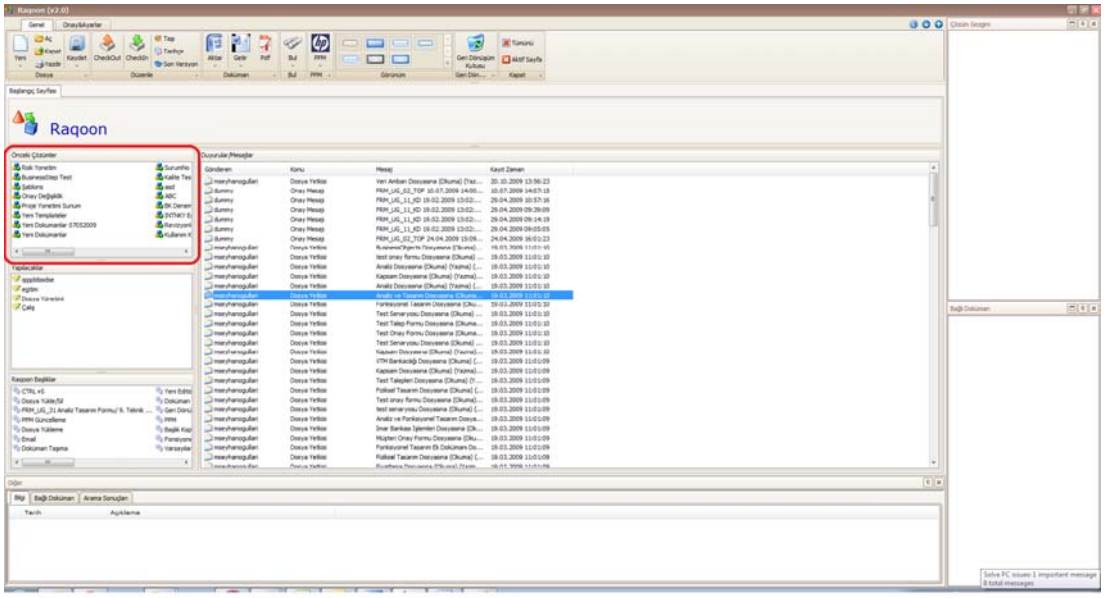
Şekil 19. Veri Tabanı

3.5. Uygulamanın Çalışma Şekli

3.5.1. Uygulama Açılış

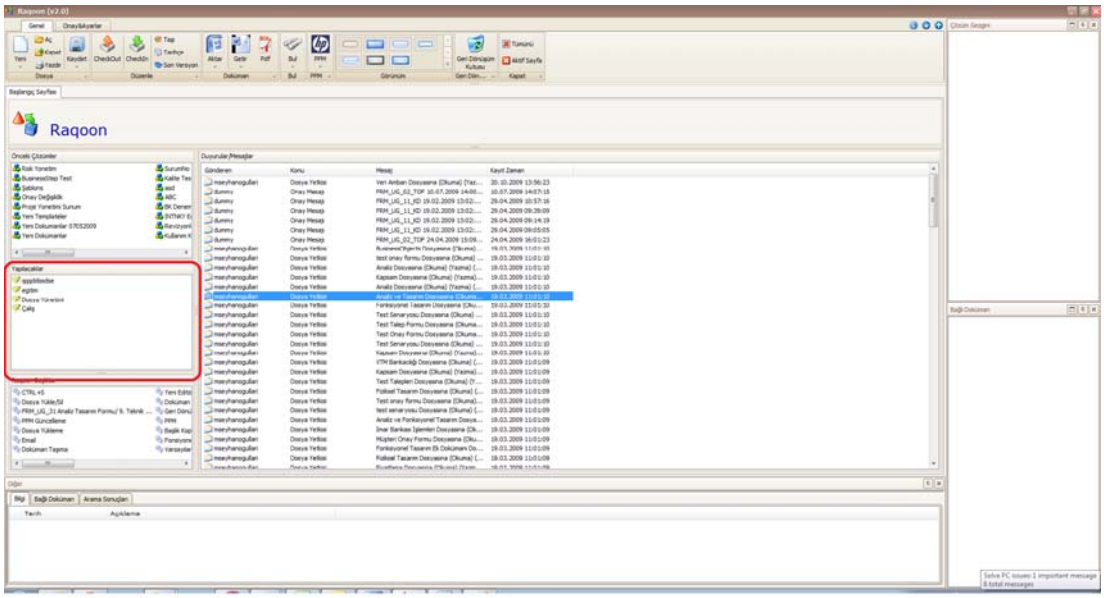
Uygulama ilk açıldığında şekil 19’ da görülen ekran açılır. Uygulamamız temel olarak yedi ana bölümden oluşmaktadır. Bu bölümler;

1. Önceki Çözümler : Uygulamamız dokümanları çözüm adını verdiğimiz sanal linkler üzerinde tutmaktadır. Bir kullanıcı uygulamayı kullanıp doküman oluşturacağı zaman dokümanı önceden oluşturmuş olduğu veya yeni oluşturacağı bir çözüm’e bağlar. Bu çözümler oluşturulan dokümanların gruplanmasını sağlar. Bu sayede kullanıcı bir doküman grubuna ulaşmak istediğinde dokümanı klasörlerden aramak yerine çözümü açarak otomatik olarak açılmasını sağlar.

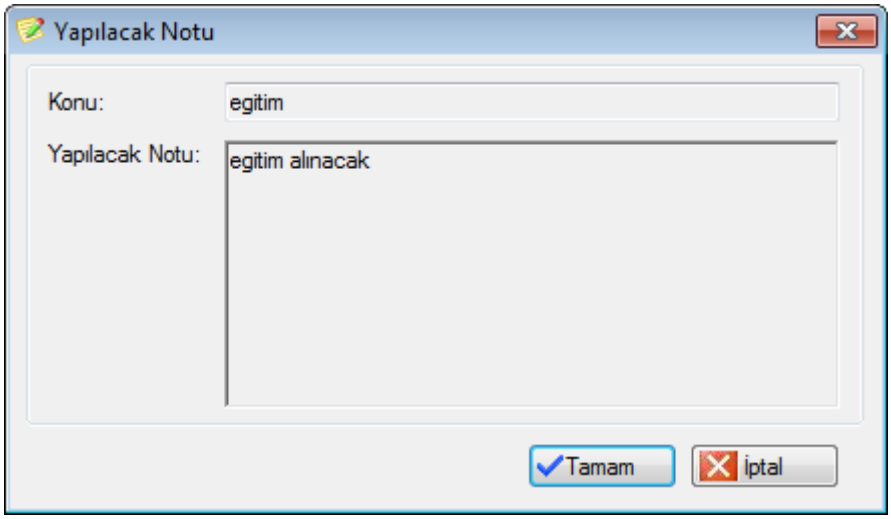


Şekil. 19. Önceki Çözümler

2. Yapılacaklar : Bu bölüm kullanıcının todo listesi olarak kullanılması amacıyla yapılmıştır. Kullanıcı bu panele sağ tılayarak yeni adımını seçer ve kendi için yapılacak notu girer. Aynı şekilde kullanıcı yapılan işi buradan silebilir. Şekil 20’de yapılacaklar ekran görüntüsünü görebilirsiniz.

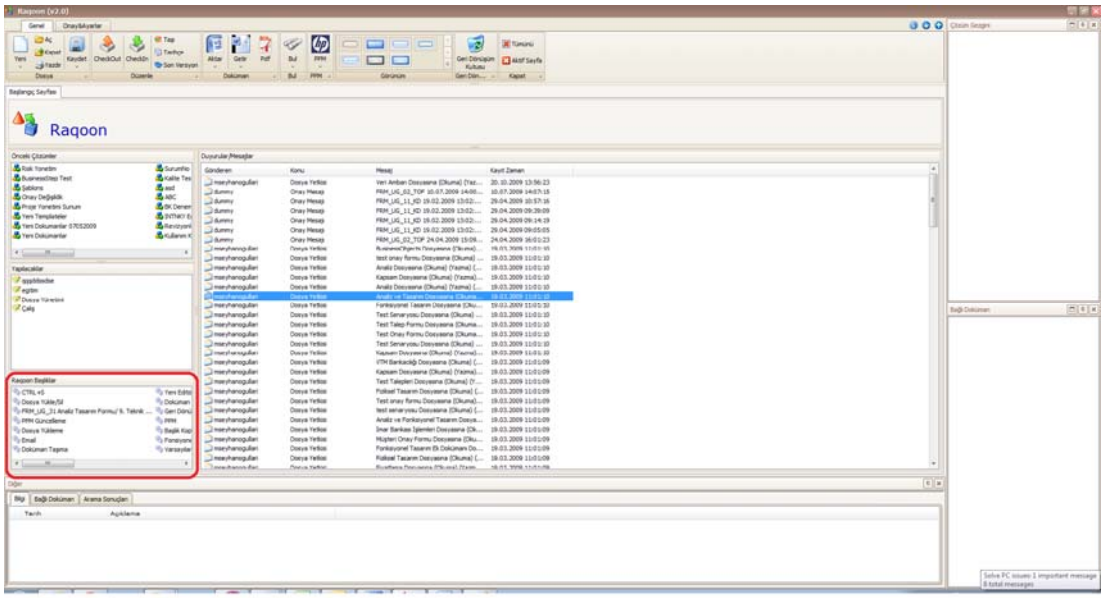


Şekil. 20. Yapılacaklar



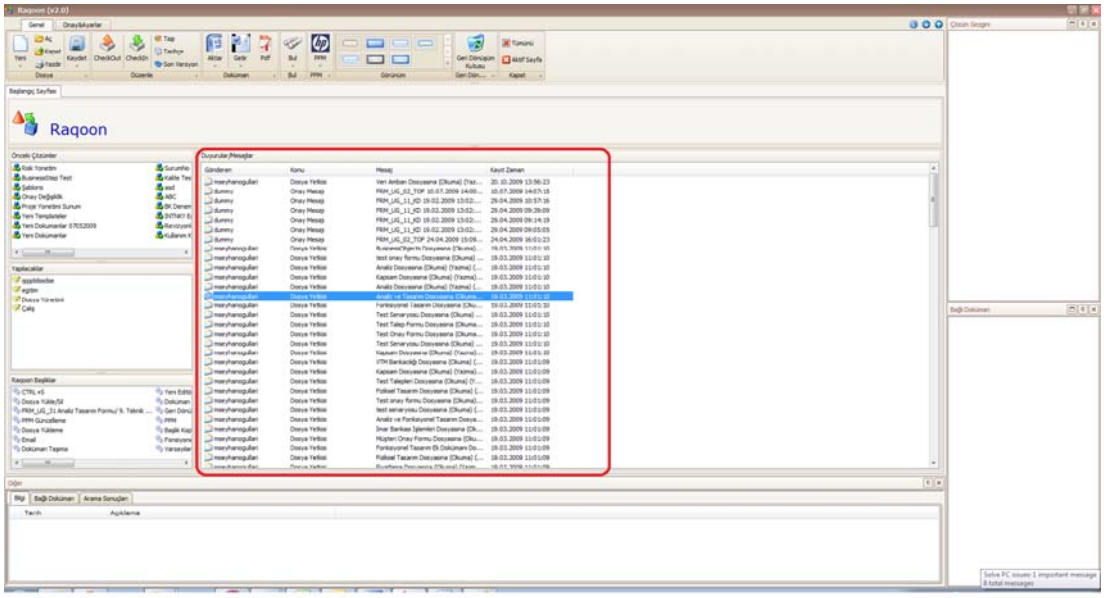
Şekil. 21. Yapılacak Notu

3. Raqoon Başlıklar : Bu panel uygulamanın yazılımcısı veya yöneticisine aittir. Uygulamanın yazılımcısı veya yöneticisi yapılan yenilikleri buradan duyurarak tüm kullanıcıları bilgilendirmektedir. Şekil 22'de Raqoon Başlıkların ekran görüntüsünü görebilirsiniz.

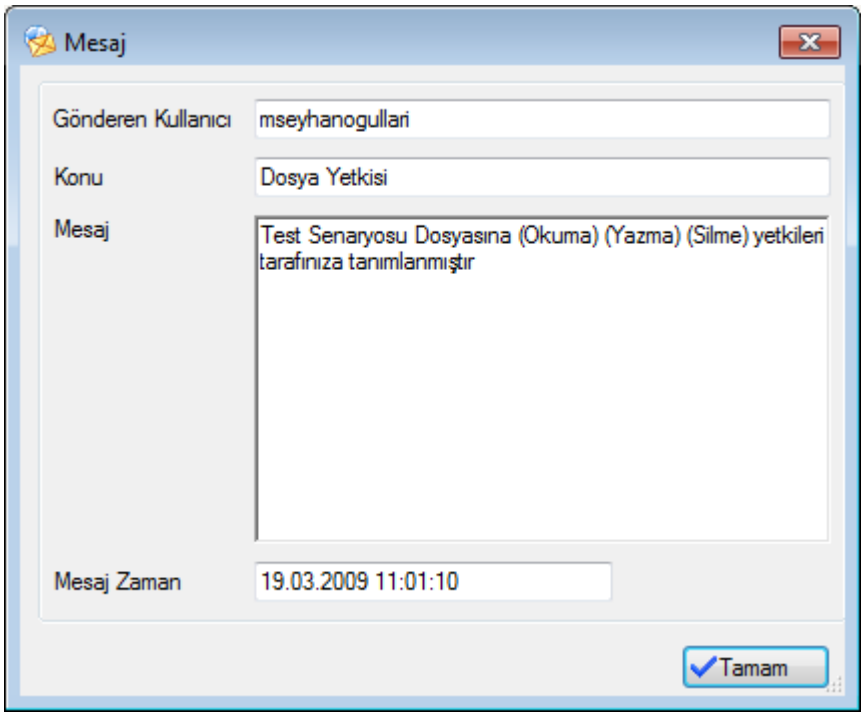


Şekil. 22. Raqoon Başlıklar

4. Duyurular/Mesajlar : Bu panel yaptığımız uygulama ile entegre çalışan (ör: Onay uygulaması) diğer uygulamalardan gelen mesajları göstermek amacıyla kullanılacaktır. Şekil 23'te örnek ekran görüntüsünü görebilirsiniz.

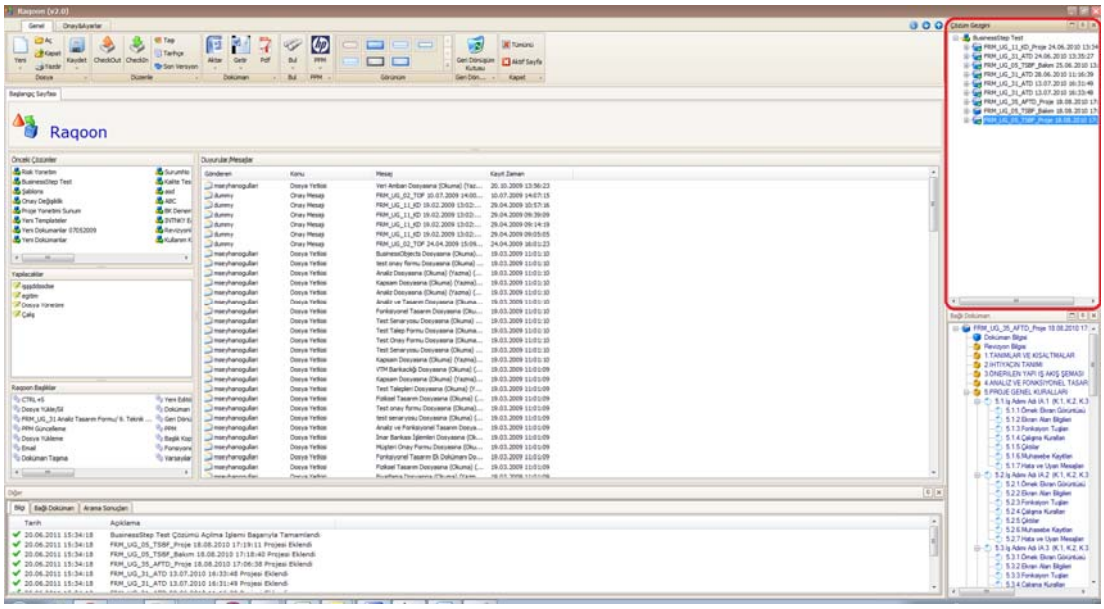


Şekil. 23. Duyurular/Mesajlar



Şekil. 24. Mesaj

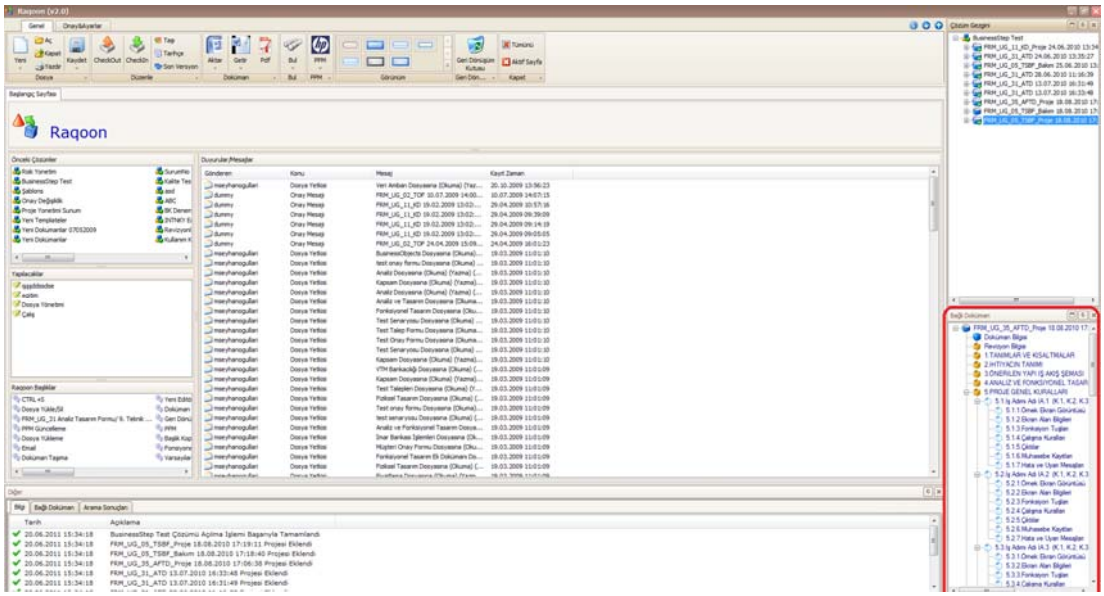
5. Çözüm Gezgini : Kulanıcı tarafından açılan çözümlerin görüntülendiği paneldir. Herseferinde yanlızca bir çözüm gösterilmektedir. Şekil 25'te açılan bir çözümü ve bu çözüme bağlanmış olan dokümanları görebilirsiniz.



Şekil. 25. Çözüm Gezgini

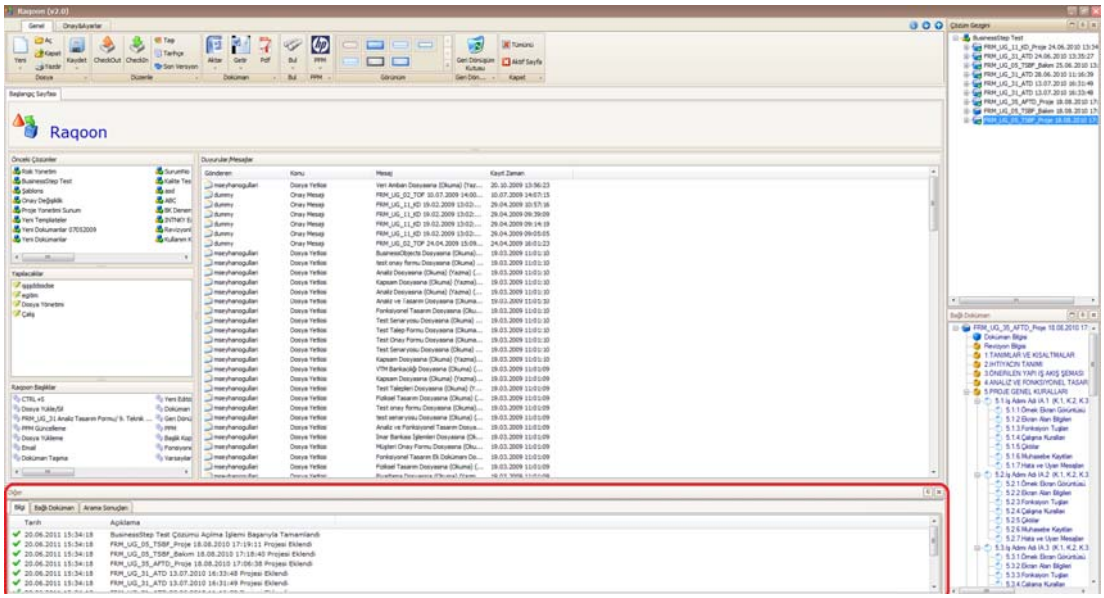
6. Bağlı Doküman : Bağlı doküman panelinde açılan bir dokümanın ilişkili olduğu dokümanları görmenizi sağlar. Uygulama geliştirme süreçlerinde dokümanlar

bir birlerinden türemektedir. Örneğin analiz dokümanı kapsam dokümanından, test senaryosu dokümanı analiz dokümanın türemektedir. Bağlı doküman panelinde bu ilişkileri görebilmeniz mümkündür.



