KIRIKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
YÜKSEK LİSANS TEZİ

ELEKTRONİK BELGE YÖNETİM SİSTEMİ ANALİZİ: YAPAY ZEKA TABANLI ÖRNEK BİR UYGULAMA

ABDULCEBAR ON

TEMMUZ 2014

Bilgisayar Mühendisliği Anabilim Dalında Abdulcebar ON tarafından hazırlanan ELEKTRONİK BELGE YÖNETİM SİSTEMİ ANALİZİ: YAPAY ZEKA TABANLI ÖRNEK BİR UYGULAMA adlı Yüksek Lisans Tezinin Anabilim Dalı standartlarına uygun olduğunu onaylarım.

Prof. Dr. Hasan ERBAY
Anabilim Dalı Başkanı

Bu tezi okuduğumu ve tezin **Yüksek Lisans Tezi** olarak bütün gereklilikleri yerine getirdiğini onaylarım.

Doç. Dr. Necaattin BARIŞÇI
Danışman

Jüri Üyeleri

Başkan : Prof. Dr. Hasan ERBAY _____
Üye (Danışman) : Doç. Dr. Necaattin BARIŞÇI _____
Üye : Yrd. Doç. Dr. Taner TOPAL _____

……

Bu tez ile Kırıkkale Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu Yüksek Lisans derecesini onaylamıştır.

Doç. Dr. Erdem Kamil YILDIRIM
Fen Bilimleri Enstitüsü Müdürü

ÖZET

Elektronik Belge Yönetim Sistemi Analizi:
Yapay Zeka Tabanlı Örnek Bir Uygulama

ON, Abdulcebar
Kırıkkale Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı, Yüksek Lisans Tezi
Danışman: Doç. Dr. Necaattin BARIŞÇI
Temmuz 2014, 75 sayfa

Bu çalışmada, Elektronik Belge Yönetim sistemi ile ilgili hizmet veren uygulamaların eksiklikleri ön plana alınmış ve yapılan iyileştirmeler ile daha özgün bir uygulama geliştirilmiştir. Yazılım esnek bir şekilde tasarlanmıştır. Buna bağlı olarak yeni ihtiyaçlar eklenebilecek şekildedir. Ayrıca kullanım kolaylığı, performans artırımı, güvenlik gibi konularda yeni yaklaşımlar ele alınmıştır. Yazılım süreçlerinden çevik metot yöntemi kullanılmıştır. Yazılımın kendisine ait özgün yapıları bulunmaktadır. Örneğin MVC yapılarından model kısmı için dinamik model yapısı, daha hızlı sonuç veren hiyerarşik SQL yapıları, çapraz kontroller, mobil kare kodla sisteme giriş yapısı, kendi içerisinde dijital imza, doküman için katmanlı şifreleme yapısı gibi yenilikler eklenmiştir. Uygulama işlevinde genel olarak evrak kayıt işlemi, evrak akışları, yönetim ekranları ve raporlamalar bulunmaktadır. Veri madenciliği ile veri analizi yapılmış ve bunun sonucunda kullanıcılar açısından zaman ve iş yükü tasarrufu sağlanmıştır. Uygulama ayrıca dış entegrasyonlara da esnek halde tasarlanmıştır.

Anahtar Kelimeler: Belge yönetimi, görsel tasarım, güvenlik, kare kod, belge performans, veri madenciliği

ABSTRACT

Analysis of Electronic Document Management System:
Artificial Intelligence Based on a Sample Application

ON, Abdulcebar
Kırıkkale University
Graduate School of Natural and Applied Sciences
Department of Computer Engineering, Master's Thesis
Supervisor: Assoc. Prof. Dr. Necaattin BARIŞÇI
July 2014, 75 pages

In this study, lack of practice related to the providing services to Electronic Document Management System has been taken to forefront and has made more unique. Software has flexibly designed. Accordingly, the new requirements are able to be added. In addition, new approaches, such as ease of use, performance improvement, security issues are handled. Agile method was used in software process. The software has its own unique structure. For instance, improvements such as the dynamic model structure for the part of the model of the MVC structure, more rapid and hierarchical sql structures, cross-checks, login to the system with mobile qr code within its own digital signature to the document layer encryption structure have been added. In the application function, overall, document recording process, document flow, management screens and reporting are available. Data mining and data analysis was carried out and as a result time and workload has been saved the in terms of user. The application is also designed to form flexible to external integrations.