Şekil. 26. Bağlı Doküman

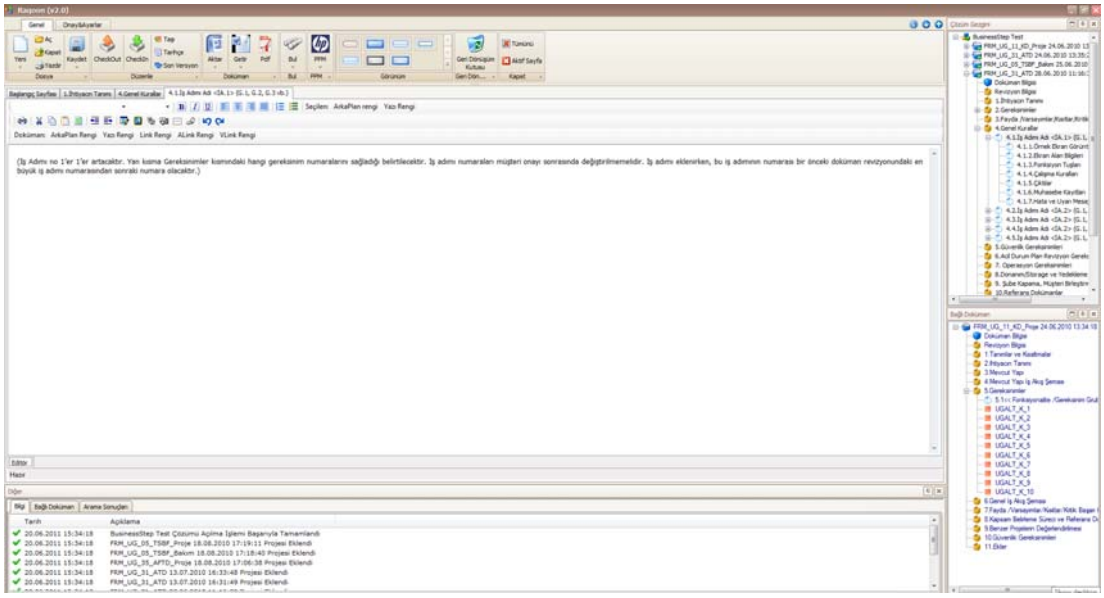
7. Diğer : Diğer paneli üç ayrı bölümden oluşmaktadır. Bunlar, Bilgi,Bağlı Doküman ve Arama Sonuçlarıdır. Bilgi pencerisnde uygulama içerisinde yapılan tüm hareketler görüntülenmektedir. Bağlı doküman penceresinde bağlı dokümana ait içerikleri görmeniz mümkündür. Arama sonuçları bölümünde adındanda anlaşılacağı gibi arama sonuçları görüntülenmektedir.



Şekil. 27. Diğer Paneli

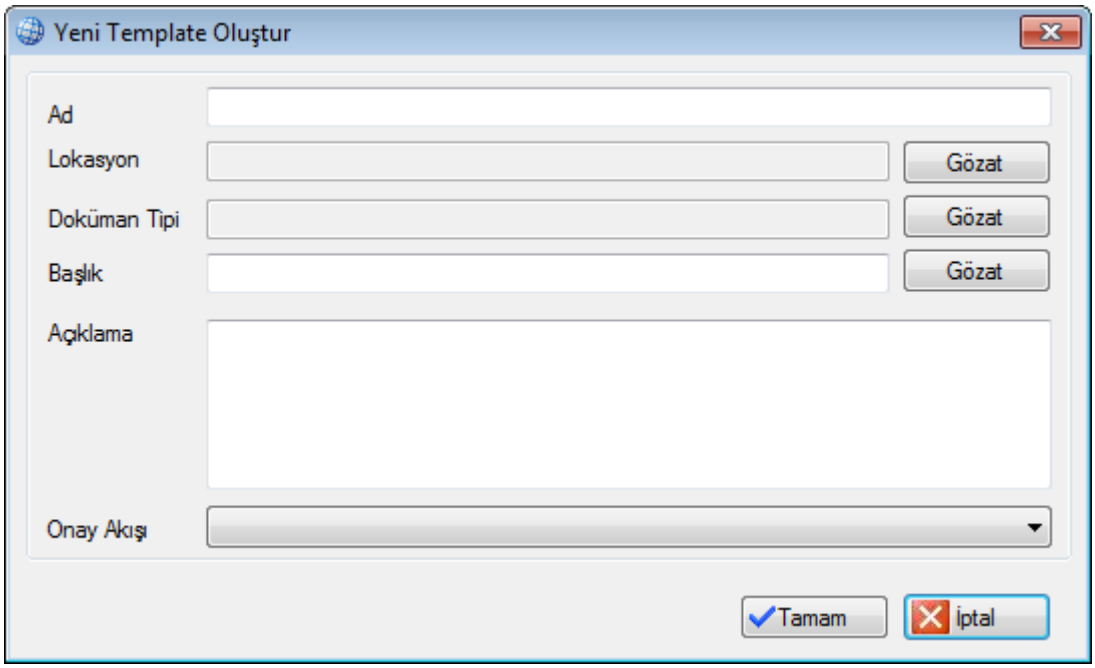
3.5.2. Şablonlar

Uygulama kullanılmaya başlanılmadan önce doküman şablonları oluşturulmalıdır. Bu şablonlar dokümanların birçok özelliğini üzerlerinde tutarlar. Şekil 32'de bir çözüme bağlanmış ve önceden oluşturulmuş şablonlar görülmektedir.



Şekil. 28. Şablonlar

Yeni şablon oluşturmak için açılmış olan çözüme sağ tıklanır ve yeni template ekle seçeneği seçilir. Bu seçenek seçildikten sonra şekil 33'de görünen pencere açılır ve şablon için gerekli bilgiler girilir. Bu bilgiler şablonun adı, lokasyonu,doküman tipi,Başlık,Açıklama ve Onay Akışıdır.

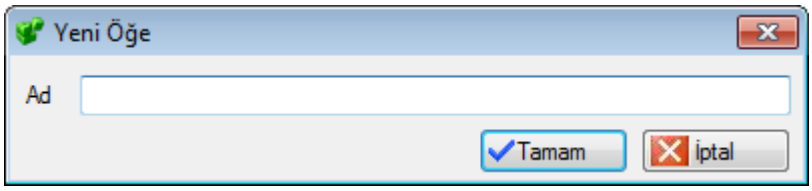


Şekil. 29. Yeni Template Oluşturma

Şablon için gerekli bilgiler girildikten sonra şablon çözüme eklenir. Bu aşamadan sonra yapılması gereken şablona gerekli başlık ve alt başlıkları eklemektir. Bu başlık ve alt başlıklar bu şablondan oluşacak dokümanların isketlerini belirler. Kullanıcı oluşturulan şablondan bir doküman oluşturmak isterse başlangıç olarak şablonun bir kopyası şeklinde doküman oluşur. Dolayısı ile başlık içeriklerine şablon oluşturulurken yazılan metinler doküman içeriklerindede aynı şekilde görülür. Bu şekilde olmasının en büyük faydası şablon oluşturulurken başlıkların içersine neler yazılması gerektiği ile ilgili kullanıcıya ipuçları yazılır. Bu sayede ilgili şablondan doküman oluşturan tüm kullanıcılar ipuçlarını okuyarak başlıların içersine neler yazılacağını öğrenebilirler.

Şablon İşlem Seti:

- Yeni Başlık Ekle : Şablona yeni bir ana başlık eklemek amacıyla kullanılır. Bu işlem tıklandığında çıkan ekrana başlık adı girilir ve yeni başlık dokümana eklenir.
- Sabit Başlık Ekle: Şablona yeni bir sabit başlık eklemek amacıyla kullanılır. Bu işlem tıklandığında çıkan ekrana başlık adı girilir ve yeni sabit başlık şablona eklenir. Sabit başlık şablona eklendikten sonra script'ler başlığın editörüne yazılmak suretiyle sabit başlık oluşturulur.

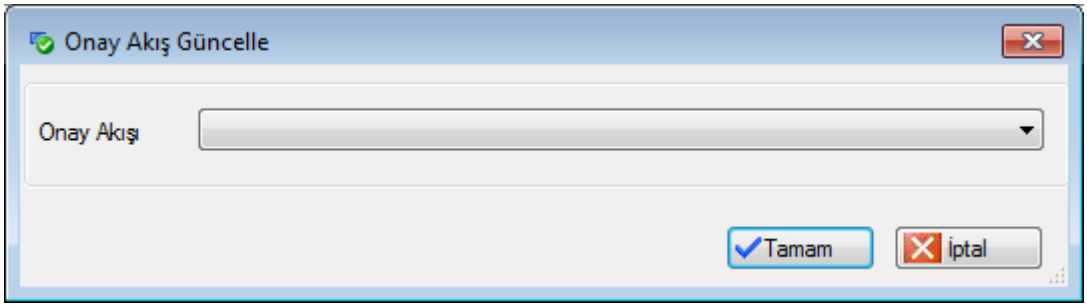


Şekil. 30. Yeni/Sabit Başlık Ekleme

- Check-Out : Uygulamamızda bir şablon aynı anda yanlızca bir kişi tarafından yazılabilir. Şablonu yazacak kullanıcı dokümanı check-out ederek şablonu üzerine alır. Böyle bir durumda aynı dokümanı yazmak isteyen kullanıcı şablonun başkası tarafından check-out edildiğini öğrenir ve yanlızca okuma amaçlı açabilir.
- Check-In: Doküman kullanıcı tarafından check-out edildikten sonra ilgili kullanıcı tarafından yazımına başlanır. Kullanıcının şablon ile işi bittikten sonra dokümanı check-in eder ve böylelikle şablon serbest kalmış olur. Böyle bir durumda bir başkası dokümanı check-out ederek şablonu yazabilir.
- Check-Out'u Geri Al: Uygulamamızın önemli özelliklerinden biri her doküman check-out edildikten sonra bir önceki versiyon kayıt altına alınır. Bu sayede kullanıcı dokümanı check-out edip gerekli işlemleri yaptıktan sonra eğer check-out'u geri al der ise doküman olduğu gibi check-out'u edilmeden önceki versiyonuna döner.
- Son Versiyonu Getir: Bilindiği gibi şablon check-out edilmiş ise yanlızca okuma amaçlı dokümanı açabiliyorsunuz. Böyle bir durumda sizin açtığınız doküman halen başkası tarafından güncelleniyor olabilir. İşte dokümanın en güncel halini getirebilmeniz için son versiyonu getir özelliğini kullanırsınız.
- Kaydet: Adındanda anlaşılacağı gibi şablonu kayıt etmek amacıyla kullanılır. Fakat bu durum çoğunlukla check-in ve check-out ile karıştırılmaktadır. Kaydet denildiği zaman check-out edilmiş ve bilgisayarınızın hafızasında duran bilgiler veri tabanına yazılır.Bu sayede herhangibir sebepten dolayı bilgisayarınız kapanır ve bellek problemi yaşar ise o zaman kadar yazmış olduğunuz bilgiler kaybolmaz.
- Adını Değiştir: Kullanıcının şablonun adını değiştirmesini sağlar.
- Sil: Şablonu silmenizi sağlar.
- Ön İzleme: Adındanda anlaşılağı gibi dokümanın tamamının izlenmesini sağlar. Bildiğiniz gibi uygulamamızda doküman başlılarının herbiri ayrı editörler

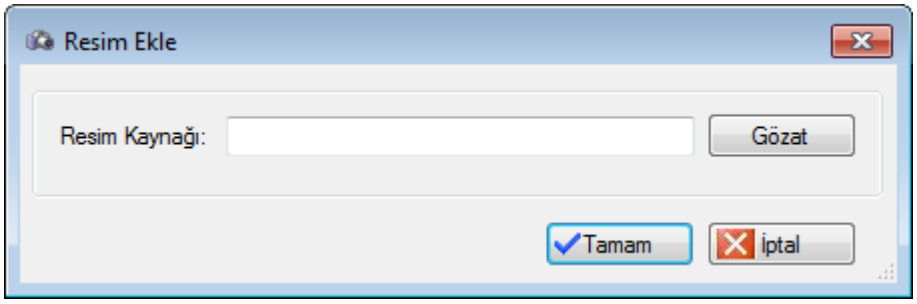
vasıtasıyla yazılmaktadır. Bunları bir bütün olarak izlemek amacıyla ön izleme fonksiyonu kullanılır.

- Yazdır: Dokümanı printer'da yazdırmak amacıyla kullanılır.
- Onay Akış Ekle Güncelle: Dokümanın şablonundan ulaşmış onay akışı dışında herhangibir başka onay akış kullanılmak istenirse bu özellik kullanılabilir.



Şekil. 31. Onay Akış Güncelleme

- Başlık Ekle/Güncelle: Bilindiği gibi tüm dokümanlar şablonlardan türetilmektedir. Başlık kısmıda dokümana atılacak başlığı belirtmektedir. Başlığın şablona tanımlanması sayesinde kullanıcılar doküman başlıklarıyla uğraşmak zorunda kalmazlar.

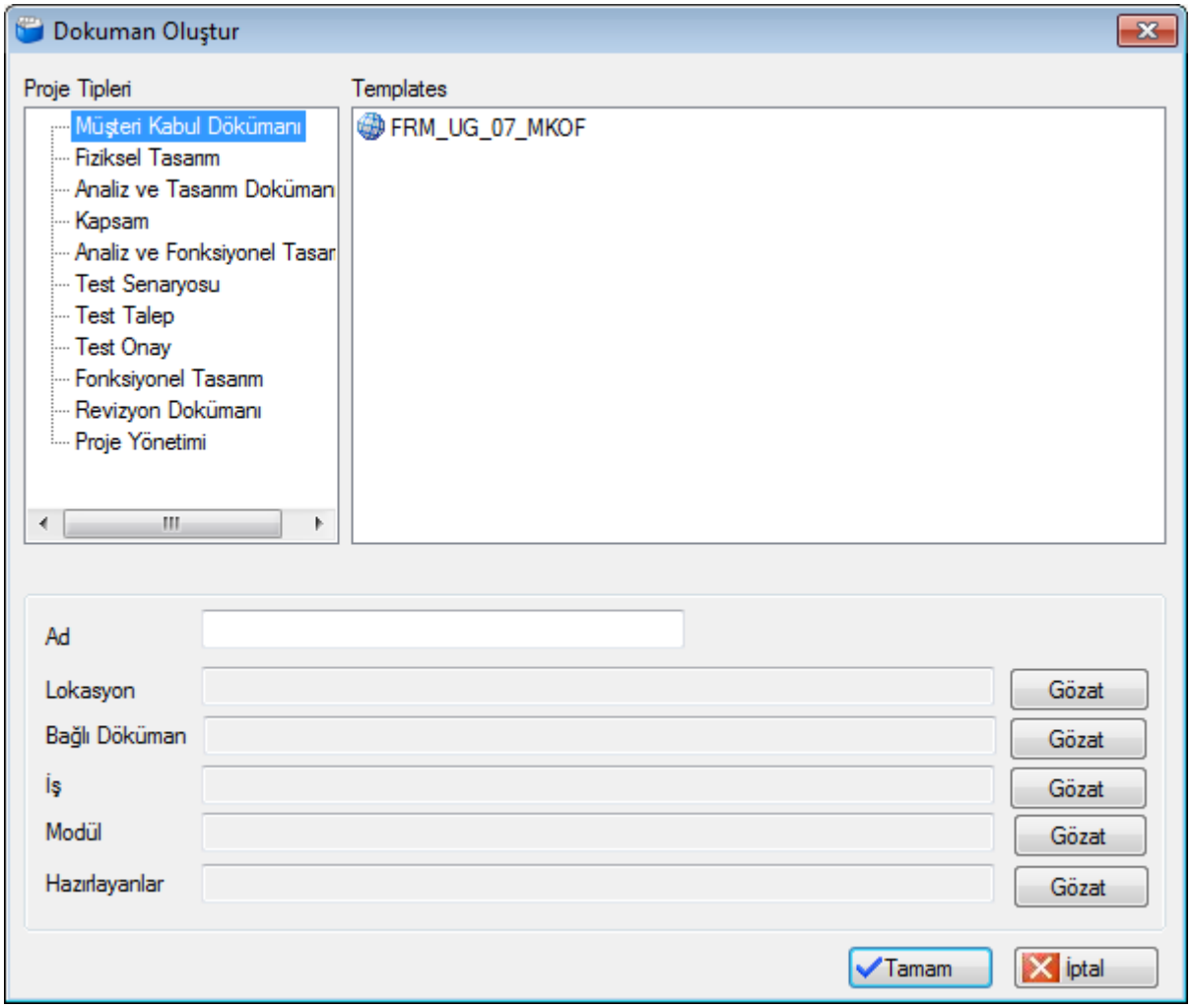


Şekil. 32. Resim Ekleme

- Başlık Sil: Şablona tanımlanmış başlığın silinmesimi sağlar.
- Çözümden Kaldır: Dokümanın eklenmiş olduğu çözümden kaldırılmasını sağlar.

3.5.3. Dokümanlar

Şablonlar oluşturulduktan sonra doküman oluşturma işlemine geçilebilir. Doküman oluşturmak için yine şablon oluşturmada yapıldığı gibi açık olan çözüme sağ tıklanarak yeni doküman ekle seçeneği seçilir. Şekil 33'te görüleceği gibi açılan yeni pencereye doküman ile ilgili bilgiler girilir. Hatırlanacağı üzere şablon oluşturulurken doküman tipi seçilmişti. Doküman oluşturabilmek içinde şablon seçmek gerekir. Bu şablonlar doküman tiplerine göre gruplanmış durumdadırlar. Kullanıcı ilgili doküman tipini seçtikten sonra sağ panelde o tipe tanımlanmış şablonlar listelenmektedir. İstenilen şablon seçildikten sonra sırasıyla, doküman adı,lokasyon,bağlı doküman,iş,Modül,hazırlayanlar bilgisi girilerek tamam tuşuna basılır ve seçilen şablondan yeni bir doküman oluşturulmuş olunur.



Şekil. 33. Doküman Oluşturma

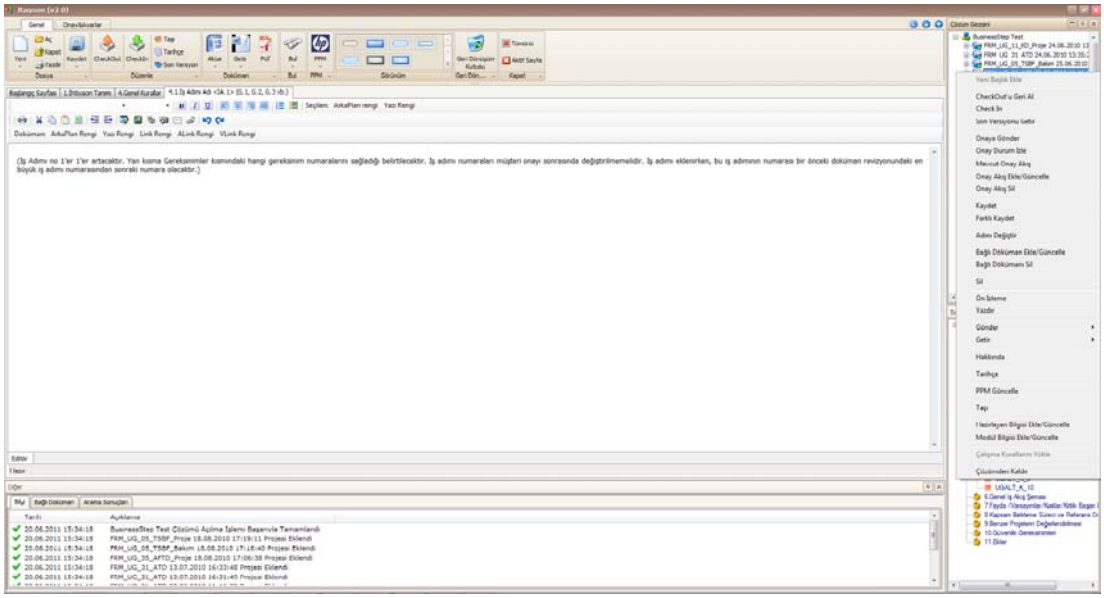
Yeni doküman oluştururken daha önce karşılaşmadığımız bazı bilgiler girilmesi istenmektedir. Bunlardan ilki iştir. Bilindiği gibi kurumlar işlerini bazı

proje yönetim araçlarıyla yönetmektedirler. Bu araçlarda proje paydaşları kaynak durumu ve işler takip edilmektedir. Yine bu uygulama vasıtasıyla hangi kaynak hangi işte çalışmakta olduğu bilgisi tutulmaktadır. İşte yaptığımız uygulama bu tarz uygulamalar ile entegre çalışabilecek kapasitededirler. Böylelikle yapılan işlere ait dokümanlar rahatlıkla takip edilebilmektedir.

Bir diğer bilgi modül bilgisidir. Modül kavramı tamamiyle oluşturulan dokümanları mantıksal gruplara ayırmak için kullanılmaktadır. Her kullanıcı doküman oluşturmadan önce çalışığı modülü seçer, bu sayede dokümanlar gruplara ayrılırken modül bilgiside seçilebilir.

Hazırlayanlar bilgisi bir doküman birden fazla kaynak tarafından hazırlanıyor ise diğer kaynaklarında kayıt altına alınması ve bu sayede doküman bilgisi bölümde görüntülenmesini sağlar.

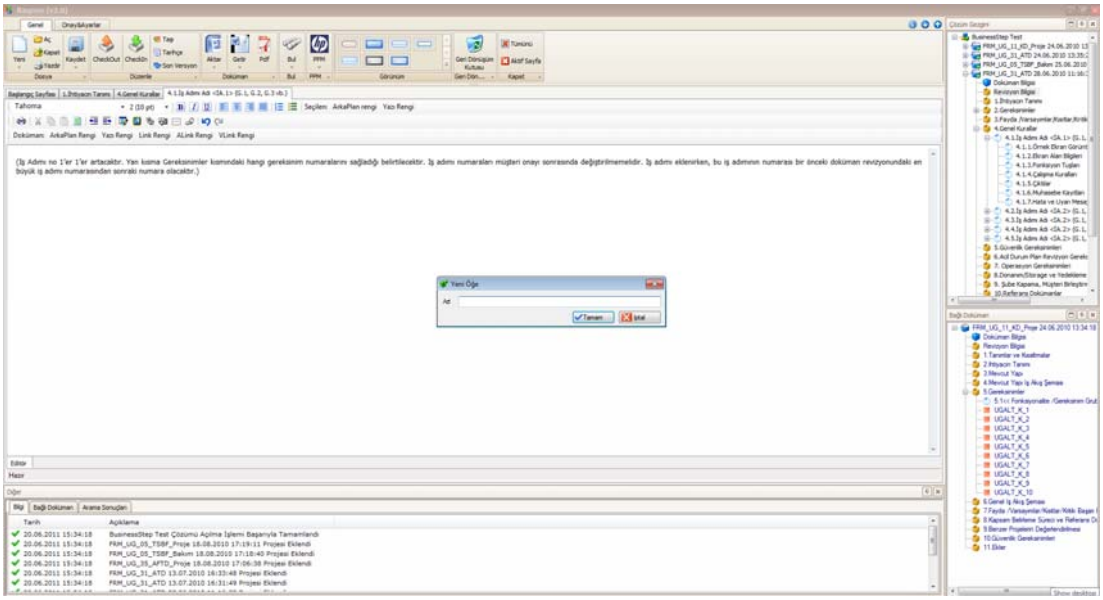
Dokümanlar üzerinde bir dizi işlem yapabilmek mümkündür. Aşağıda doküman üzerinde yapılabilecek işlem setini başlıklar halinde anlatacağım. Şekil 39'da doküman işlem fonksiyonlarını görebilirsiniz.



Şekil. 34. Doküma İşlem Fonsiyonları

Doküman İşlem Seti:

• Yeni Başlık Ekle : Dokümana yeni bir ana başlık eklemek amacıyla kullanılır. Bu işlem tıklandığında çıkan ekrana başlık adı girilir ve yeni başlık dokümana eklenir. Şekil 4.3'te yeni başlık eklemeye bir örnek bulabilirisiniz.



Şekil. 35. Yeni Başlık Ekleme

• Check-Out : Uygulamamızda bir doküman aynı anda yanlızca bir kişi tarafından yazılabilir. Dokümanı yazacak kullanıcı dokümanı check-out ederek dokümanı üzerine alır. Böyle bir durumda aynı dokümanı yazmak isteyen kullanıcı dokümanın başkası tarafından check-out edildiğini öğrenir ve yanlızca okuma amaçlı açabilir.

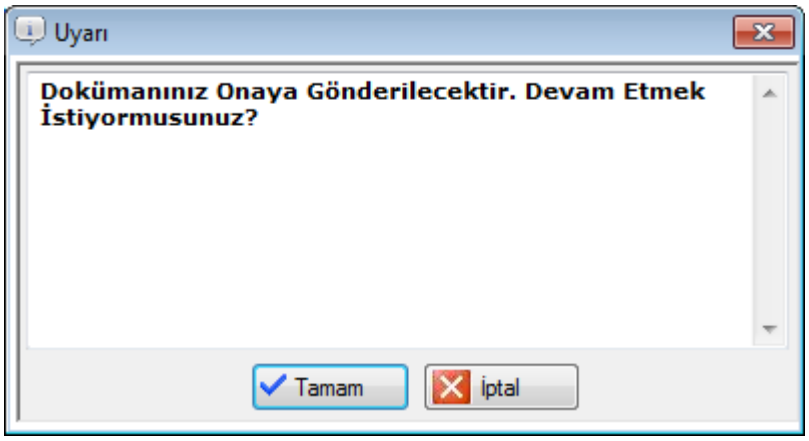
• Check-In: Doküman kullanıcı tarafından check-out edildikten sonra ilgili kullanıcı tarafından yazımına başlanır. Kullanıcının doküman ile işi bittikten sonra dokümanı check-in eder ve böylelikle doküman serbest kalmış olur. Böyle bir durumda bir başkası dokümanı check-out ederek dokümanı yazabilir.

• Check-Out'u Geri Al: Uygulamamızın önemli özelliklerinden biri her doküman check-out edildikten sonra bir önceki versiyon kayıt altına alınır. Bu sayede kullanıcı dokümanı check-out edip gerekli işlemleri yaptıktan sonra eğer check-out'u geri al der ise doküman olduğu gibi check-out'u edilmeden önceki versiyonuna döner.

• Son Versiyonu Getir: Bilindiği gibi doküman check-out edilmiş ise yanlızca okuma amaçlı dokümanı açabiliyorsunuz. Böyle bir durumda sizin açtığınız

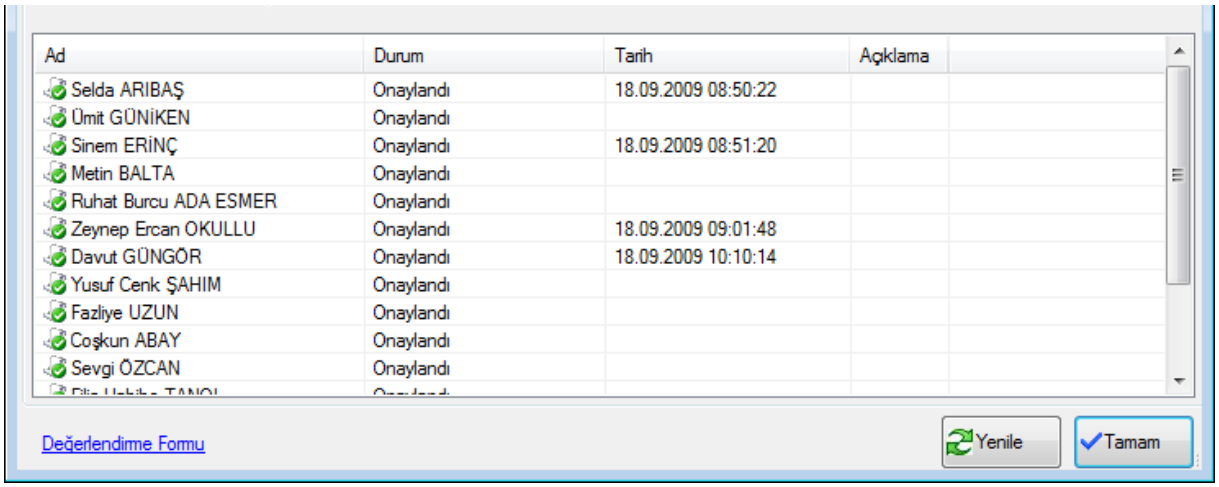
doküman halen başkası tarafından güncelleniyor olabilir. İşte dokümanın en güncel halini getirebilmeniz için son versiyonu getir özelliğini kullanırsınız.

- Onay'a Gönder: Daha önceden belirttiğim gibi uygulamamız herhangibir onay veya iş akış uygulaması ile entegre çalışabilecek durumdadır. Onay'a gönder özelliğide dokümanı ilgili uygulama vasıtasıyla onaya göndermek amacıyla kullanılır.



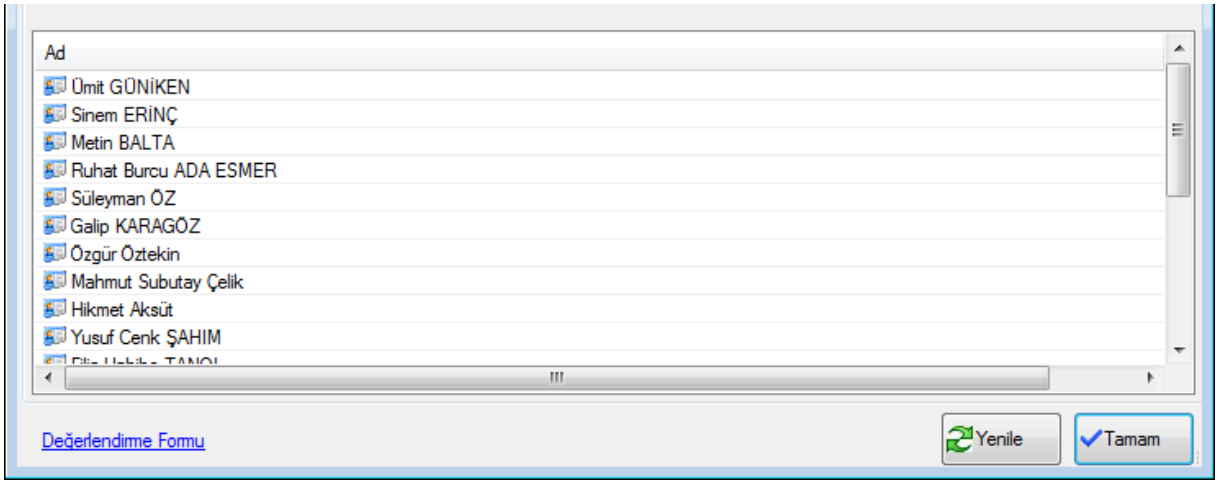
Şekil. 36. Onaya Gönderme

- Onay Durum İzle: Onay'a göndermiş olduğunuz dokümanın onay durumunu izlemek amacıyla kullanılır.



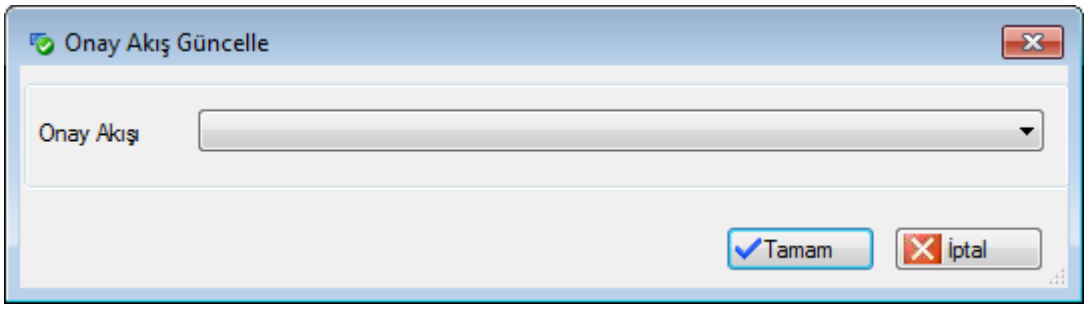
Şekil. 37. Onay Durum İzleme

• Mevcut Onay Akış: Dokümanın şablonunun üzerinde tanımlı olan onay akışını görüntülenmesi sağlar. Bilindiği gibi şablon üzerine tanımlanmış bilgilerin tümü dokümana aynı şekilde yansır.



Şekil. 38. Mevcut Onay Akış

• Onay Akış Ekle Güncelle: Dokümanın şablonundan ulaşmış onay akışı dışında herhangibir başka onay akış kullanılmak istenirse bu özellik kullanılabilir.



Şekil. 39. Onay Akış Ekle/Güncelle

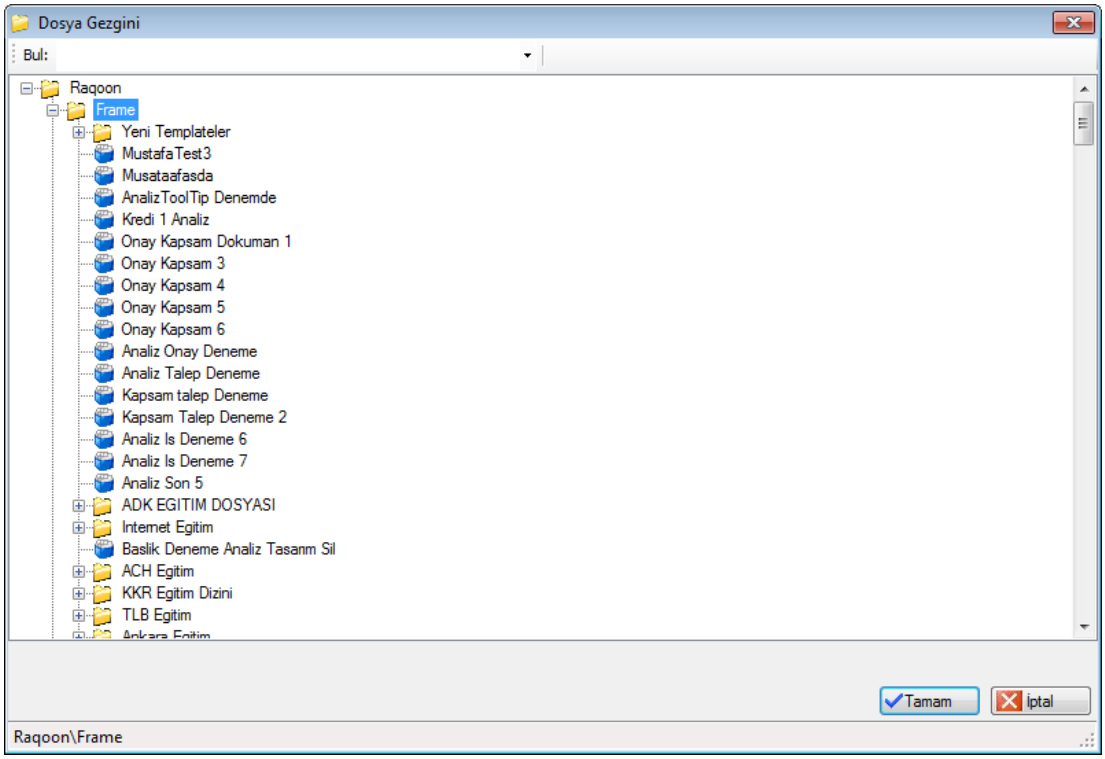
• Onay Akış Sil: Doküman üzerinde tanımlı onay akış tanımı kaldırılır.

• Kaydet: Adındanda anlaşılacağı gibi dokümanı kayıt etmek amacıyla kullanılır. Fakat bu durum çoğunlukla check-in ve check-out ile karıştırılmaktadır. Kaydet denildiği zaman check-out edilmiş ve bilgisayarınızın hafızasında duran bilgiler veri tabanına yazılır.Bu sayede herhangibir sebepten dolayı bilgisayarınız kapanır ve bellek problemi yaşar ise o zaman kadar yazmış olduğunuz bilgiler kaybolmaz.

• Farklı Kaydet: Dokümanınızı farklı bir isimle aynıo özelliklerde kayıt etmenizi sağlar.

• Adını Değiştir: Kullanıcının dokümanının adını değiştirmesini sağlar.

• Bağlı Doküman Ekle/Güncelle: Kullanıcının dokümanı ile ilişkili dokümanını seçmesine veya değiştirmesini yardımcı olur.



Şekil. 40. Bağlı Doküman Ekle/Güncelle

• Bağlı Dokümanı Sil: Kullanıcının öncede tanımlamış olduğu dokümanı ile ilişkili doküman tanımını kaldırmasını sağlar.

• Sil: Dokümanı silmenizi sağlar.

• Ön İzleme: Adındanda anlaşılağı gibi dokümanın tamamının izlenmesini sağlar. Bildiğiniz gibi uygulamamızda doküman başlılarının herbiri ayrı editörler vasıtasıyla yazılmaktadır. Bunları bir bütün olarak izlemek amacıyla ön izleme fonksiyonu kullanılır.

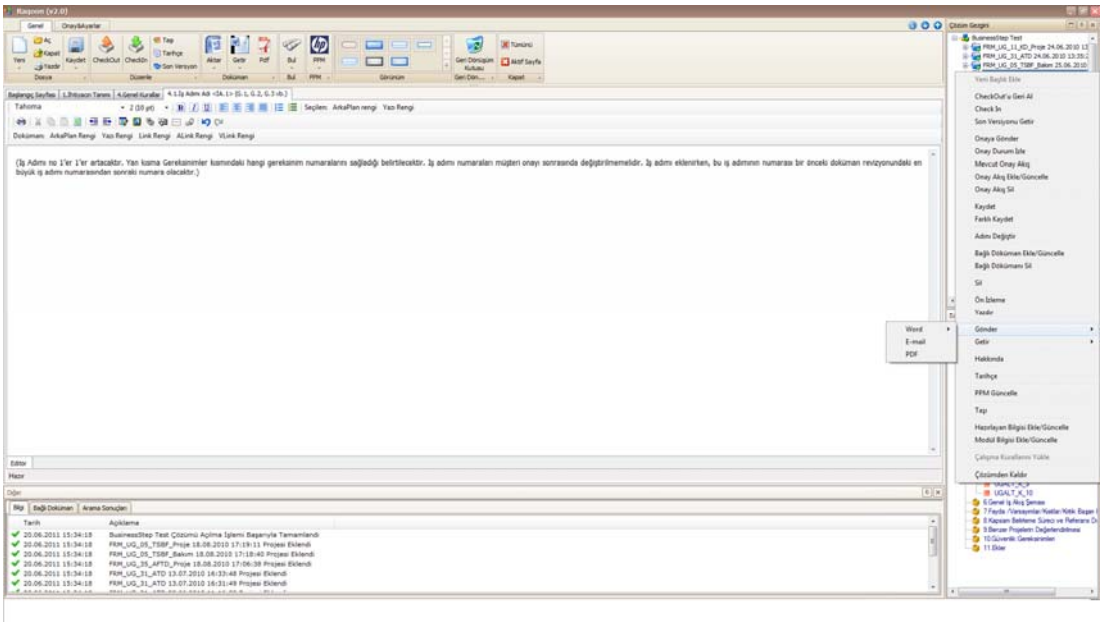
• Yazdır: Dokümanı printer'da yazdırmak amacıyla kullanılır.

• Gönder: Gönder özelliği üç ayrı özellik içermektedir. Şekil 4.4'te gönder özelliğinin diğer özelliklerini görebilirsiniz.

o Word: Dokümanı Word'e aktarmanızı sağlar.

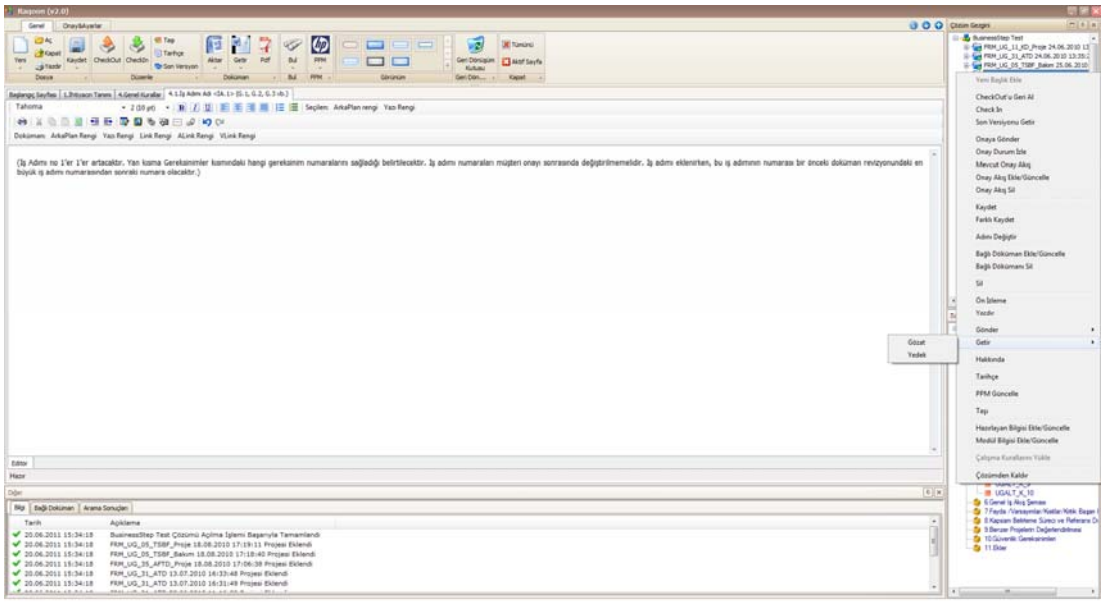
o Email: Dokümanı Email olarak göndermenizi sağlar.

o Pdf: Dokümanınızı pdf dosyasına aktarmanızı sağlar.



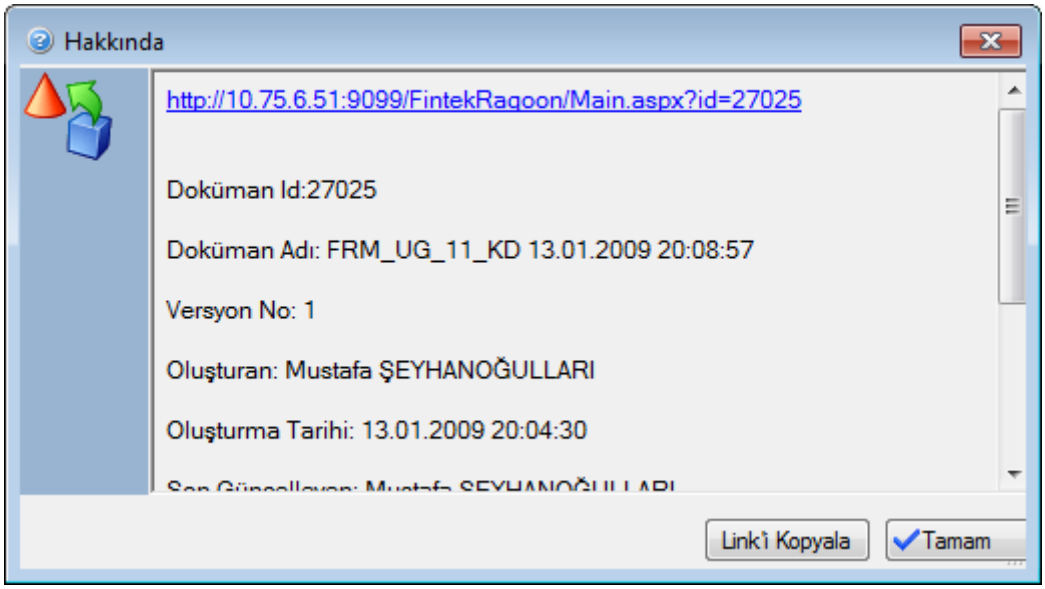
Şekil. 41. Doküman Aktarımı

- Getir: Getir özelliğide gönder özelliği gibi farklı özelliklerden oluşmaktadır. Bunlar;
- Gözat: Daha önce nitelikli olarak word'e aktarılmış dokümanınızı uygulama içerisine alır.
- Yedek: Uygulamamız her 15 dakikada bir (Parametrik bir değerdir, dilediğiniz değer set edilebilir) bilgisayarınızın temp dizinine yedek almaktadır. Bu özellik alınan yedekten geri getirmek amacıyla kullanılabilir.