Keywords: Document management, visual design, security, QR code, performance of document, data mining

ÖNSÖZ

Elektronik Belge Yönetim sistemleri günümüzde birçok kurum ya da kuruluşlarda kullanılmaya başlanmıştır. Bununla beraber yakın zamanda kullanım alanı gittikçe genişlemiştir. Geçmişte tecrübesi olmayan elektronik belge yönetim sistemleri kullanımlarına bağlı olarak sorunlar yaşamaktadır. Sorunlar genel olarak sistem güvenliği sorunu, uygulamanın kullanılabilirlik, performans, uygulamanın süreç içerisinde yeni ihtiyaçlara cevap verememesi, yenilik sunamaması gibi sorunları örnek verilebilir. Elektronik Belge Yönetim Sistemleri ile ilgili olarak gün geçtikçe daha fazla uygulama piyasaya çıkmaktadır. Bu uygulamalar kullanıldığında hepsinde neredeyse benzer problemler yaşanmaktadır. Sistemsel sorunların çözümü veya sisteme yeni bir ihtiyacın eklenmesi uygulamanın esnekliğine bağlıdır. Bir uygulama kullanıcının tüm ihtiyaçlarını yerine getirir gibi gözükebilir. Ancak uygulama kullanıldıkça sorunlar zamanla daha net ortaya çıkmakta ve kısa sürede sorunlar çözülemeyecek duruma gelmektedir. Bu çalışmada Elektronik Belge Yönetim sistemi Uygulamasının yukarıda ifade edilen sorunlara çözüm sunacak şekilde tasarlanmaya çalışılmıştır. Çalışmada ele alınan başlıklar güvenlik, kullanılabilirlik, performans, esneklik ve veri madenciliği ile veri analizidir.

İÇİNDEKİLER DİZİNİ

Sayfa

KISALTMALAR DİZİNİ

EBYS	Elektronik Belge Yönetim Sistemi
DES	Data Encryption Standard
UML	Unified Modeling Language
MVC	Model View Controller
XHTML	Extensible HyperText Markup Language
XUL	Extensible Markup Language User Interface Language
ZUML	ZK User Interface Markup Language
MD5	Message-Digest algorithm 5
JKS	Java KeyStore
JDK	Java Development Kit
POJO	Plain Old Java Objects
AOP	Aspect Oriented Programming
ORM	Object Relational Mapping
API	Application Programming Interface
DTVT	Devlet Teşkilatı Veri Tabanı

ŞEKİLLER DİZİNİ

ŞEKİL Sayfa

1. GİRİŞ

Elektronik Belge Yönetim Sistemi (EBYS) yazışmalarla ilgili bütün süreçlerin bilgisayar ortamında yönetilmesidir. Yazışmalar belli standartlar çerçevesinde gerçekleşir. Başbakanlık Devlet Arşivleri Genel Müdürlüğü ile Türk Standartları Enstitüsünün ortak çalışmaları sonucunda hazırlanan TS 13298 Elektronik Belge Yönetimi yazışma standartları oluşturulmuştur. Bu çalışmada standartlar dikkate alınarak hazırlanmıştır [1].

1.1. EBYS İle Amaçlanan Nedir?

- Yazışmaların standartlaşması,
- Yazışmaların imza sürecinin kısaltılması
- Harcanan emek, zaman ve kırtasiye maliyetlerinden tasarruf edilmesi,
- Yazışmaların sağlıklı bir şekilde arşivlenmesi
- Yazışmalar ile ilgili raporlamaların daha sağlıklı yapılması
- Yazışmaları web üzerinden erişimi sağlanarak her konumdan ulaşılması hedeflenmektedir.

1.2. EBYS İle Ne Yapılabilir?

- Evrak akışları oluşturulur.
- Evraklar her zaman alıp gönderilebilir.
- Evrak akışlarında evrak durumu takip edilebilir.
- Evrak akışları sonlandırılabilir.
- Yetkilere göre raporlamalar yapılabilir.
- Evrak birden fazla yere gönderilebilir.
- İmzacı listesi sırası oluşturulabilir.
- Evrak dijital imza ile imzalanabilir.
- Kullanıcı yönetimi ile evrak ile ilgili işlem rolleri belirlenebilir.

- Elektronik imza ile evrak geçerli hale getirilebilir.