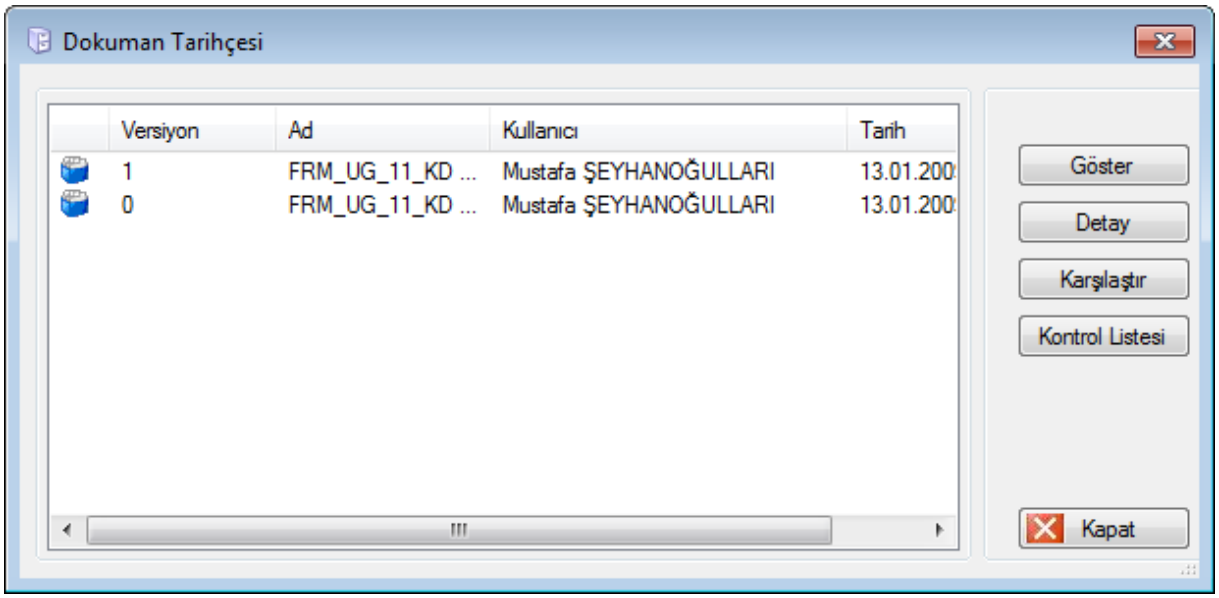
Şekil. 42. Yedek Alma

- Hakkında: Dokümanınız hakkında genel bilgileri görüntüler.



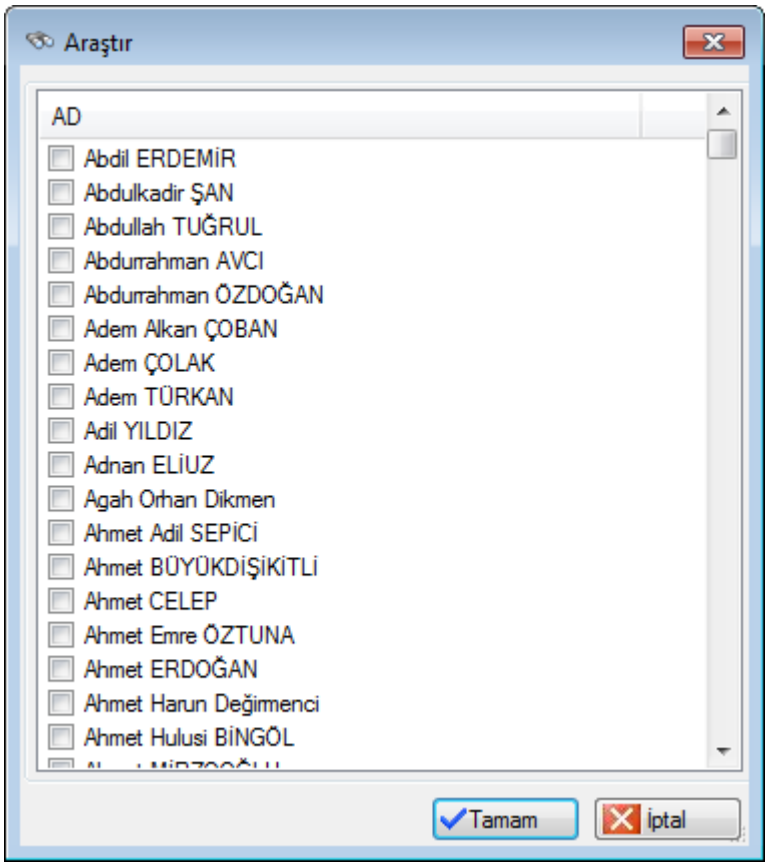
Şekil. 43. Hakkında

- Tarihçe: Bilindiği gibi dokümanlar her check-out edildiklerinde versiyonlanıyorlardı. İşte tarihçe özelliği dokümanın eski versiyonlarına ulaşmanızı sağlıyor. Ayrıca iki ayrı versiyonu karşılaştırma yeteneğinede sahip bir ekrandır. Şekil 49'da tarihçe ekran örneği görebilirsiniz.



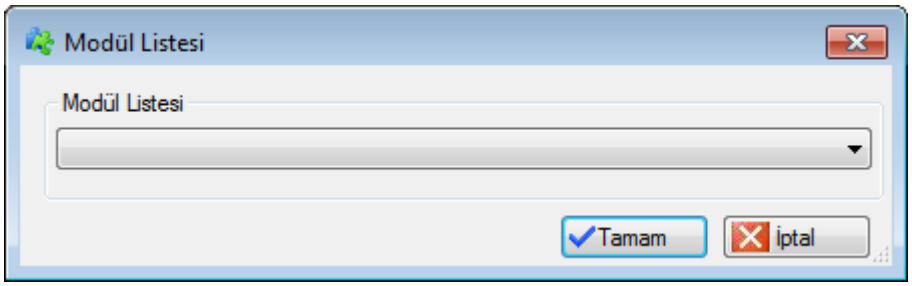
Şekil. 44. Doküman Tarihçesi

- Taşı: Dokümanınızı farklı bir klasöre taşımanızı sağlar.
- Hazırlanan Bilgisi Ekle/Güncelle: Daha önceden belirttiğim gibi dokümanlar birden fazla kişi tarafından hazırlanmış ise hazırlanan diğer kişilerde eklenebiliyordu. İşte bu özellik sayesinde doküman oluşturulduktan sonra bile hazırlayan kişi bilgisi eklenebilmektedir.



Şekil. 45. Hazırlayan Bilgisi

- Modül Bilgisi Ekle/Güncelle: Dokümanı oluştururken belirtilen modül bilgisi güncellenebilir veya eklenmemiş ise eklenebilmektedir.



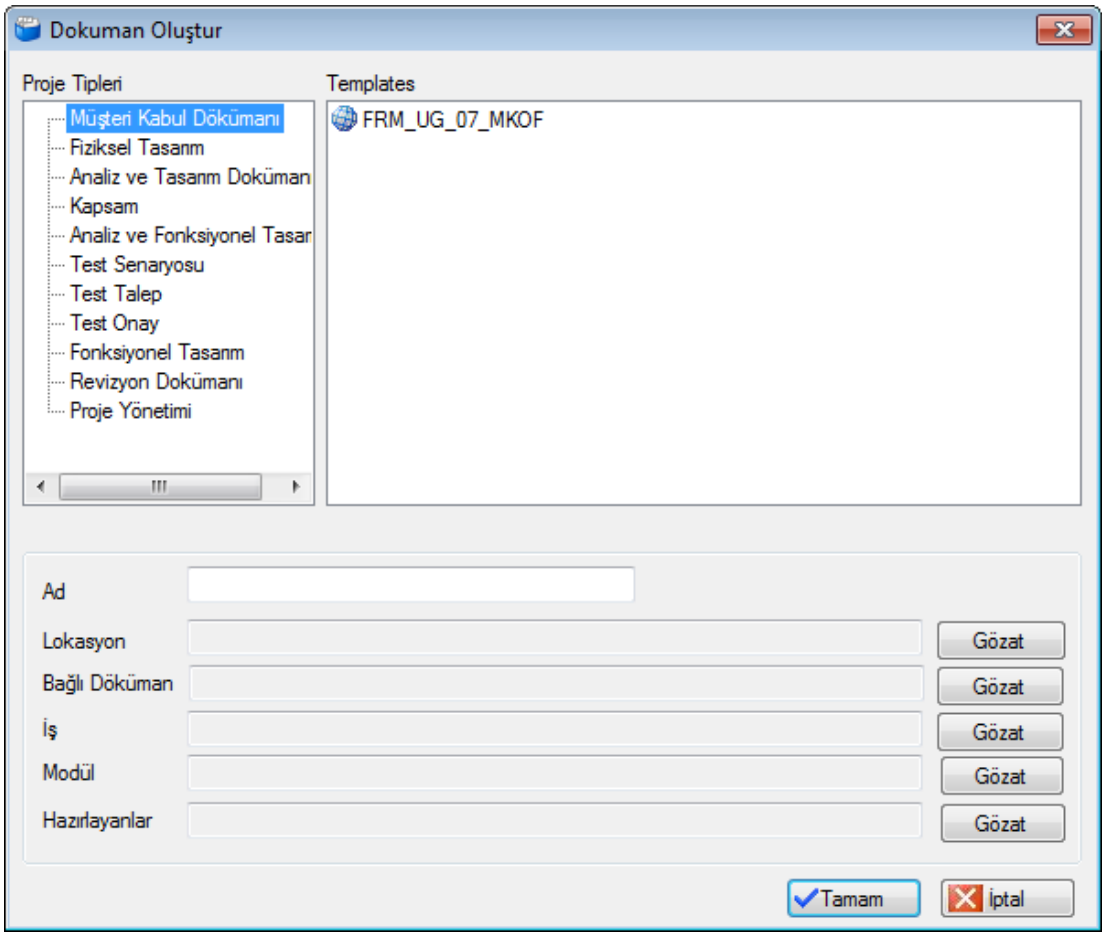
Şekil. 46. Modül Listesi

- Çözümden Kaldır: Dokümanın eklenmiş olduğu çözümden kaldırılmasını sağlar.

3.5.4. Çözümler

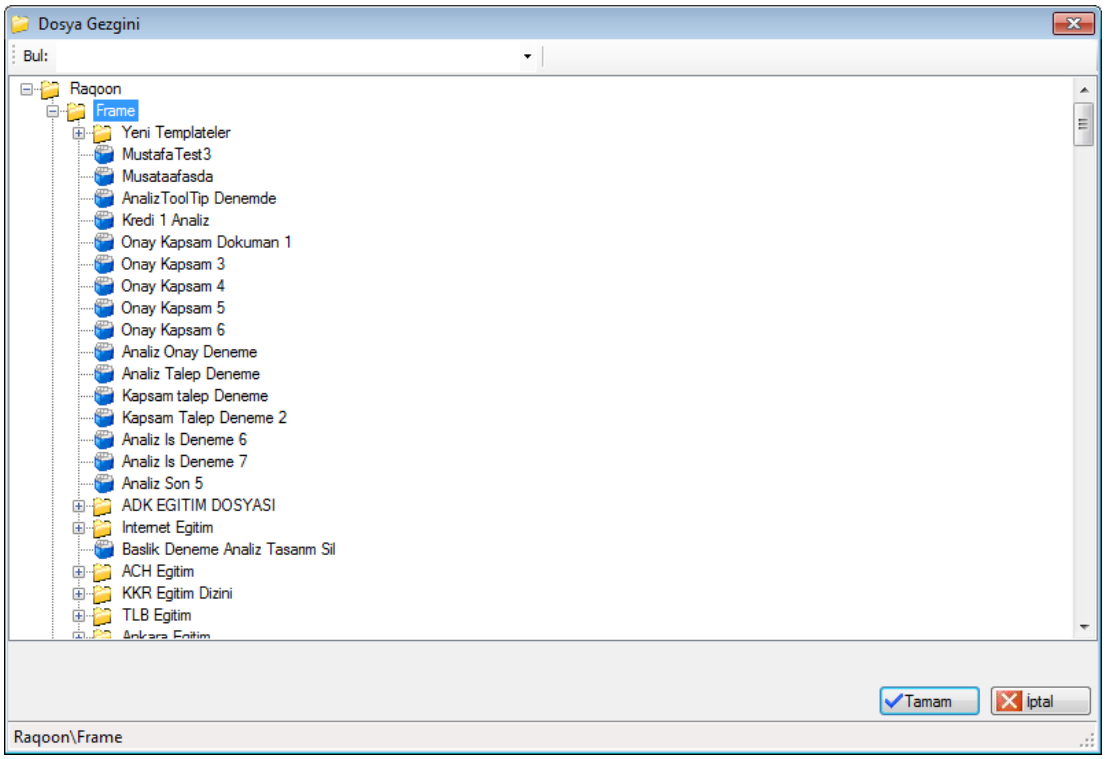
Çözüm İşlem Seti:

- Yeni Doküman Ekle: Yeni bir doküman oluşturmak için bu fonksiyonalite kullanılır. Bu özelliğe tıklandıktan sonra yeni bir ekran açılır. Şekil 52'de bu ekranı görmek mümkündür.



Şekil. 47. Doküman Oluştur

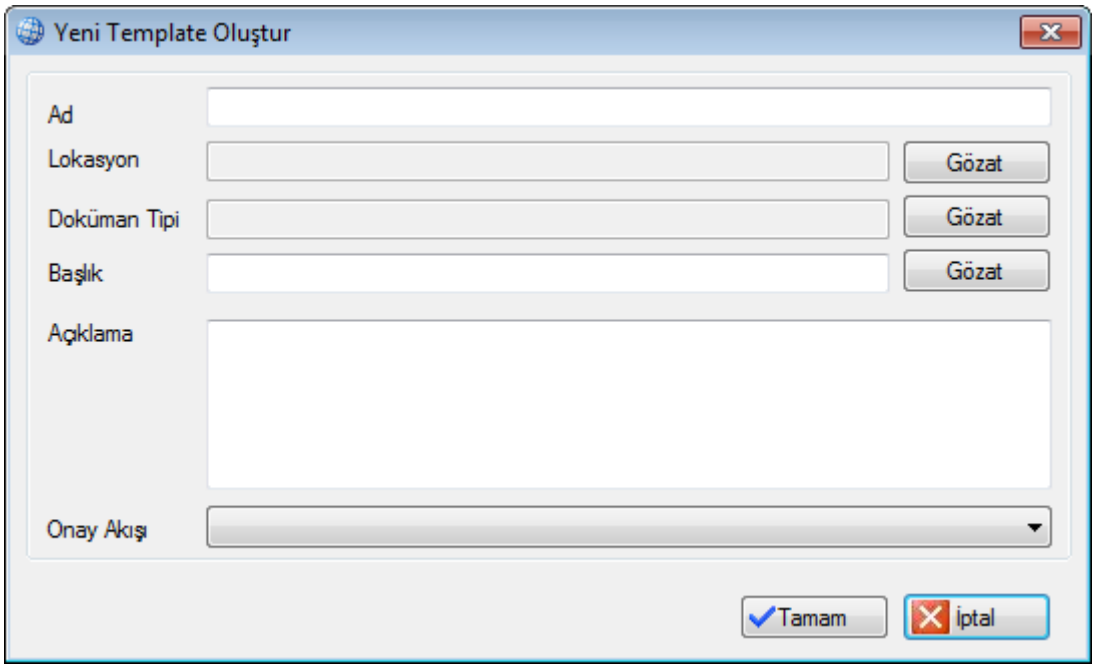
Açılan ekranda öncelikle doküman tipi seçilmelidir. Daha sonra bu doküman tiplerine ait şablonlardan biri seçilerek seçilen şablona göre doküman oluşturulacaktır. Şablon seçildikten sonra lokasyon seçimine geçilmelidir, lokasyonlar veri tabanında oluşturulmuş mantıksal alanlardır. Şekil 53'de lokasyon seçimini görebilirsiniz.



Şekil. 48. Lokasyon Seçimi

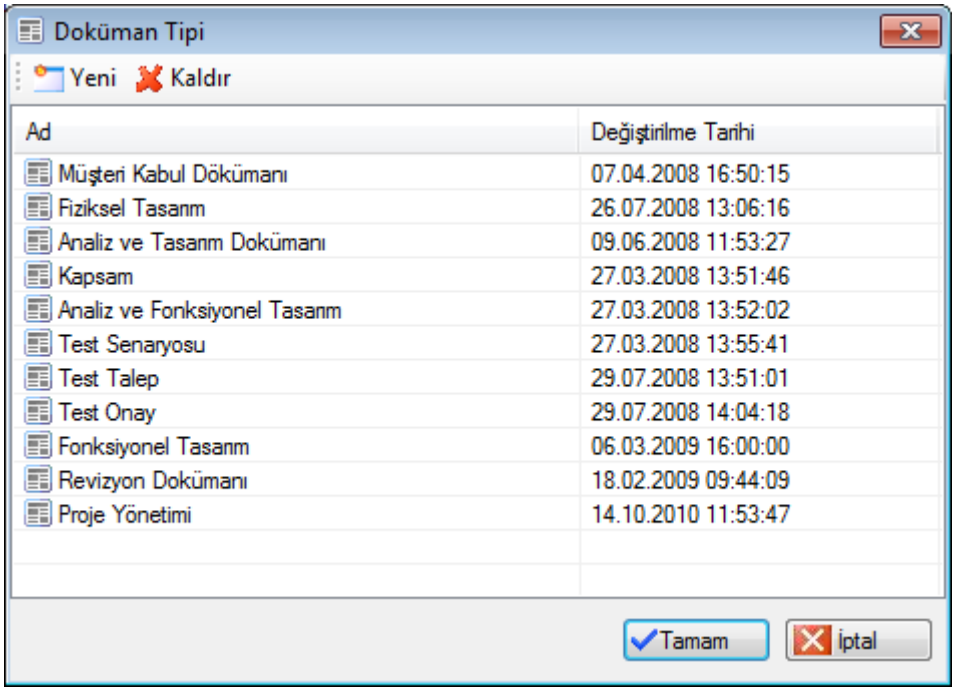
Lokasyon seçiminin hemen arkasından eğer varsa bağlı doküman seçilmelidir. Bağlı doküman daha önceden de açıkladığımız gibi dokümanın ilişkili olduğu dokümandır. Bağlı doküman bilgisi seçiminin hemen arkasından sırasıyla eğer var is iş seçilir daha sonra modül seçimi ve hazırlayan bilgisi seçilir ve doküman oluşturulur.

- Yeni Template Ekle: Yeni bir şablon oluşturulmak istendiğinde bu özellik kullanılır. Bu özelliğe tıklandıktan sonra yeni bir ekran açılır. Şekil 1.2'de bu ekranı görmek mümkündür.



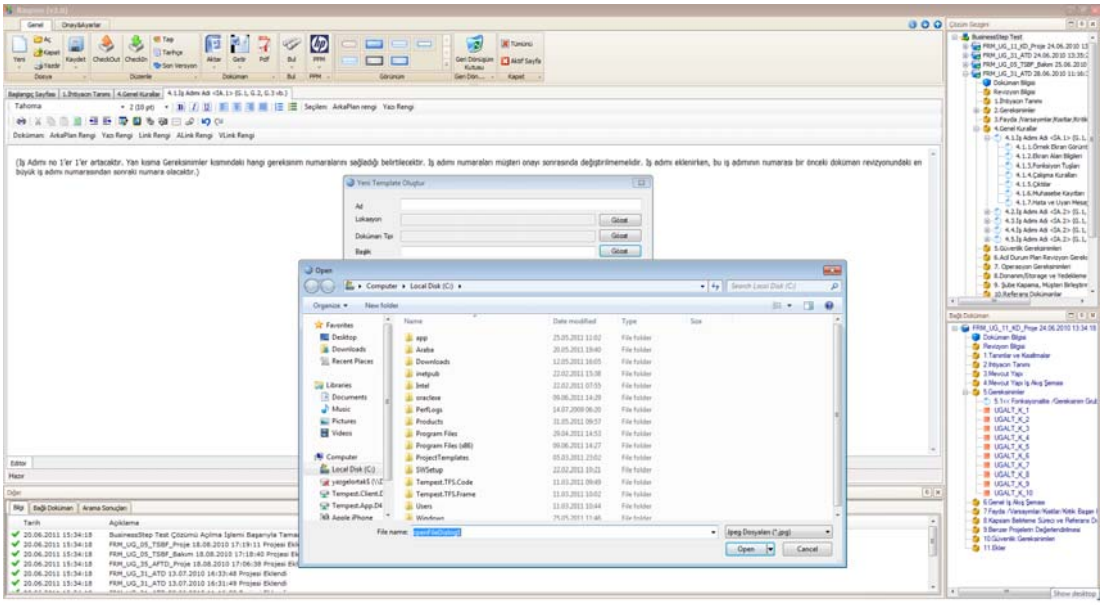
Şekil. 49. Yeni Template Ekle

Yeni şablon oluşturmak için öncelikle şablona bir isim verilir. Daha sonra yine doküman oluşturmada olduğu gibi şablon için bir lokasyok seçilir. Lokasyon seçimi tamamlandıktan sonra doküman tipi seçimine geçilir. Doküman tipi seçimi için Şekil 50'de görüldüğü gibi bir ekran açılır.



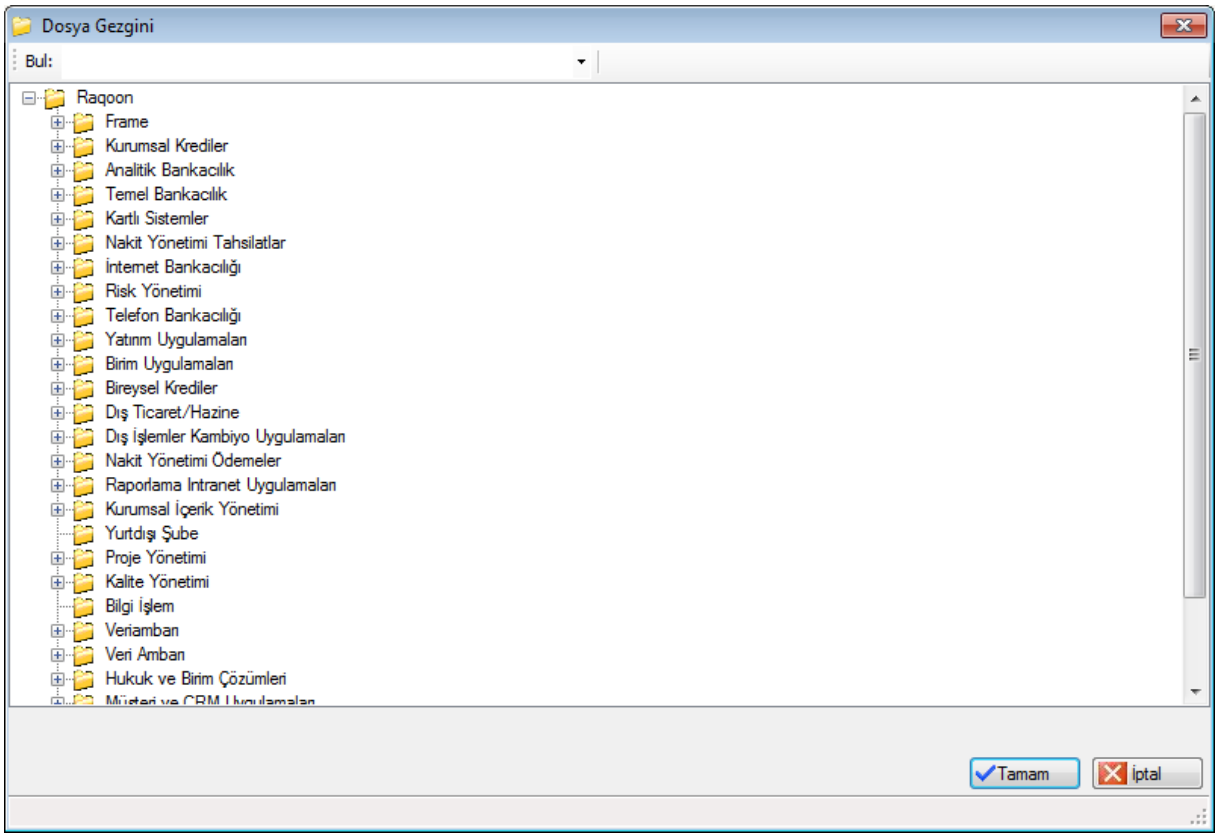
Şekil. 50. Doküman Tipi Seçimi

Kullanıcı bu ekrandan mevcut bir doküman tipi seçebileceği gibi dilerse yeni bir doküman tipide ekleyebilir. Doküman tipi seçiminin ardından kullanıcı başlık resmi seçmelidir. Uygulamamızda doküman başlıkları resimlerden oluşmaktadır. Bu nedenden dolayı başlık eklerken kullanıcı tarafından bir resim seçilir. Şekil 51'de başlık ekleme örneği görebilirsiniz.



Şekil. 51. Başlık Ekleme

- Mevcut Doküman Ekle: Kullanıcı daha önceden kayıt edilmiş bir dokümanı eklemek için bu özelliği kullanmaktadır. Mevcut doküman ekle özelliği tıklandığı zaman Şekil 52'de görüldüğü gibi kalsör ekranı açılır ve kullanıcı kalsörler içersinden istedikleri dokümanı seçerek açarlar.



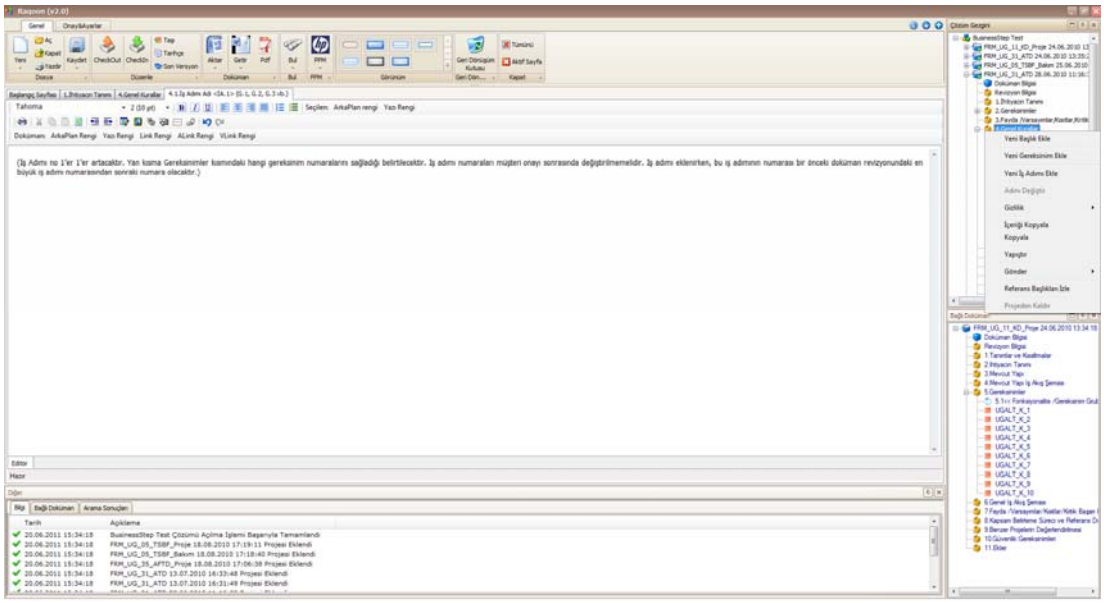
Şekil. 52. Mevcut Doküman Ekleme

- Kaydet: Kaydet özelliğine tıklandığında çözüme eklenmiş olan tüm dokümanlar veri tabanına kayıt edilir.
- Adını Değiştir: Adındanda anlaşılacağı gibi çözümün adını değiştirmek amacıyla kullanılır.
- Çözümü Kapat: Kullanıcının açmış olduğu çözüm kapatılır.

3.5.5. Başlıklar

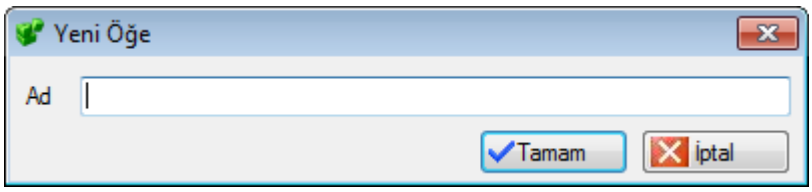
Şablonlara ekleyebileceğimiz dört tip başlık vardır. Bunlar ana başlık,alt başlık ve sabit başlıklardır.

- Ana Başlık : Dokümanda level 1'de bulunan başlıklardır. Yani dokümanın en üst seviyesinde bulunurlar.



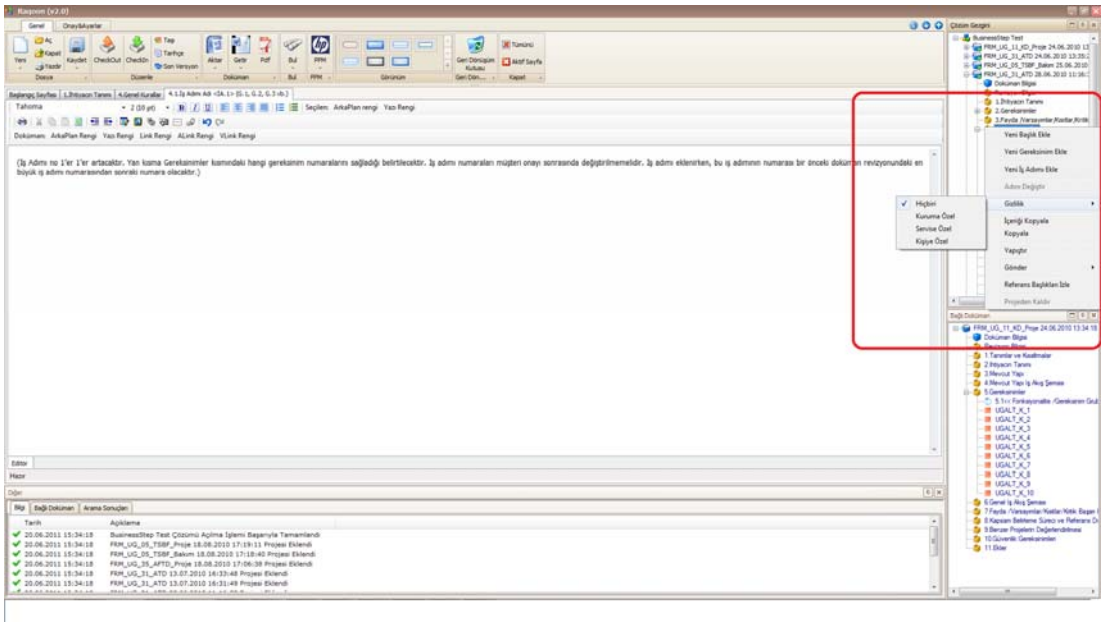
Şekil. 53. Ana Başlık

- Yeni Başlık Ekle: Ana başlıklara alt başlık eklemek amacıyla kullanılır.



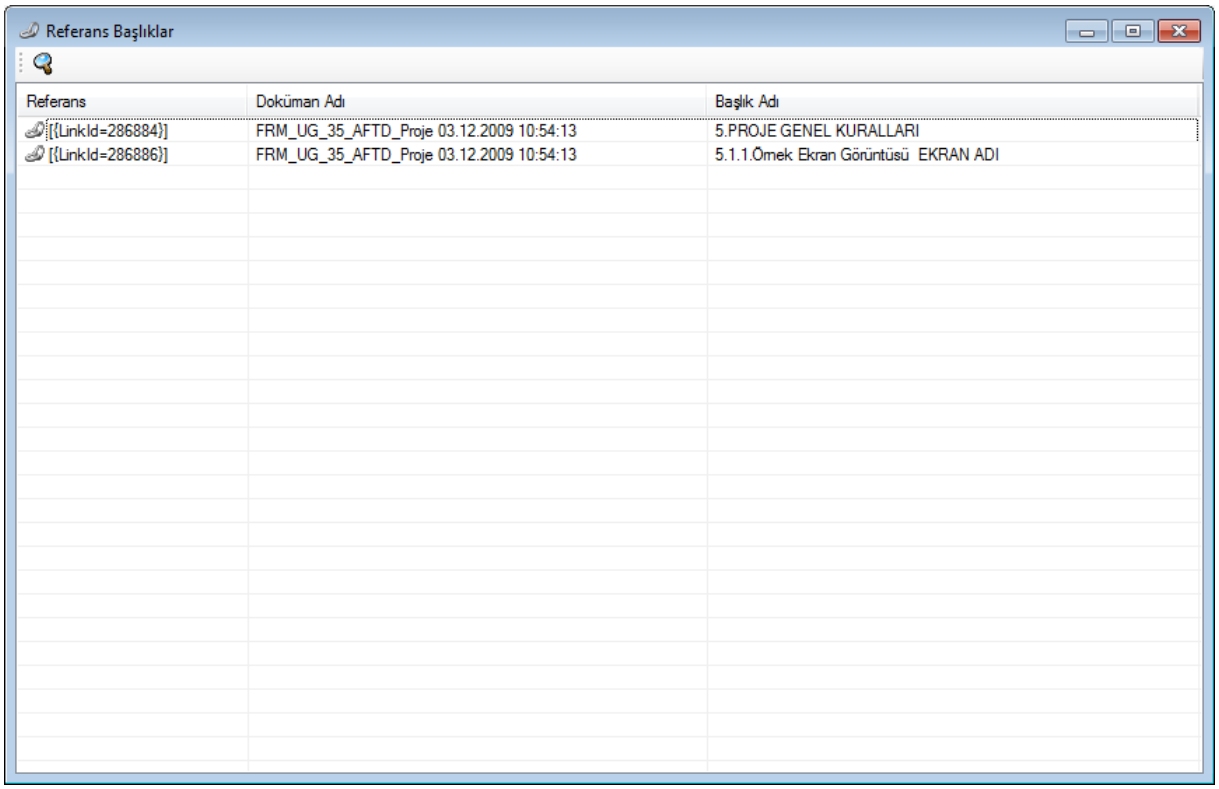
Şekil. 54. Yeni Başlık Ekleme

- Yeni Gereksinim Ekle: Ana başlıklara gereksinim başlık eklemek amacıyla kullanılır. Bu fonksiyon tıklandıkta sonra kaç adet gereksinim ekleneceğini soran bir ekran ile karşılaşılır. Sayı girildikten sonra gereksinimler başlıklara eklenir.
- Gizlilik: Başlık içerileri dilenirse kullanıcıya göre gizli tutlabilmektedir. Uygulamamızda kuruma özel,departmana özel ve kişiye özel şeklinde gizlilik set edilebilir.

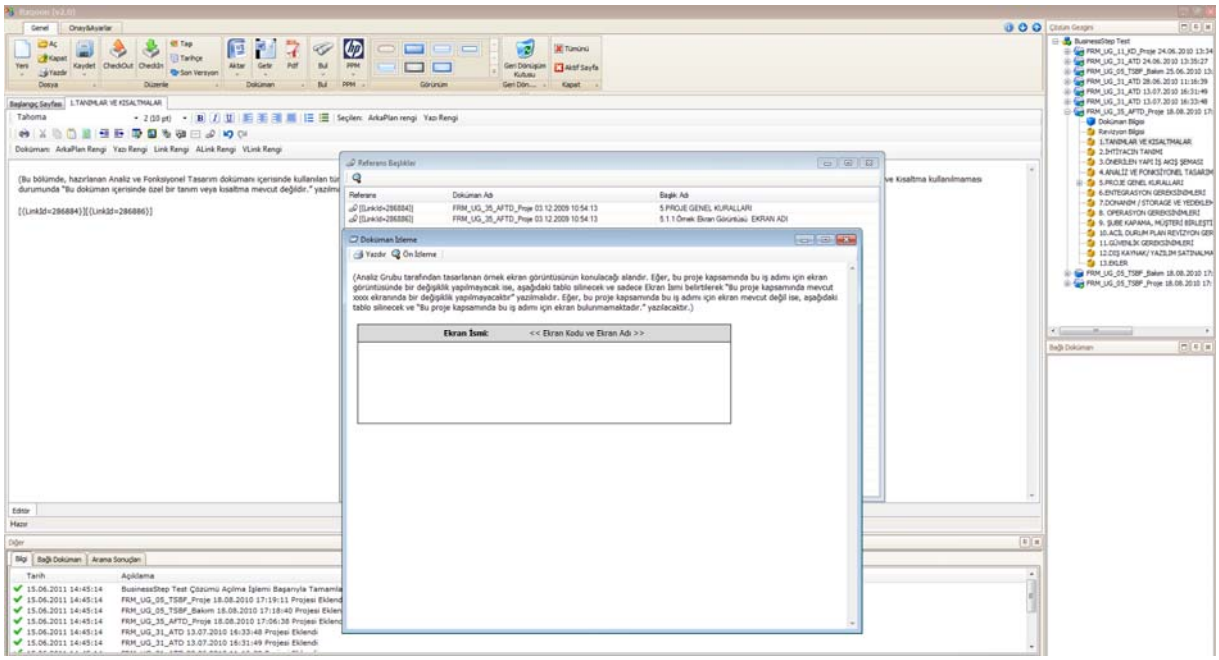


Şekil. 55. Gizlilik

- İçeriği Kopyala: Seçilen başlığın içeriğini text olarak kopyalar ve kullanıcı dilerse başka bir başlığın içeriğine yapıştırmak suretiyle ilgili datayı kullanabilir.
- Kopyala: Kopyala fonksiyonu içeriği kopyalamadan farklı olarak objeyi kopyalar. Yani eğer kullanıcı bir başlık,alt başlık veya gereksinim kopyalar ise başka bir dokümana paste etmek suretiyle o objeyi başka dokümandada kopyalayabilir. Bu özelliğin sağladığı en büyük kolaylık ise kopyalanan başlık altındaki tüm objeler ile beraber kopyalanır ve yapıştırılabilinir.
- Yapıştır: İçeriği kopyala veya kopyala ile kopyalanmış obje veya içerikin dilenilen yere yapıştırılmasını sağlar.
- Gönder: Başlığın içeriğinin email olarak gönderilmesini sağlar.
- Referans Başlık İzle: Eğer başlığa referans başlık veya başlıklar eklenmiş ise hem onların neler olduğu görüntülenebilir hemde içerikleri izlenebilir.



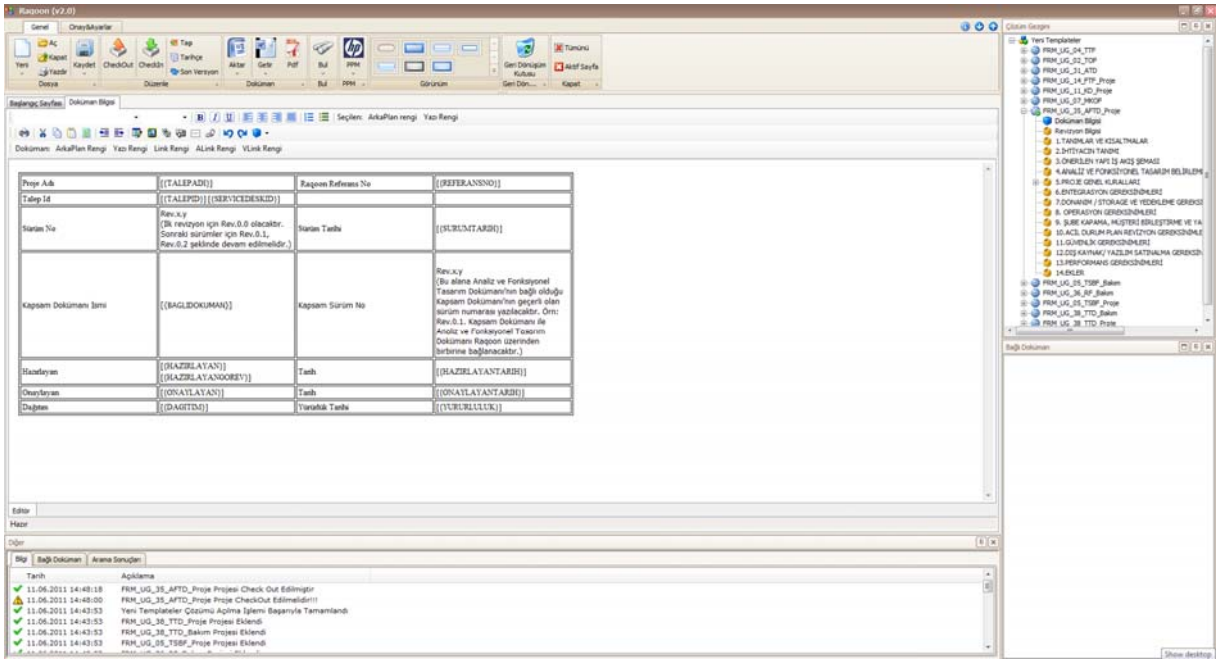
Şekil. 56. Referans Başlık İzle



Şekil. 57. Referans Başlık Görüntüle

- Alt Başlık: Ana başlıkların alt seviyelerinde bulunan başlıklardır. Alt başlıklar dilediğiniz kadar alt seviyede bulunabilir. Alt başlıklar içersinede alt başlık eklenebilmektedir. Alt başlıkların işlem seti yine ana başlıktakilerle aynıdır.

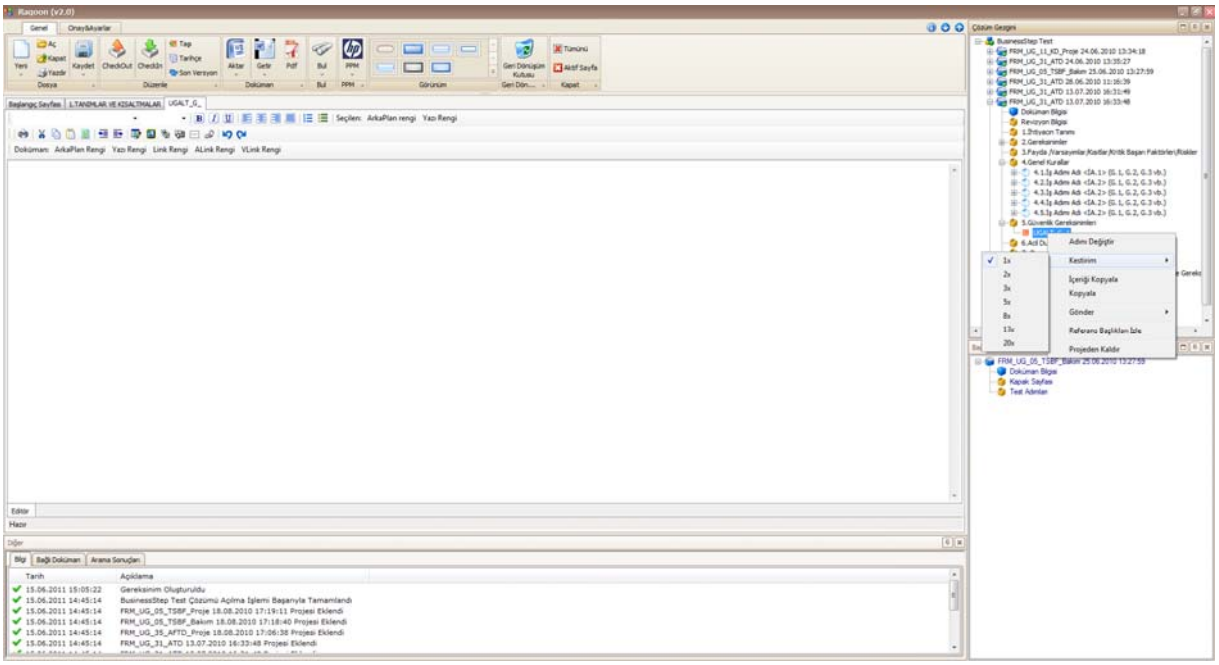
- Sabit Başlık: Sabit başlıklar uygulamadaki özel başlıklardır. Uygulamanın script dili kullanılarak içersine scriptler yazılır. Dokümanınız açılırken uygulama sabit başlıkları okuyarak gerekli bilgileri sabir başlık üzerine yazar. Şekil 58'de sabit başlığa bir örnek bulabilirsinz.