1.3. EBYS Faydaları

- Doküman maliyeti azalır.
- Kırtasiye masrafları ve zamandan tasarruf sağlanır.
- Evrak takibi kolaylaşır.
- Evraklar ile ilgili güvenlik sağlanır.
- Raporlamalar ile karar destek sistemi sağlanır.
- Arşivler düzgün saklanır. Böylece evraka hızlıca ulaşılır.
- Evraklar Web üzerinden yönetildiği için 3.parti uygulamalara ihtiyaç duyulmamaktadır.

1.4. Çalışmanın Özgünlüğü

Bu çalışmada geliştirilen EBYS sisteminin diğer EBYS sistemlerine göre farklı yaklaşımları bulunmaktadır. Daha önce geliştirilen sistemlerin eksiklikleri dikkate alınarak geliştirilmiş ve yeni özellikler eklenmiştir.

Yazılım web ortamında hizmet sunduğu için kullanıcılar web internet adresi ile erişimi sağlamaktadırlar. İnternet ortamında erişime açık olan uygulama yazılımları istemci-sunucu mantığında çalışmaktadır. Bu çalışmada yazılımla ilgili süreçlerin büyük bir çoğunluğunun sunucu tarafında işlem görmesi sağlanmıştır. Dolayısıyla istemci tarafına oransal olarak daha az işlem yükü gelmesi sağlanmıştır. Sunucu tarafında Java istisna yönetimi, kontrol ve doğrulama algoritmaları kullanılmıştır. Örneğin; yazılımda 2 farklı kullanıcı aynı anda evrak oluştururken aynı evrak numarası almaması gerekir. Bunun kontrolü için veri tabanı tablosuna benzersiz anahtar kuralı koyulmuştur. Bu kural ile birlikte aynı evrak numarası yazılımda eklenmeye çalışıldığında sunucuda benzersiz hata kuralına düşecektir. Ve bu hata kuralı istisna yönetimi tarafından ele alınmış olacaktır. Veri tabanı tablosu tarafından kaydedilmesi engellenen bu hata kuralına ait Java istisna yönetiminde kod karşılığı

bulunmaktadır. Bu istisna türü kullanıcılar için farklı evrak numarası oluşturarak bir kontrol işlemini gerçekleştirmiş olacaktır. Burada istisna yönetiminin sağladığı fayda işlem yapılırken önüne çıkabilecek hataları düzelterek veya yapılan işlemi kurallara uygun hale getirilmesini sağlayarak süreçlerine devam etmesidir. Sonuçta uygulama sunucusunda bir işlem yapılmıştır ve yazılım için zaman ve performans tasarrufu sağlamıştır. Eğer ki istisna yönetimi ile bu işlem gerçekleşmeseydi evrak numarası için 2 işlem yapılacaktı. Önce yeni oluşturulan evrak numarası sistemde var mı yok mu diye bir kontrol işlemi, sonra normal evrak kayıt işlemi olacaktı.

Evrak işlemleri yapılırken sunucu için performans çok önemlidir. Dolayısıyla yazılımda iş yükü tasarrufu sağlanmalıdır. Evraklar birden fazla yere gönderildiğinde programlama dili döngüsü ile veri tabanına çoklu kayıt eklenir. Bu çalışmada doküman kaydı yapılırken evrak için gideceği yer çoklu seçildiğinde ve sonrasında kayıt işlemi gerçekleştirildiğinde sunucuda işlemler veri tabanı dili döngüleri ile yapılmış olacaktır. EBYS sistemlerinde çoklu kayıt işlemi yapıldığında yazılım programlama dili döngüsü ile işlemler gerçekleştirmektedir. Yükseköğretim Kurulu EBYS, İçişleri Bakanlığı EBYS, Dışişleri Bakanlığı EBYS gibi sistemler örnek verilebilir. Bu çalışmada benzer parametrelerin kullanıldığı çoklu kayıt işlemlerinde programlama döngüleri yerine veri tabanı dili döngüleri kullanılmıştır. Bunun avantajı; veri tabanında hem kayıtlar bulunmaktadır, hem de veri tabanı erişimi için ara katman kullanılmadan direkt veriler üzerinde kayıt döngüleri kullanılmaktadır. Sonuçta kayıt işlemleri için performans artırılmıştır.

Kurumsal yazılımlarda kayıt işlemlerine ait veriler veri tabanına aktarılırken uygulama modelleriyle yani yazılım nesneleri ile yapılmaktadır. Örneğin; evrak işlemi için evrak modeli tanımlanması gerekir. Modellerin yazılım geliştiricileri tarafından elle kodlanmasının dezavantajları bulunmaktadır. Örneğin; Evrak için geliştirme sürecinde yeni bir özellik eklenmesi durumunda, geliştiricilerin kod kısmında bu özelliği eklemesi gerekmektedir. Ayrıca zamanla model için özellikler artabilir ve kod kısmında modellerde sürekli değişiklik yapılması gerekebilir. Bu çalışmada kullanılan model yapısı daha farklı bir şekilde ele alınmıştır. Dinamik model olarak kullandığımız Java kütüphanesine ait HashMap nesnesiyle yeni eklenen özellik dinamik olarak kendi içerisinde yeni özelliği oluşturacaktır. Böylece sürekli

olarak kod kısmında değişiklik yapılmasına gerek kalmayacaktır. Veri tabanında tabloya yeni bir kolon eklendiğinde ve bu tablodan HashMap nesnesi ile veri alındığında kod kısmında değişiklik yapılmadan veriler güncel olarak gelecektir. Geliştirme bölümünde HashMap nesnesi detaylı olarak ele alınmıştır. Geliştirilen diğer EBYS sistemlerin çoğunda model yapısı bulunmaktadır. Açık kaynak yazılım hizmeti veren EBYS sistemlerinde genel olarak geliştiriciler tarafından tanımlanan model yapıları bulunmaktadır. Bu alanda yapılmış ticari ve akademik çalışma örnekleri arasında Knowledge Tree, Alfresco, Documentum, Main-Pyrus DMS, OpenKM, LogicalDoc gibi sistemler bulunmaktadır.

Kurumsal uygulamalarda genellikle giriş ekranlarında kullanıcı adı ve şifre bilgisine ihtiyaç duyulur. Günümüzde yaşanan en büyük sorunlardan bir tanesi de şifrelerin unutulması durumudur. Bu yüzden kullanıcılar, giriş gerektiren uygulama ekranlarında 'Beni Hatırla' seçeneğini çok sık kullanmaktadırlar. Çünkü her seferinde kullanıcı adı ve şifre bilgileri girişini yük olarak görmektedirler. Çalışmada bu sorunun çözümü için mobil kare kod uygulaması tasarlanmıştır. Kullanıcı, yazılım için hazırlanmış uygulamayı mobil telefona yükler. Sonra mobil uygulamaya ait kullanıcı bilgilerini bir sefere mahsus olarak girerek kaydeder. Web uygulama ekranında Mobil Kare kod giriş seçeneği ile ekrandaki kare kodu Mobil uygulama ile tarayıp giriş yapmasını sağlayacaktır. Sonuçta kullanıcı bilgilerinin web uygulama ekranında devamlı girilmesi işlemi ortadan kalkmış olacaktır. Kare kod doğrulama sistemi kullanıcı giriş ekranlarında henüz kullanılmamıştır. Alış veriş siteleri için İş Bankası tarafından geliştirilen para kod uygulaması kare kod doğrulama sistemini kullanmaktadır. Ancak bu çalışmada farklı olarak doğrulama sistemi kullanıcı bilgileri için kullanılmıştır.

Bu çalışmada veri analizi için veri madenciliği kullanılmıştır. Veri madenciliği için Weka'nın Multinomial Naive Bayes (MNB) algoritması kullanılmıştır. Kullanılan bu yöntemle girilen bilgilerin metin sınıflandırılması yapılarak evrakı oluşturan kullanıcıya öneriler sunulmuştur. Bu öneriler ile kullanıcı daha önce oluşturduğu benzer evrak bilgileri ile ekranı doldurarak bilgi girişini hızlandırmıştır. Veri madenciliği için Weka metin sınıflandırma yöntemi birçok bilimsel çalışmada ele alınmıştır. Elektronik ortamda saldırı tespit yöntemi, web uygulamalarının log

yönetimi, Yapay sinir ağı tabanlı güneş radyasyonu tahmini için en uygun girdi parametrelerinin seçimi gibi alanlarda bu yöntem kullanılmıştır [2] [3] [4]. Ayrıca yazılımda kayıt işlemlerinde öneriler için Levanshtein Distance yöntemi kullanılmıştır. Bu yöntemle kayıt aramalarında aranan kelimeye benzer diğer kelimelerin benzerlik karşılaştırılması yapılarak görüntülenebilmesi sağlanmıştır [5].