Şekil. 58. Sabit Başlık

- Gereksinim Başlık: Uygulamamızda gereksinimler ayrı başlık tipleridir. Bunun nedeni gereksinim başlıklarının diğer başlıklardan ayrı özelliklerde olmasından dolayıdır.

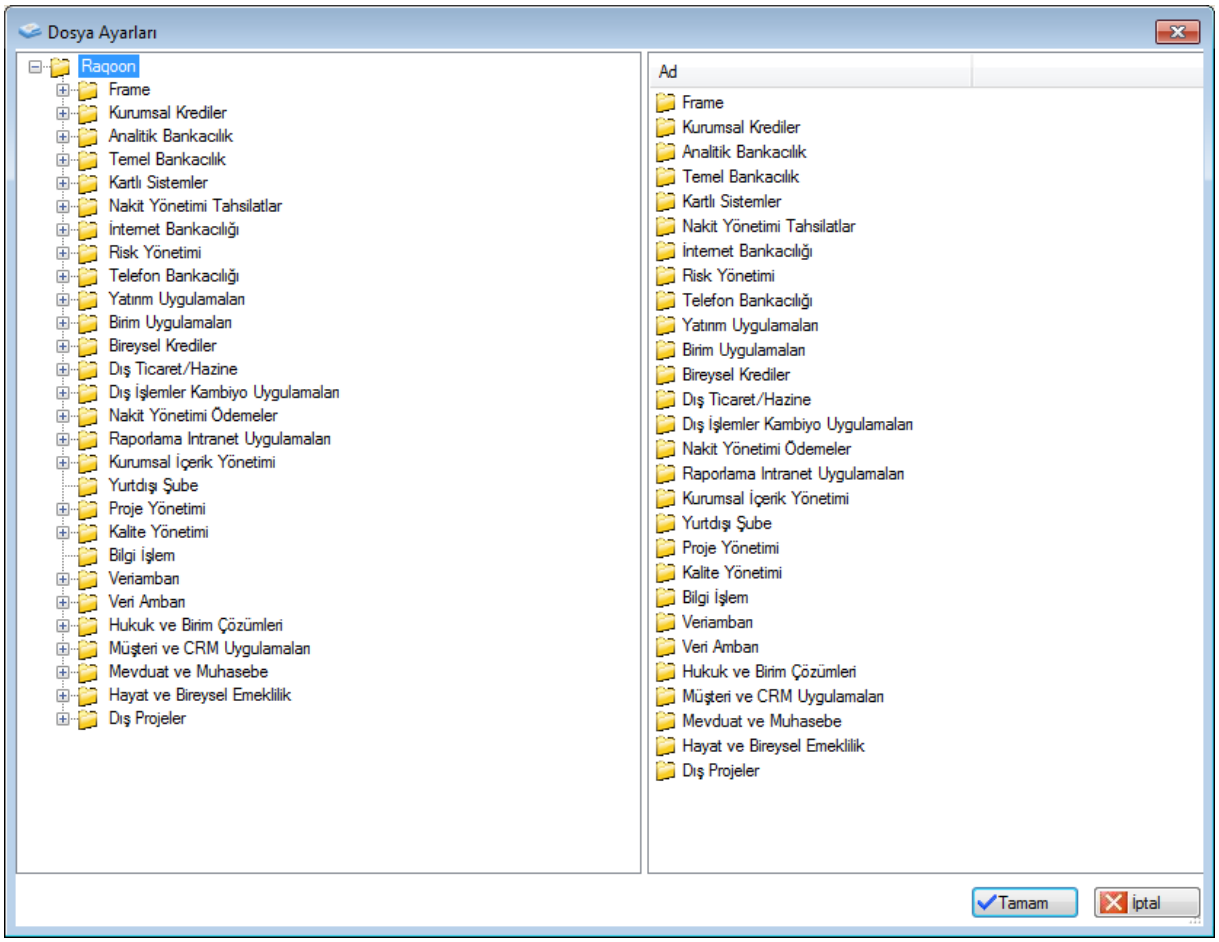
Gereksinim başlıkta işlem seti olarak ana başlıktan farklı olarak Kestirim özelliği bulunmaktadır. Kestirim özelliği girilen gereksinimin zorluk derecesini belirtmektedir. Uygulama geliştirme süreçlerinde yapılan büyük projelerde kestirim önemli bir faktör olduğu için böyle bir özellik eklenmiştir.



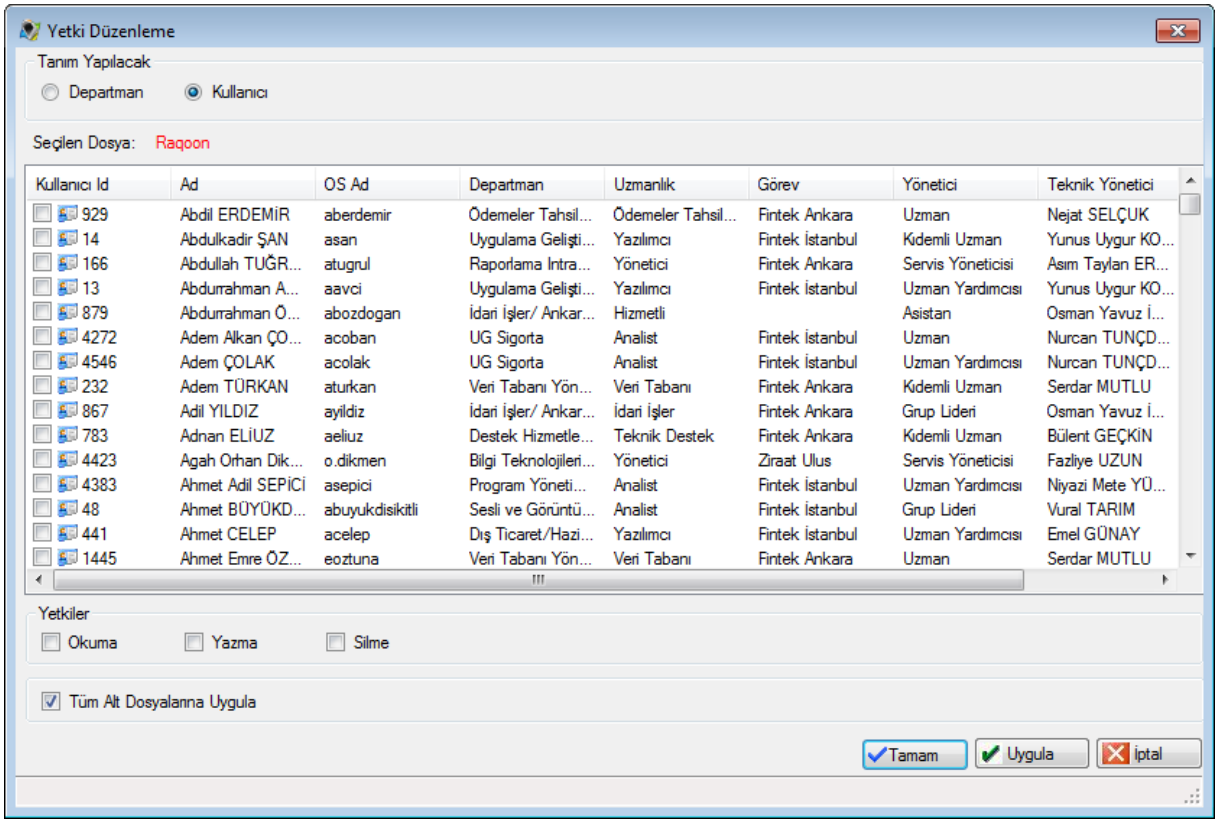
Şekil. 59. Gereksinim Başlık

3.5.6. Diğer İşlem Setleri

- Klasör Ayarları: Bilindiği gibi dokümanlar sanal klasörlerde saklanmaktadır. Bu klasörlerin yönetilebilmesi ve güvenlik amacıyla yetki verilebilmesi amacı ile klasör ayarları penceresi kullanılmaktadır. Şekil 60'ta klasör ayarları ekranını görebilirsiniz.



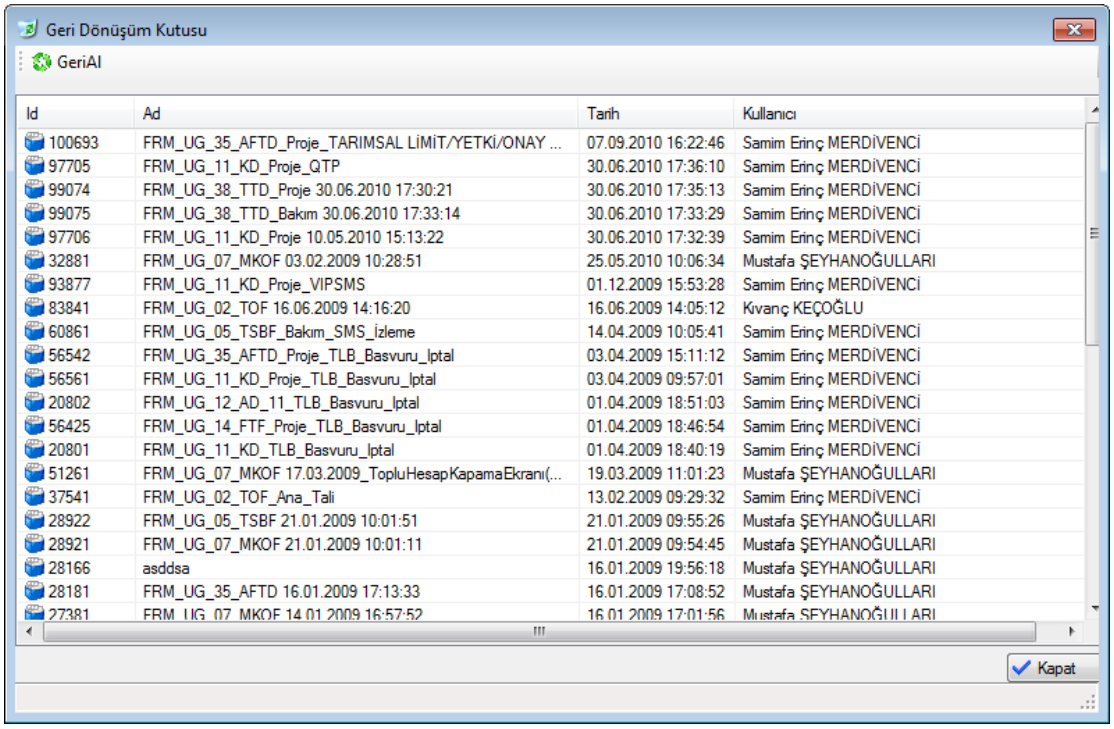
Şekil. 60. Klasör Yetkilendirme



Şekil. 61. Yetki Düzenleme

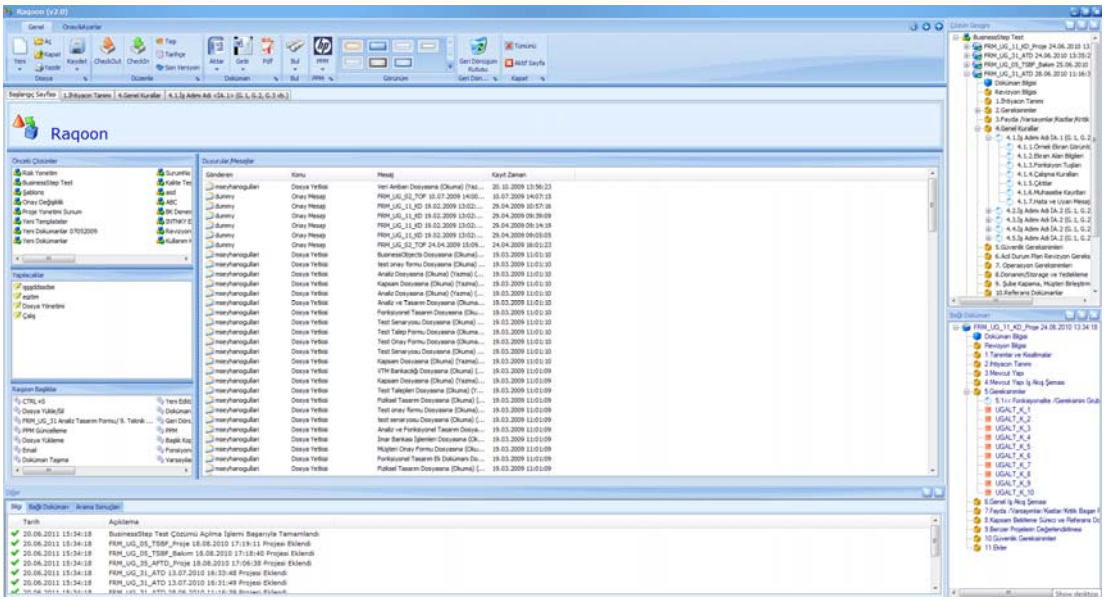
Kullanıcı yeni bir klasör yaratmak ve kullanıcılara yetki vermek amacıyla bu ekranı kullanır. Yetki düzenle denildiğinde Departman veya Kullanıcı seçeneği seçilir. Departman seçeneği seçildiğinde bir departmana yani departman altındaki tüm kullanıcılara, kullanıcı seçeneği seçildiğinde bir kullanıcı seçilir ve Okuma,Yazma ve Silme yetkilerinden biri veya hepsi tanımlanır.

- Geri Dönüşüm Kutusu: Doküman'la silindiğinde fiziksel olarak silinmeyip yanlızca statu'leri değişir. Kullanıcılar dilerse silinmiş statüsündeki dokümanlarını geri dönüşüm kutusundan aktif hale getirilebilir. Şekil 62'de geri dönüşüm kutusu örneği bulabilirsiniz.



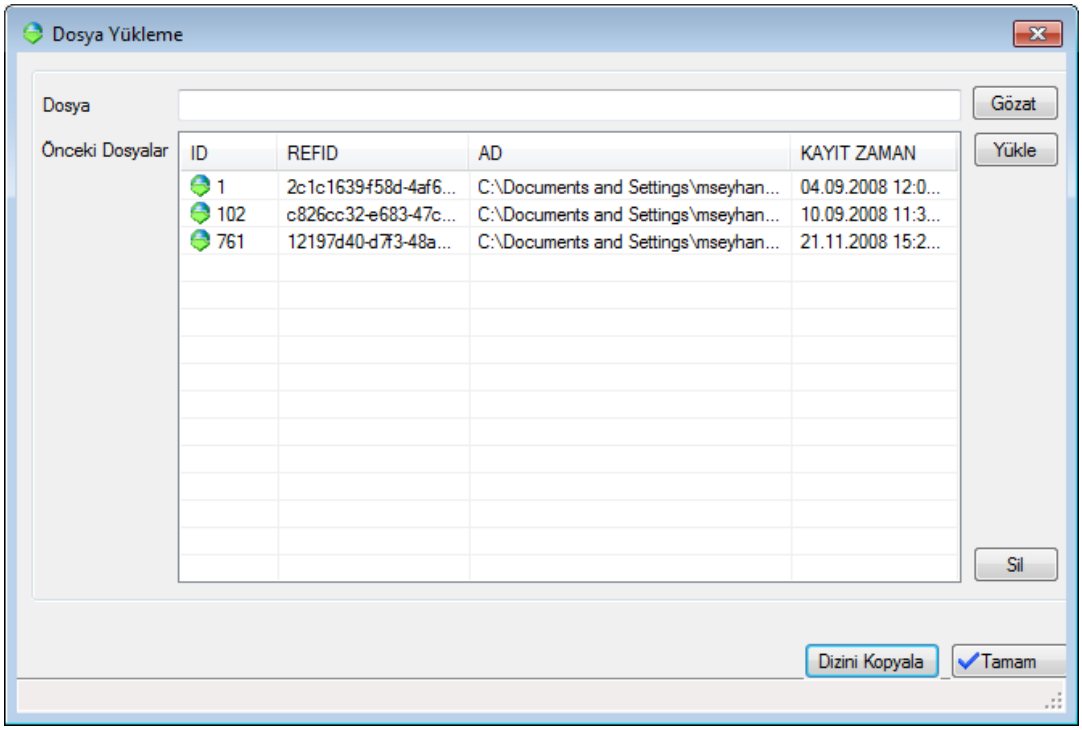
Şekil. 62. Geri Dönüşüm Kutusu

- Görünüm: Kullanıcıların guygulamanın görünüm rengini değiştirmek amacıyla kullandıkları bir özelliktir. Şekil 63'te renk değişikliğini görmek mümkündür.



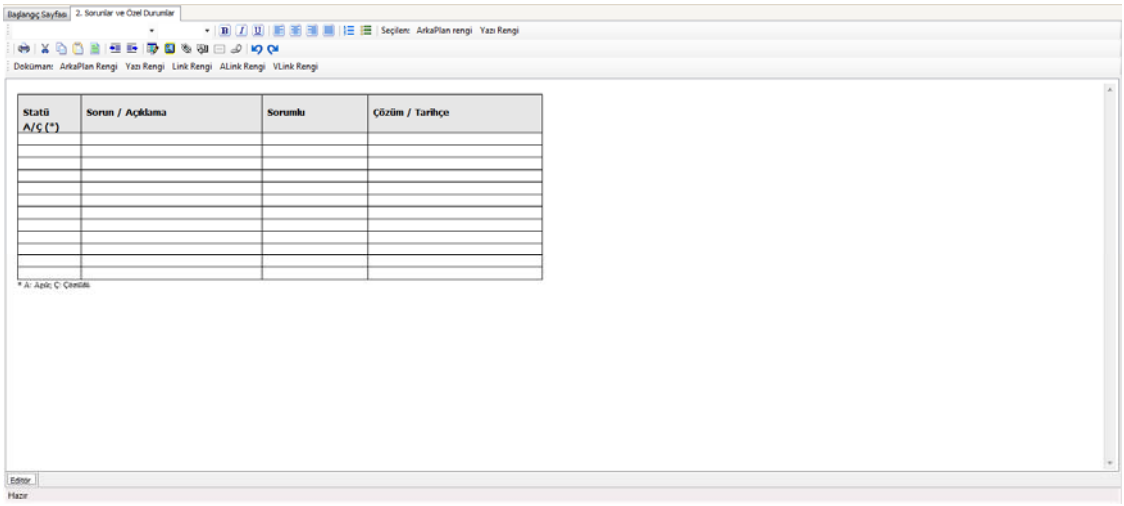
Şekil. 63. Tema Değişikliği

- Dosya Yükleme: Dosya yükleme yapılan projelerde ortaya çıkan uygulama dışı dokümanlar için kullanılır. Örneğin toplantı notları,emailler vs. Dosya yükle ekranında gözat butonuna basılarak çıkan ekrandan dosya seçilir. Daha sonra yükle butonuna basılarak dosya önceden tanımlanmış olan FTP dizininine yüklenir. Daha sonra kullanıcı dosyanın bulunduğu FTP linkini kopyalayarak dokümanındaki herhangi bir başlığın editörüne link olarak eklenir.



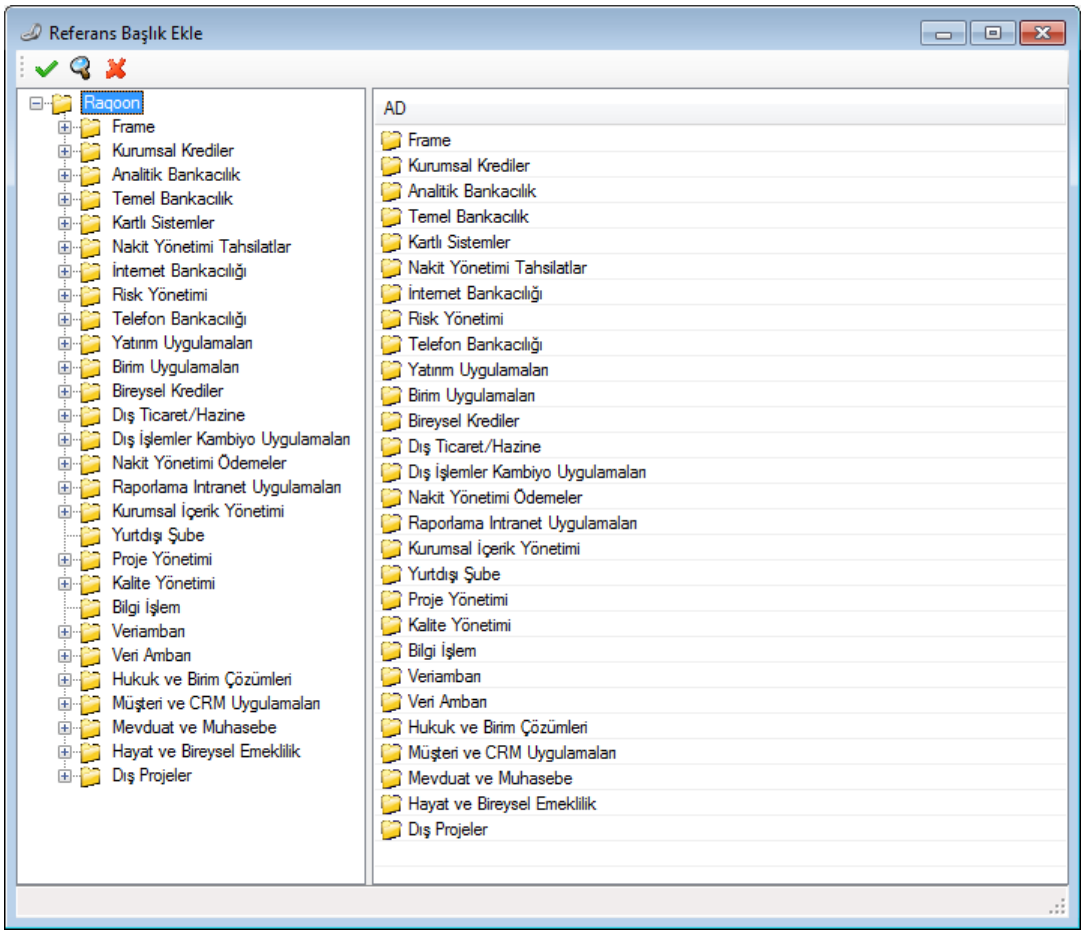
Şekil. 64. Dosya Yükleme

- Editör: Daha önceden de belirttiğim gibi uygulamamızda doküman yazılımı yapılacağı zaman her bir başlık için ayrı bir editör açılır. Açılan bu editörler tamamiyle html editörlerdir. Yani yazılan her kelime arka planda bir html text oluştururlar. Yine bu editör üzerinde yazı karakterlerini değiştirme,yazı tipini değiştirme,renk değişikliği yapma,kopyalama,hizalama gibi birçok işlemi yapılabilmektedir.