EBYS sisteminin elektronik ortamda kullanılmasının faydaları bulunmaktadır. EBYS sistemi elektronik ortamda hizmet sağladığından dolayı sanal ortamda her mekândan erişimi sağlanabilmektedir. Bu özelliğinden dolayı bulut teknolojisi yapısına uygun hale gelebilmektedir. Yapılan bir çalışmada doküman yönetiminde her kullanıcı için belli bir sunucu bellek alanı tahsis edilerek, bulut teknolojisi imkânlarından faydalanılması sağlanmıştır [6]. Bunun yanında EBYS sistemi, sanal ortamda evrak takibini kolaylaştırdığından dolayı İngiltere'de yerel yönetim mahkemeleri tarafından dava evrakları için tercih edilmiştir [7].

Tezin ikinci bölümünde, EBYS sistemi için yazılım geliştirme yöntemi ele alınmış ve yazılım için kullanılan teknolojilerden bahsedilmiştir.

Üçüncü bölümde, EBYS sisteminin yeniliklerinden ve diğer EBYS sistemlerine göre iyileştirilmesi yapılmış yazılım yöntemlerinden bahsedilmiştir.

Dördüncü bölümde, EBYS yazılımı için yenilik, güvenlik, kullanılabilirlik ve veri madenciliği alanlarının sonuçları ele alınmıştır.

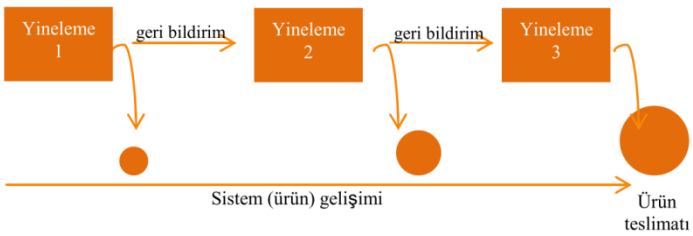
2. MATERYAL VE YÖNTEM

EBYS sistemi kurum içi veya kurumlar arası evrak akışını elektronik ortamda sağlamak için oluşturulmuş bir otomasyon sistemidir. Otomasyon sisteminde zaman içerisinde evraklar ile ilgili ihtiyaçların artması ve mevcut modüllerde değişikliklerin yapılması kaçınılmaz hale gelmiştir. Bu durumlar dikkate alınarak yazılım esnek bir şekilde tasarlanmıştır. Yazılımın nasıl olması gerektiği, hangi amaçlar için kullanılacağı, hangi kuralların olması gerektiği, EBYS sistemini daha önce kullanmış olan kullanıcıların tecrübeleri vb bilgiler uygulama yapısının oluşumuna katkı sağlamıştır. Sonuçta EBYS sisteminin mevcut işlevini yerine getirebilmesi sağlanmıştır ve yeni ihtiyaçları karşılayabilecek şekilde tasarlanmıştır.

EBYS sistemleri için standartlar mevcuttur. Standartlar, evrakların nasıl bir yöntem üzerinde sürdürüleceği hakkında fikir vermektedir. Ayrıca standartların uygunluğu açısından belgeler Pdf ve Word dosya biçimlerinde olmalıdır.

2.1. EBYS Sistemi Geliştirme Süreci

Uygulama geliştirme sürecinde kullanıcılar ile sürekli iletişim halinde olunmuştur. Bu yüzden en uygun yazılım geliştirme süreçlerinden biri olan Çevik metot yöntemi kullanılmıştır (Şekil 2.1) [8].



Şekil 2.1. Çevik Metot Modeli

Çevik metot yönteminde uygulama prototiplerinin oluşturulması ile süreçler yönetilir. Oluşturulan Prototipler belli aralıklarla düzenli olarak kullanıcılara sunulur ve yazılımda taleplerin uygunluğu ölçülür. Bu yöntem ile birlikte elde edilen geri beslemeler ile eksiklikler giderilmektedir.