Şekil. 65. Editör

- Referans Başlık: Referans başlık özelliği uygulamanın en güçlü özelliklerinden biridir. Kullanıcı herhangibir dokümandaki başka bir içeriği kendi dokümanında referans olarak gösterebilir. Bu durum şu anlama gelmektedir. Referans olarak gösterin başka bir başlıkta değişiklik olduğu zaman bu değişiklik olduğu gibi kullanıcının dokümanınada yansır. Referans başlık özelliği şu şekilde çalışmaktadır; Kullanıcı başka bir başlığı referans olarak eklediği zaman uygulama seçilen başlığın id'sini alarak ilgili dokümana bir script ekler. (Ör:Link[{1250}]) Bu doküman açıldığı zaman uygulama referans başlığın id'sini alır ve ilgili başlığın içeriğini getirerek kullanıcının baktığı dokümana yapıştırır.



Şekil. 66. Referans Başlık Ekleme

4. Sonuç Ve Öneriler

Hazırlanan uygulama mevcut haliyle gereksinim ve doküman yönetimlerini yapabilecek kapasitede bir uygulama haline gelmiştir. Uygulama önceki bölümlerde de belirttiğim gibi Visual Studio :net ortamında c# dili kullanılarak hazırlanmıştır. Ayrıca uygulama veri tabanı olarak oracle 10g veri tabanı kullanmaktadır.

Hazırlanan uygulama için öncelikle IRaqoon interface'i oluşturulmuştur. Bu interface'in kullanılmasının temel olarak iki amacı vardır. Bunların birincisi bu interface'den türeyen tüm sınıfların belirli bir şablon çerçevesinde geliştirilmesidir. Bilindiği gibi belirli bir interface'ten türeyen sınıfların interface'te tanımlanmış olan metod ve özellikleri kullanma zorunluluğu vardır, bu sayede sınıflarınızıda belirli bir standartı sağlamış olursunuz. İkinci olarakta uygulama geliştiricilerin çok iyi bildiği gibi jenerik metodların oluşturulmasını sağlar. Örneğin bu interface türünden parametre alan bir metodunuza, bu interface'ten türeyen tüm sınıflarınızı parametre

olarak geçebilirsiniz ve bu sayede her sınıfa ayrı metod yazmak yerine hepsi için tek bir metod yazarak uygulamanızı kod kalabalığından kurtarabilirsiniz.

Interface yazılımını tamamladıktan sonra iki temel sınıfımız olan ItemBase ve ProjectBase sınıfları oluşturulmuştur. Bu iki sınıfımızda TreeNode adını verdiğimiz Framework 2.0 kütüphanelerinde tanımlı olan bir sınıftan türemektedir. Bu sınıftan türemelerinin temel sebebi her iki sınıfında kullanıcı önyüzünde treeview komponentinde gösterilmelerinden kaynaklanmaktadır. Çünkü treeview komponetine yanlızca treenode sınıfı tipinden objeler eklenebilmektedir. Oluşturduğumuz bu temel sınıflar hiçbir zaman yanlız başlarına kullanılmamktadırlar. Bu temel sınıflar yanlızca yeni oluşturulacak sınıfların bu sınıfları kullanarak türemesini sağlarlar. Hazırlanan uygulamada ItemBase temel sınıfı kullanıclacak başlık sınıflarının oluşturulmasını sağlamak amacıyla kullanılmıştır. (Ana Başlık,Alt Başlık,Sabit Başlık,Gereksinim Başlığı vs.) Uygulamaya yeni bir başlık tipi eklemek için uygulamayı geliştirecek kişinin yeni oluşturacağı başlık sınıfını ItemBase temel sınıfından türetme zorunluluğu vardır. Benzer bir şekilde ProjectBase temel sınıfıda doküman ve şablon sınıflarının oluşturulması amacıyla kullanılır. Yani oluşturulacak tüm doküman ve şablon sınıfları projectbase temel sınıfından türemek zorundadırlar.

Temel sınıfların yazılımı tamamlandıktan sonra başlık ve doküman sınıflarının yazılımına geçilebilir. Bu sınıflar önceden oluşturduğumuz temel sınıflardan ve IRaqoon interface'inden türemek zorundadırlar. Bu sınıflar uygulama içersinde kullanılacak ana sınıfları oluşturacaklardır. Öte yandan uygulama içersinde belirli fonksiyonaliteler için ana sınıflara yardımcı olacak başka sınıflarda kullanılmaktadır. (Ör: DocumentHelper,FTPHelper,HttpHelper,Utility vs...)

Veri tabanı kodlaması c# sınıf ve kodlarının yazılması ile paralel yürütülür. Veri tabanı işlemleri tamamiyle stored prosedür ve oracle fonsiyonlar ile yapılmaktadır. Bu şekilde yapılmasının temel sebebi yönetiminin çok kolay olmasından kaynaklanmaktadır. Prosdür ve fonsiyonların yazılımı için pl sql aracı kullanılmıştır.

Yapılan bu geliştirmelerin ardından uygulama temel fonksiyonlarıyla çalışırı hale getirilebilir. Bu aşamdan sonra uygulama geliştirici kurulan bu mimariyi kurarak uygulamaya yeni özellikler katabilir. Uygulama mevcut haliyle şablonların oluşturulmasını ve bu şablonlardan doküman türetimesini sağlamaktadır. Üretilen bu

dokümanlar hangi şablondan türemişse o şablonun özelliklerini aynen kendisine aktarmaktadır. Bu sayede dokümalar için belirli bir standart sağlanmış olur. Yine uygulamada doküman ve başlıklara bir çok özellik kazandırılmıştır. Dokümanın veya başlığın email olarak gönderilmesi, herhangibir onay uygulaması ile entegre olarak çalışabilmesi, versiyonlama yapılması, oluşturulan versiyonların karşılaştırılabilmesi sayabileceğimiz bir kaç özelliğinden biridir.

Uygulamayı bir adım öteye götürmek amacıyla mevcut haline bir takım yeni özellikler eklenebilir. Örneğin günümüzde çok yaygın olarak kullanılan ve hem dokümanın güvenliğini hemde kişiselleştirmesini sağlayan elektronik imza entegrasyonu yapılabilir. Uygulamanın mevcut hali bu entegrasyon için elverişlidir. Uygulamaya yapılacak en büyük değişiklik ise masaüstü uygulamadan web uygulamasına çevrilmesidir. Tabikide web uygulaması yapmak oldukça maliyetli bir iş olacaktır, fakat bu sayede uygulamanın uzak bilgisarlarda herhangi bir kurulum gerektirmeden kullanılabilecektir.

KAYNAKLAR

[1] Dr. M. Erhan Sarıdoğan, Yazılım Mühendisliği, s. 4-11, Papatya Yayıncılık Eğitim, İstanbul, 2008.

[2] Hamza Kandur, Elektronik Belge Yönetimi Sistem Kriterleri Referans Modeli, İstanbul, 2006.

[3] Doküman ve Süreç Yönetimi Nedir?, http://www.envision.com.tr/About_DMS.aspx, 2009

[4] Bilgisayar Dersanesi, IIS Nedir?, http://www.bilgisayardershanesi.com/bilgisayar_dersleri/iis-nedir.html, 2009

[5] Scott McLean, Microsoft .Net Remoting, s. 10-150, Microsoft Yayınları, 2002

[6] Borland Caliber, http://btgrubu.com

EKLER

EK-1 : Raqoon Main Kod Örneği

```
private void trvMenu_AfterSelect(object sender, TreeViewEventArgs e) {

        IRaqoon selectedItem = (IRaqoon)trvMenu.SelectedNode;
        //If Selected Item is Header,SubHeader or Requirement
        if (selectedItem.ItemType != ItemTypes.Solution &&
           selectedItem.ItemType != ItemTypes.Template)
        {
           Document relatedDocument = null;
           //Get Related Document
           if(selectedItem.ItemType == ItemTypes.Document)
              relatedDocument =
((ProjectBase)trvMenu.SelectedNode).RelatedDocument;
           else
              relatedDocument =
((ProjectBase)(((ItemBase)trvMenu.SelectedNode).Project)).RelatedDocument;
           if (relatedDocument != null)
           {
              //Add related Document to treeview
              if (trvRelatedDocument.Nodes.Count > 0)
              {
                 if (relatedDocument.Id !=
((ProjectBase)trvRelatedDocument.Nodes[0]).Id)
                 {
                    //Clear Nodes
                    trvRelatedDocument.Nodes.Clear();
                    //Add Related Document
```

```
                    addRelatedDocument(relatedDocument);

                }
            }
            else
            {
                //Add Related Document
                addRelatedDocument(relatedDocument);
            }
        }
    }
}
//Add Related Document To Related Document Treeview
private void addRelatedDocument(ProjectBase relatedDocument) {
    trvRelatedDocument.Nodes.Add(relatedDocument);
    //Set Related Document Events
    relatedDocument.ContextMenu = null;
    //Expand document
    relatedDocument.ExpandAll();
}
```

EK-2 : Raqoon UI Kod Örneği

```
/// <summary>
    /// Preview Control of Project
    /// </summary>
    public Preview PreviewControl {
        get { return m_PreviewControl; }
        set { m_PreviewControl = value; }
    }
    /// <summary>
    /// TabPage of Item
    /// </summary>
    public ExTabPage ItemTabPage {
        get { return m_ItemTabPage; }
        set { m_ItemTabPage = value; }
    }
    /// <summary>
    /// Related Document of Project
    /// </summary>
    public Document RelatedDocument {
        get { return m_RelatedDocument; }
        set { m_RelatedDocument = value; }
    }
    /// <summary>
    /// Get or Set Check Out Status of Project
    /// </summary>
    public bool IsCheckOut {
        get { return m_IsCheckOut; }
        set { m_IsCheckOut = value; }
    }
```

```
/// <summary>
/// Get or Set Checkout person of Project
/// </summary>
public int CheckOutBy {
    get { return m_CheckOutBy; }
    set { m_CheckOutBy = value; }
}
/// <summary>
/// Does Project Need versioning or not
/// </summary>
public bool Versioning {
    get { return m_Versioning; }
    set { m_Versioning = value; }
}
/// <summary>
/// Get Create Date of Document
/// </summary>
public DateTime CreateDate {
    get { return m_CreateDate; }
}
/// <summary>
/// Get Document Creator
/// </summary>
public int CreateBy {
    get { return m_CreateBy; }
    set {  m_CreateBy = value; }
}
/// <summary>
/// Get Project Information
/// </summary>
public string About {
    get { return m_About; }
}
/// <summary>
```

```
/// Get or Set Document Header
/// </summary>
public string HeaderURL {
   get { return m_HeaderURL; }
   set { m_HeaderURL = value; }
}
/// <summary>
/// Get or Set PPM Id
/// </summary>
public int PPMId {
   get { return m_PPMId; }
   set { m_PPMId = value; }
}
/// <summary>
/// Get ImportLog
/// </summary>
public List<ListViewItem> LvwImportLog {
   get { return m_LvwImportLog; }
}
/// <summary>
/// Get or Set Module of Document
/// </summary>
public Module DocumentModule {
   get { return m_DocumentModule; }
   set { m_DocumentModule = value; }
}
/// <summary>
/// Load the TreeView content
/// </summary>B
/// <param name="tree"></param>
/// <param name="filename"></param>
/// <returns>Errorcode as int</returns>
#region Constructors
public ProjectBase() {
```

```
    this.EditBy = Globals.UserInfo.Id;
    this.Status = "A";
}
public ProjectBase(int id, bool isFolderId)
    : this() {
    if (id != -1)
    {
        //Get Project from DB by Folder
        DataTable dt = new DataTable();
        fDocument document = new fDocument();
        if (isFolderId)
            dt = readProject(id, isFolderId);
        else
        {
            dt = readProject(id, isFolderId);
        }
        //Set Project Properties by using DB values
        try
        {
            initializeProject(dt.Rows[0]);
        }
        catch { }
    }
}
#endregion Constructors
/// <summary>
/// Initialize Existing project by Using DB Values
/// </summary>
/// <param name="drProject"></param>
public void initializeProject(DataRow drProject) {
    this.Id = int.Parse(drProject["ID"].ToString());
    this.Name = drProject["NAME"].ToString();
    this.Text = drProject["NAME"].ToString();
    this.Description = drProject["DESCRIPTION"].ToString();
```

```
            this.Status = drProject["STATUS"].ToString();
            this.ProjectFolder = new
Folder(int.Parse(drProject["FOLDERID"].ToString()));
            this.Version = int.Parse(drProject["VERSION"].ToString());
            this.EditDate = DateTime.Parse(drProject["MODIFIEDDATE"].ToString());
            this.m_CreateDate =
DateTime.Parse(drProject["CREATEDDATE"].ToString());
            this.m_CreateBy = int.Parse(drProject["CREATEDBY"].ToString());
            this.EditBy = int.Parse(drProject["MODIFIEDBY"].ToString());
            this.ProjectDocumentType = new
DocumentType(int.Parse(drProject["DOCUMENTTYPEID"].ToString()));
            this.ProjectTemplate = int.Parse(drProject["TEMPLATEID"].ToString());
            this.RelatedDocument =
int.Parse(drProject["RELATEDDOCUMENTID"].ToString()) == -1 ? null : new
Document(int.Parse(drProject["RELATEDDOCUMENTID"].ToString()), false);
            if (this.RelatedDocument != null)
                this.RelatedDocument.addItems(Constants.LATESTVERSION);
            this.IsCheckOut = drProject["ISCHECKOUT"].ToString() == "Y" ? true :
false;
            this.CheckOutBy = int.Parse(drProject["CHECKOUTBY"].ToString());
            if(this.Status!="A") //Has No Approvement
                this.m_DocApprovement = new
Approvement(int.Parse(drProject["APPROVEMENTID"].ToString()),
int.Parse(drProject["FLOWID"].ToString()),int.Parse(drProject["APPROVEMENTS
ENDER"].ToString()), int.Parse(drProject["ID"].ToString()),
DateTime.Parse(drProject["APPROVEMENTSENDDATE"].ToString()),
DateTime.Parse(drProject["APPROVEMENTDATE"].ToString()),Constants.ONAY
_ONEMI);
            else
                this.m_DocApprovement = new
Approvement(int.Parse(drProject["FLOWID"].ToString()), Globals.UserInfo.Id,
"Orta", this.Id);

            this.PPMId = int.Parse(drProject["PPMID"].ToString());
```

```
            this.HeaderURL = drProject["HEADERURL"].ToString();
            this.DocumentModule = new
Module(int.Parse(drProject["MODULEID"].ToString()),
drProject["MODULENAME"].ToString());
            //Prepare ToolTipText of Project
            prepareToolTipText();
            //Clear All Nodes
            this.Nodes.Clear();
        }
       //Prepare ToolTipText For The Project
        private void prepareToolTipText() {

            string toolTipText = string.Empty;
            toolTipText += Constants.URL_PREFIX + this.Id.ToString();
            toolTipText += Environment.NewLine;
            toolTipText += Environment.NewLine;
            toolTipText += "Doküman Id:" + this.Id.ToString() + Environment.NewLine;
;
            toolTipText += Environment.NewLine;
            toolTipText += "Doküman Adı: " + this.Name + Environment.NewLine;
            toolTipText += Environment.NewLine;
            toolTipText += "Versyon No: " + this.Version.ToString() +
Environment.NewLine;
            toolTipText += Environment.NewLine;
            toolTipText += "Oluşturan: "+new
fUser().Read(this.m_CreateBy).Rows[0]["NAME"].ToString()+Environment.NewLi
ne;
            toolTipText += Environment.NewLine;
            toolTipText += "Oluşturma Tarihi: " + this.CreateDate.ToString() +
Environment.NewLine;
            toolTipText += Environment.NewLine;
            toolTipText += "Son Güncelleyen: " +new
fUser().Read(this.EditBy).Rows[0]["NAME"].ToString()+Environment.NewLine;
            toolTipText += Environment.NewLine;
```

```
            toolTipText += "Son Güncelleme Tarihi: " + this.EditDate.ToString() +
Environment.NewLine;
            toolTipText += Environment.NewLine;
            toolTipText += "Onay Akış: ";
            //Add Module Name

            if (this.DocApprovement != null)
            {
               if (this.DocApprovement.FlowId == -1)
                  toolTipText += "Tanımsız" + Environment.NewLine;
               else
                  toolTipText += this.DocApprovement.FlowId.ToString() +
Environment.NewLine;
            }
            toolTipText += Environment.NewLine;
            if(DocumentModule != null)
               toolTipText += "Modül:" + this.DocumentModule.Name;
            //Write PPM Id
            toolTipText += Environment.NewLine;
            toolTipText += Environment.NewLine;
            toolTipText += "PPM Talep: ";
            if (this.PPMId == -1)
               toolTipText += "Tanımsız" + Environment.NewLine;
            else
               toolTipText += this.PPMId.ToString() + Environment.NewLine;
            //Add Check Out Information
            if (this.IsCheckOut)
            {
               toolTipText += Environment.NewLine;
               toolTipText += "CheckOut: " + new
fUser().Read(this.CheckOutBy).Rows[0]["NAME"].ToString();
            }
            //Set Approvement Tip Text
            else if (this.Status == "B")
```

```
        {
            toolTipText += Environment.NewLine;
            toolTipText += "Onay Durum: Bekliyor";
        }
        else if (this.Status == "O")
        {
            toolTipText += Environment.NewLine;
            toolTipText += "Onay Durum: Onaylandı";
        }
        else if (this.Status == "R")
        {
            toolTipText += Environment.NewLine;
            toolTipText += "Onay Durum: Rededildi";
        }

        //Set About Text of Project
        m_About = toolTipText;

    }
```

EK-3 : Raqoon Editor Kod Örneği

```
  public const int NOERROR = 0;
      public const int S_OK = 0;
      public const int S_FALSE = 1;
      public const int E_PENDING = unchecked((int)0x8000000A);
      public const int E_HANDLE = unchecked((int)0x80070006);
      public const int E_NOTIMPL = unchecked((int)0x80004001);
      public const int E_NOINTERFACE = unchecked((int)0x80004002);
      //ArgumentNullException. NullReferenceException uses
COR_E_NULLREFERENCE
      public const int E_POINTER = unchecked((int)0x80004003);
      public const int E_ABORT = unchecked((int)0x80004004);
      public const int E_FAIL = unchecked((int)0x80004005);
      public const int E_OUTOFMEMORY = unchecked((int)0x8007000E);
      public const int E_ACCESSDENIED = unchecked((int)0x80070005);
      public const int E_UNEXPECTED = unchecked((int)0x8000FFFF);
      public const int E_FLAGS = unchecked((int)0x1000);
      public const int E_INVALIDARG = unchecked((int)0x80070057);

      //Wininet
      public const int ERROR_SUCCESS = 0;
      public const int ERROR_FILE_NOT_FOUND = 2;
      public const int ERROR_ACCESS_DENIED = 5;
      public const int ERROR_INSUFFICIENT_BUFFER = 122;
      public const int ERROR_NO_MORE_ITEMS = 259;
```

```
        //Ole Errors
        public const int OLE_E_FIRST = unchecked((int)0x80040000);
        public const int OLE_E_LAST = unchecked((int)0x800400FF);
        public const int OLE_S_FIRST = unchecked((int)0x00040000);
        public const int OLE_S_LAST = unchecked((int)0x000400FF);
        //OLECMDERR_E_FIRST = 0x80040100
        public const int OLECMDERR_E_FIRST = unchecked((int)(OLE_E_LAST +
1));
        public const int OLECMDERR_E_NOTSUPPORTED =
unchecked((int)(OLECMDERR_E_FIRST));
        public const int OLECMDERR_E_DISABLED =
unchecked((int)(OLECMDERR_E_FIRST + 1));
        public const int OLECMDERR_E_NOHELP =
unchecked((int)(OLECMDERR_E_FIRST + 2));
        public const int OLECMDERR_E_CANCELED =
unchecked((int)(OLECMDERR_E_FIRST + 3));
        public const int OLECMDERR_E_UNKNOWNGROUP =
unchecked((int)(OLECMDERR_E_FIRST + 4));

        public const int OLEOBJ_E_NOVERBS = unchecked((int)0x80040180);
        public const int OLEOBJ_S_INVALIDVERB = unchecked((int)0x00040180);
        public const int OLEOBJ_S_CANNOT_DOVERB_NOW =
unchecked((int)0x00040181);
        public const int OLEOBJ_S_INVALIDHWND = unchecked((int)0x00040182);

        public const int DV_E_LINDEX = unchecked((int)0x80040068);
        public const int OLE_E_OLEVERB = unchecked((int)0x80040000);
        public const int OLE_E_ADVF = unchecked((int)0x80040001);
        public const int OLE_E_ENUM_NOMORE = unchecked((int)0x80040002);
        public const int OLE_E_ADVISENOTSUPPORTED =
unchecked((int)0x80040003);
        public const int OLE_E_NOCONNECTION = unchecked((int)0x80040004);
        public const int OLE_E_NOTRUNNING = unchecked((int)0x80040005);
        public const int OLE_E_NOCACHE = unchecked((int)0x80040006);
```

```
        public const int OLE_E_BLANK = unchecked((int)0x80040007);
        public const int OLE_E_CLASSDIFF = unchecked((int)0x80040008);
        public const int OLE_E_CANT_GETMONIKER =
unchecked((int)0x80040009);
        public const int OLE_E_CANT_BINDTOSOURCE =
unchecked((int)0x8004000A);
        public const int OLE_E_STATIC = unchecked((int)0x8004000B);
        public const int OLE_E_PROMPTSAVECANCELLED =
unchecked((int)0x8004000C);
        public const int OLE_E_INVALIDRECT = unchecked((int)0x8004000D);
        public const int OLE_E_WRONGCOMPOBJ = unchecked((int)0x8004000E);
        public const int OLE_E_INVALIDHWND = unchecked((int)0x8004000F);
        public const int OLE_E_NOT_INPLACEACTIVE =
unchecked((int)0x80040010);
        public const int OLE_E_CANTCONVERT = unchecked((int)0x80040011);
        public const int OLE_E_NOSTORAGE = unchecked((int)0x80040012);
        public const int RPC_E_RETRY = unchecked((int)0x80010109);
    }

    public sealed class Iid_Clsids
    {
        //SID_STopWindow = {49e1b500-4636-11d3-97f7-00c04f45d0b3}
        public static Guid IID_IUnknown = new Guid("00000000-0000-0000-C000-
000000000046");
        public static Guid IID_IViewObject = new Guid("0000010d-0000-0000-C000-
000000000046");
        public static Guid IID_IAuthenticate = new Guid("79eac9d0-baf9-11ce-8c82-
00aa004ba90b");
        public static Guid IID_IWindowForBindingUI = new Guid("79eac9d5-bafa-
11ce-8c82-00aa004ba90b");
        public static Guid IID_IHttpSecurity = new Guid("79eac9d7-bafa-11ce-8c82-
00aa004ba90b");
        //SID_SNewWindowManager same as IID_INewWindowManager
```

```
public static Guid IID_INewWindowManager = new Guid("D2BC4C84-3F72-4a52-A604-7BCBF3982CBB");
public static Guid IID_IOleClientSite = new Guid("00000118-0000-0000-C000-000000000046");
public static Guid IID_IDispatch = new Guid("{00020400-0000-0000-C000-000000000046}");
public static Guid IID_TopLevelBrowser = new Guid("4C96BE40-915C-11CF-99D3-00AA004AE837");
public static Guid IID_WebBrowserApp = new Guid("0002DF05-0000-0000-C000-000000000046");
public static Guid IID_IBinding = new Guid("79EAC9C0-BAF9-11CE-8C82-00AA004BA90B");
public static Guid IID_IBindStatusCallBack = new Guid("79EAC9C1-BAF9-11CE-8C82-00AA004BA90B");
public static Guid IID_IOleObject = new Guid("00000112-0000-0000-C000-000000000046");
public static Guid IID_IOleWindow = new Guid("00000114-0000-0000-C000-000000000046");
public static Guid IID_IServiceProvider = new Guid("6d5140c1-7436-11ce-8034-00aa006009fa");
public static Guid IID_IWebBrowser = new Guid("EAB22AC1-30C1-11CF-A7EB-0000C05BAE0B");
public static Guid IID_IWebBrowser2 = new Guid("D30C1661-CDAF-11d0-8A3E-00C04FC9E26E");
public static Guid CLSID_WebBrowser = new Guid("8856F961-340A-11D0-A96B-00C04FD705A2");
public static Guid CLSID_CGI_IWebBrowser = new Guid("ED016940-BD5B-11CF-BA4E-00C04FD70816");
public static Guid CLSID_CGID_DocHostCommandHandler = new Guid("F38BC242-B950-11D1-8918-00C04FC2C836");
public static Guid CLSID_ShellUIHelper = new Guid("64AB4BB7-111E-11D1-8F79-00C04FC2FBE1");
public static Guid CLSID_SecurityManager = new Guid("7b8a2d94-0ac9-11d1-896c-00c04fb6bfc4");
```

```
        public static Guid IID_IShellUIHelper = new Guid("729FE2F8-1EA8-11d1-8F85-00C04FC2FBE1");
        public static Guid Guid_MSHTML = new Guid("DE4BA900-59CA-11CF-9592-444553540000");
        public static Guid CLSID_InternetSecurityManager = new Guid("7b8a2d94-0ac9-11d1-896c-00c04fB6bfc4");
        public static Guid IID_IInternetSecurityManager = new Guid("79EAC9EE-BAF9-11CE-8C82-00AA004BA90B");
        public static Guid CLSID_InternetZoneManager = new Guid("7B8A2D95-0AC9-11D1-896C-00C04FB6BFC4");
        public static Guid CGID_ShellDocView = new Guid("000214D1-0000-0000-C000-000000000046");
        //SID_SDownloadManager same as IID
        public static Guid SID_SDownloadManager = new Guid("988934A4-064B-11D3-BB80-00104B35E7F9");
        public static Guid IID_IDownloadManager = new Guid("988934A4-064B-11D3-BB80-00104B35E7F9");
        public static Guid IID_IHttpNegotiate = new Guid("79eac9d2-baf9-11ce-8c82-00aa004ba90b");
        public static Guid IID_IStream = new Guid("0000000c-0000-0000-C000-000000000046");
        public static Guid DIID_HTMLDocumentEvents2 = new Guid("3050f613-98b5-11cf-bb82-00aa00bdce0b");
        public static Guid DIID_HTMLWindowEvents2 = new Guid("3050f625-98b5-11cf-bb82-00aa00bdce0b");
        public static Guid DIID_HTMLElementEvents2 = new Guid("3050f60f-98b5-11cf-bb82-00aa00bdce0b");

        public static Guid IID_IDataObject = new Guid("0000010e-0000-0000-C000-000000000046");

        public static Guid CLSID_InternetShortcut = new Guid("FBF23B40-E3F0-101B-8488-00AA003E56F8");
```

```
        public static Guid IID_IUniformResourceLocatorA = new Guid("FBF23B80-E3F0-101B-8488-00AA003E56F8");
        public static Guid IID_IUniformResourceLocatorW = new Guid("CABB0DA0-DA57-11CF-9974-0020AFD79762");
        public static Guid IID_IHTMLBodyElement = new Guid("3050F1D8-98B5-11CF-BB82-00AA00BDCE0B");

        public static Guid CLSID_CUrlHistory = new Guid("3C374A40-BAE4-11CF-BF7D-00AA006946EE");

        public static Guid CLSID_HTMLDocument = new Guid("25336920-03F9-11cf-8FD0-00AA00686F13");
        public static Guid IID_IPropertyNotifySink = new Guid("9BFBBC02-EFF1-101A-84ED-00AA00341D07");

        public static Guid IID_IProtectFocus = new Guid("D81F90A3-8156-44F7-AD28-5ABB87003274");

        public static Guid IID_IHTMLOMWindowServices = new Guid("3050f5fc-98b5-11cf-bb82-00aa00bdce0b");
    }

    public sealed class WinApis
    {
        public const uint MAX_PATH = 512;
        public const uint STGM_READ = 0x00000000;
        public const uint SHDVID_SSLSTATUS = 33;
        public const int GWL_WNDPROC = -4;

        public const short
            //defined inWTypes.h
            // 0 == FALSE, -1 == TRUE
            //typedef short VARIANT_BOOL;
            VAR_TRUE = -1,
```

```
        VAR_FALSE = 0;

    [DllImport("shell32.dll", CharSet = CharSet.Auto)]
    public static extern IntPtr ExtractIcon(IntPtr hInst, string lpszExeFileName, int
nIconIndex);

    [DllImport("user32.dll")]
    public static extern bool DestroyIcon(IntPtr hIcon);

    [DllImport("user32", SetLastError = true, CharSet = CharSet.Auto)]
    public static extern int CallWindowProc(
      IntPtr lpPrevWndFunc, IntPtr hWnd, int Msg, int wParam, int lParam);

    [DllImport("user32", SetLastError = true, CharSet = CharSet.Auto)]
    public static extern IntPtr SetWindowLong(
      IntPtr hWnd, int nIndex, IntPtr newProc);

    [DllImport("ole32.dll", SetLastError = true,
    ExactSpelling = true, CharSet = CharSet.Auto)]
    public static extern int RevokeObjectParam(
      [In, MarshalAs(UnmanagedType.LPWStr)] string pszKey);

    [DllImport("user32.dll", SetLastError = true)]
    public static extern IntPtr GetWindowDC(IntPtr hWnd);

    //MSDN
    //This function should no longer be used. Use the CoTaskMemFree and
CoTaskMemAlloc functions in its place.
    [DllImport("shell32.dll", SetLastError = true)]
    public static extern int SHGetMalloc(out IMalloc ppMalloc);

    [DllImport("gdi32.dll", SetLastError = true)]
    public static extern IntPtr CreateCompatibleBitmap(IntPtr hdc, int nWidth,
      int nHeight);
```

```
        [DllImport("gdi32.dll", ExactSpelling = true, SetLastError = true)]
        public static extern IntPtr CreateCompatibleDC(IntPtr hdc);

        [DllImport("gdi32.dll", ExactSpelling = true, SetLastError = true)]
        public static extern bool DeleteDC(IntPtr hdc);

        [DllImport("gdi32.dll", ExactSpelling = true, SetLastError = true)]
        public static extern IntPtr SelectObject(IntPtr hdc, IntPtr hgdiobj);

        [DllImport("gdi32.dll", ExactSpelling = true, SetLastError = true)]
        public static extern bool DeleteObject(IntPtr hObject);

        [DllImport("gdi32.dll", SetLastError = true)]
        public static extern bool SetStretchBltMode(IntPtr hdc, StretchMode
iStretchMode);

        [DllImport("gdi32.dll", SetLastError = true)]
        public static extern bool BitBlt(IntPtr hObject, int nXDest, int nYDest,
            int nWidth, int nHeight,
            IntPtr hObjSource, int nXSrc, int nYSrc,
            TernaryRasterOperations dwRop);

        [DllImport("gdi32.dll", SetLastError = true)]
        public static extern bool StretchBlt(IntPtr hdcDest, int nXDest, int nYDest,
            int nWidthDest, int nHeightDest,
            IntPtr hdcSrc, int nXSrc, int nYSrc, int nWidthSrc, int nHeightSrc,
            TernaryRasterOperations dwRop);

        [DllImport("ole32.dll", ExactSpelling = true, CharSet = CharSet.Auto)]
        public static extern int CreateBindCtx(
            [MarshalAs(UnmanagedType.U4)] uint dwReserved,
            [Out, MarshalAs(UnmanagedType.Interface)] out IBindCtx ppbc);
```

```
[DllImport("ole32.dll", ExactSpelling = true, CharSet = CharSet.Auto)]
public static extern int CreateAsyncBindCtx(
    [MarshalAs(UnmanagedType.U4)] uint dwReserved,
    [MarshalAs(UnmanagedType.Interface)] IBindStatusCallback pbsc,
    [MarshalAs(UnmanagedType.Interface)] IEnumFORMATETC penumfmtetc,
    [Out, MarshalAs(UnmanagedType.Interface)] out IBindCtx ppbc);

[DllImport("urlmon.dll", ExactSpelling = true, CharSet = CharSet.Auto)]
public static extern int CreateURLMoniker(
    [MarshalAs(UnmanagedType.Interface)] IMoniker pmkContext,
    [MarshalAs(UnmanagedType.LPWStr)] string szURL,
    [Out, MarshalAs(UnmanagedType.Interface)] out IMoniker ppmk);

public const uint URL_MK_LEGACY         = 0;
public const uint URL_MK_UNIFORM        = 1;
public const uint URL_MK_NO_CANONICALIZE = 2;
[DllImport("urlmon.dll", ExactSpelling = true, CharSet = CharSet.Auto)]
public static extern int CreateURLMonikerEx(
    [MarshalAs(UnmanagedType.Interface)] IMoniker pmkContext,
    [MarshalAs(UnmanagedType.LPWStr)] string szURL,
    [Out, MarshalAs(UnmanagedType.Interface)] out IMoniker ppmk,
    uint URL_MK_XXX); //Flags, one of

[DllImport("user32.dll", CharSet = CharSet.Auto)]
public static extern IntPtr SendMessage(HandleRef hWnd, uint Msg,
    IntPtr wParam, IntPtr lParam);

[DllImport("user32.dll", CharSet = CharSet.Auto)]
public static extern IntPtr SendMessage(HandleRef hWnd, uint Msg,
    IntPtr wParam, ref StringBuilder lParam);

[DllImport("user32.dll", CharSet = CharSet.Auto)]
public static extern void SendMessage(HandleRef hWnd, uint msg,
    IntPtr wParam, ref tagRECT lParam);
```

```
[DllImport("user32.dll", CharSet = CharSet.Auto)]
public static extern IntPtr SendMessage(HandleRef hWnd, uint msg,
    IntPtr wParam, ref tagPOINT lParam);

[DllImport("ole32.dll", CharSet = CharSet.Auto)]
public static extern int CreateStreamOnHGlobal(IntPtr hGlobal, bool
fDeleteOnRelease,
    [MarshalAs(UnmanagedType.Interface)] out IStream ppstm);

[DllImport("user32.dll")]
public static extern short GetKeyState(int nVirtKey);

[DllImport("ole32.dll", ExactSpelling = true, PreserveSig = false)]
[return: MarshalAs(UnmanagedType.Interface)]
public static extern object CoCreateInstance(
  [In, MarshalAs(UnmanagedType.LPStruct)] Guid rclsid,
  [MarshalAs(UnmanagedType.IUnknown)] object pUnkOuter,
  CLSCTX dwClsContext,
  [In, MarshalAs(UnmanagedType.LPStruct)] Guid riid);

[DllImport("user32.dll", CharSet = CharSet.Auto)]
public static extern uint MessageBox(
    IntPtr hWnd, String text, String caption, uint type);

[DllImport("user32.dll")]
public static extern bool GetClientRect(IntPtr hWnd, out tagRECT lpRect);

[DllImport("user32.dll")]
public static extern bool IsWindow(IntPtr hWnd);

[DllImport("user32.dll")]
public static extern bool IsWindowVisible(IntPtr hWnd);
```

```
        [DllImport("user32.dll")]
        public static extern bool BringWindowToTop(IntPtr hWnd);

        [DllImport("user32.dll", SetLastError = true, CharSet = CharSet.Auto)]
        public static extern IntPtr FindWindow(string lpClassName, string
lpWindowName);

        [DllImport("user32.dll")]
        public static extern IntPtr GetWindow(IntPtr hWnd, uint uCmd);

        [DllImport("user32.dll")]
        public static extern int GetWindowText(IntPtr hWnd, StringBuilder title, int
size);

        [DllImport("user32.dll")]
        public static extern uint RealGetWindowClass(IntPtr hWnd, StringBuilder
pszType, uint cchType);

        public static string GetWindowName(IntPtr Hwnd)
        {
            if (Hwnd == IntPtr.Zero)
                return string.Empty;
            // This function gets the name of a window from its handle
            StringBuilder Title = new StringBuilder((int)WinApis.MAX_PATH);
            WinApis.GetWindowText(Hwnd, Title, (int)WinApis.MAX_PATH);

            return Title.ToString().Trim();
        }
```

EK-4 : fRaqoon Kod Örneği

```
public class fMessage : MarshalByRefObject
        {
                public fMessage()
                {
                }
                /// <summary>
                /// Add User New Message
                /// </summary>
                /// <param name="dt"></param>
                public void AddUserMessage(DataTable dt) {
                        DB db = new DB(fRaqoon.TEMPEST_DB);
                        DataRow dr=dt.Rows[0];
                        // Create parameters
                        UniParameter prmMessageFrom =
db.CreateParameter("P_MESSAGEFROM",
int.Parse(dr["MESSAGEFROM"].ToString()));
                        UniParameter prmMessageTo =
db.CreateParameter("P_MESSAGETO",int.Parse(dr["MESSAGETO"].ToString()));
                        UniParameter prmSubject =
db.CreateParameter("P_SUBJECT",dr["SUBJECT"].ToString());
                        UniParameter prmMessageBody =
db.CreateParameter("P_MESSAGEBODY", dr["MESSAGEBODY"].ToString());
                        UniParameter prmRecordBy =
db.CreateParameter("P_RECORDBY", int.Parse(dr["RECORDBY"].ToString()));
                        UniParameter prmStatus = db.CreateParameter("P_STATUS",
dr["STATUS"].ToString());

                        try {
                                db.BeginTran();
```

```
                db.ExecuteNonQuery("RQN_INS_MESSAGE_SP",
prmMessageFrom,prmMessageTo,prmSubject,prmMessageBody,prmRecordBy,prm
Status);
                db.CommitTran();
            }catch (Exception ex) {
                if (db.InTransaction) {
                    db.RollBackTran();
                }
                throw new Exception("Add new Message failed.\n" +
ex.ToString());
            }
        }
        /// <summary>
        /// Add User New Message
        /// </summary>
        /// <param name="dt"></param>
        public void AddDepartmentMessage(DataTable dt) {
            DB db = new DB(fRaqoon.TEMPEST_DB);
            DataRow dr=dt.Rows[0];
            // Create parameters
            UniParameter prmMessageFrom =
db.CreateParameter("P_MESSAGEFROM",
int.Parse(dr["MESSAGEFROM"].ToString()));
            UniParameter prmMessageTo =
db.CreateParameter("P_MESSAGETO",int.Parse(dr["MESSAGETO"].ToString()));
            UniParameter prmSubject =
db.CreateParameter("P_SUBJECT",dr["SUBJECT"].ToString());
            UniParameter prmMessageBody =
db.CreateParameter("P_MESSAGEBODY", dr["MESSAGEBODY"].ToString());
            UniParameter prmRecordBy =
db.CreateParameter("P_RECORDBY", int.Parse(dr["RECORDBY"].ToString()));
            UniParameter prmStatus = db.CreateParameter("P_STATUS",
dr["STATUS"].ToString());
```

```
                try {
                        db.BeginTran();

        db.ExecuteNonQuery("RQN_INS_DEPMESSAGE_SP",
prmMessageFrom,prmMessageTo,prmSubject,prmMessageBody,prmRecordBy,prm
Status);
                        db.CommitTran();
                }catch (Exception ex) {
                        if (db.InTransaction) {
                                db.RollBackTran();
                        }
                        throw new Exception("Add new Message failed.\n" +
ex.ToString());
                }
        }
        /// <summary>
        /// Update Message Status of User
        /// </summary>
        /// <param name="id"></param>
        /// <param name="status"></param>
        public void UpdateUserMessage(int id,string status) {
                DB db = new DB(fRaqoon.TEMPEST_DB);
                // Create parameters
                UniParameter prmID = db.CreateParameter("P_ID",id);
                UniParameter prmStatus = db.CreateParameter("P_STATUS",
status);

                try {
                        db.BeginTran();
                        db.ExecuteNonQuery("RQN_UPD_MESSAGES_SP",
prmID,prmStatus);
                        db.CommitTran();
                }catch (Exception ex) {
                        if (db.InTransaction) {
```

```
                    db.RollBackTran();
                }
                throw new Exception("Update Message failed.\n" +
ex.ToString());
            }
        }
        /// <summary>
        /// Get UnRead Messages
        /// </summary>
        /// <param name="userId"></param>
        /// <param name="status"></param>
        /// <returns></returns>
        public DataSet Read(int userId,int lastID) {
            DB db = new DB(fRaqoon.TEMPEST_DB);
            try {
                UniParameter prmUserId =
db.CreateParameter("P_USERID",userId);
                UniParameter prmLastId =
db.CreateParameter("P_LASTID",lastID);

                return
db.ExecuteDataset("RQN_SEL_UNREADMESSAGES_SP",prmUserId,prmLastId);
            }catch (Exception ex) {
                throw new Exception("Read  UnRead Messages
failed.\n" + ex.ToString());
            }
        }
        /// <summary>
        /// Get All Messages
        /// </summary>
        /// <param name="userId"></param>
        /// <param name="status"></param>
        /// <returns></returns>
        public DataSet ReadAll(int userId) {
```

```
                DB db = new DB(fRaqoon.TEMPEST_DB);
                try {
                        UniParameter prmUserId =
db.CreateParameter("P_USERID",userId);
                        UniParameter prmStatus =
db.CreateParameter("P_STATUS","A");

                        return
db.ExecuteDataset("RQN_SEL_USERMESSAGES_SP",prmUserId,prmStatus);
                }catch (Exception ex) {
                        throw new Exception("Read Messages failed.\n" +
ex.ToString());
                }
        }
        /// <summary>
        /// Send Document As Email
        /// </summary>
        /// <param name="messageFrom"></param>
        /// <param name="messageTo"></param>
        /// <param name="?"></param>
        /// <param name="messageSubject"></param>
        /// <param name="messageBody"></param>
        public void SendEmail(string messageFrom, string messageTo,string
messageCc,string messageSubject, string messageBody) {
                try {
                        MailMessage mailMessage = new MailMessage();
                        mailMessage.BodyFormat = MailFormat.Html;
                        mailMessage.From  = messageFrom;
                        mailMessage.To        = messageTo;
                        mailMessage.Cc        = messageCc;
                        mailMessage.Subject = messageSubject;
                        mailMessage.Body = messageBody;
```

```
                SmtpMail.SmtpServer = "10.75.6.12";
                SmtpMail.Send(mailMessage);
            }
            catch(HttpException he) {
                throw new Exception(he.ToString());
            }
            catch(Exception ex) {
                throw ex;
            }
        }
    }
```