Çevik metot yöntemi, tekrarlanan yazılım geliştirme metodundaki her bir yineleme kendi içinde bir yazılım projesinin gereksinim analizi, tasarım, kodlama ve test gibi adımları içerir. Ancak, her bir yinelemenin içerisindeki bu proje adımlarının ağırlığı değişkendir. Öyle ki, ilk yinelemelerde gereksinim analizi kodlamaya göre daha yoğunken ileriki yinelemelerde durum değişerek gereksinim analizinin içeriği küçülmekte ve kodlamanın ağırlığı artmaktadır. Çevik metot yönteminde ekran çıktıları çizilir, böylece talepler daha da somutlaşmaktadır. Çevik metot yöntemi kullanılarak, diğer yazılım süreçleri gibi benzer aşamalar bu çalışmada da ele alınmıştır.

Yazılım geliştirme süreçleri standart olarak 5 adımdan oluşmaktadır;

- Analiz
- Tasarım
- Geliştirme
- Test
- Kurulum

2.1.1. Analiz ve Tasarım

EBYS sisteminin amacının belirlenmesi için gerekli dokümanlar incelenmiş ve son kullanıcıya nasıl hizmet edeceği konusunda araştırmalar yapılmıştır. Bu incelemeler ve araştırmalar sonucunda sistemin nasıl tasarlanacağı, hangi geliştirme araçlarının kullanılacağı ve nasıl bir süreçten geçileceği konusunda bir çalışma yapılmıştır. Bu çalışmalar analiz ve tasarım aşamasında ele alınmıştır. Yazılım geliştirme süreçlerinin ilk aşaması olan analiz ve tasarım diğer yazılımlarda olduğu gibi EBYS sistemi için de temel oluşturmaktadır. Çünkü yazılımın ilk iskelet yapısının oluşumu bu aşamada gerçekleşmektedir.

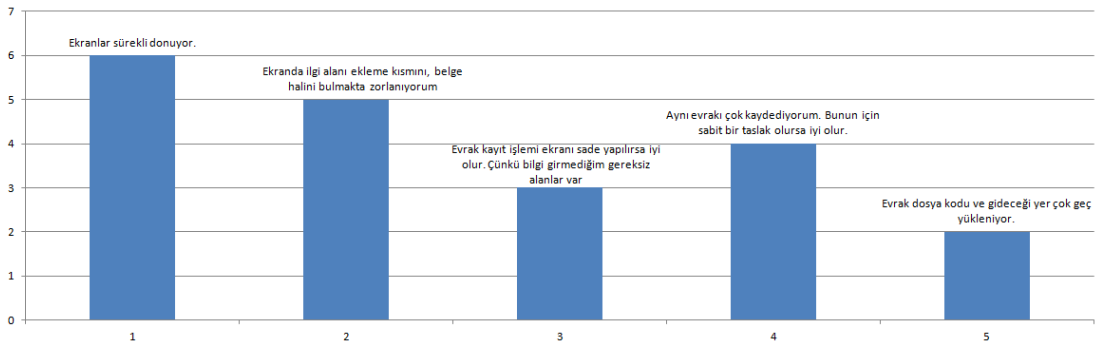
Toplanan veriler sonucunda EBYS Sistemlerinde bir evrakla ilgili gerekli olan bilgiler genel olarak evrak kayıt bilgisi, akış durumu, işlem durumu, imza durumu arşiv bilgisi ve dosya bilgisi raporlarıdır [9].

Evrak kayıt bilgisi alınırken gerekli olan bilgiler evrak kayıt numarası, evrak ekleme tarihi, gizlilik modu, ivedilik modu, kurumdaki durumu, konusu, evrak için standart dosya kodu, açıklaması, gideceği yer veya yerler, imzacı listesi, ilgisi, ekleri, dağıtım listesi, metin kısmı, gereği kısmı ve bilgi kısmı bilgileridir [10].

Uygulama yazılımlarının genel sorunlardan bir tanesi formların kullanıcı dostu şeklinde tasarlanmamasıdır. Ekranların sadeliği kullanıcı adaptasyonunu daha hızlı sağlayacaktır. Bu konuda yapılan bir araştırma örneğini inceleyelim [11].

Yükseköğretim Kurulu Elektronik Belge Yönetim Sistemini kullanan kamu personelleri ile anket yapılmıştır. 20 kişiyle yapılan anket sonucuna göre sıralama olarak yaşanan sorunların listesi şu şekildedir;

1. Ekranlar sürekli donuyor.
2. Ekranda ilgi alanı ekleme kısmını, belge halini bulmakta zorlanıyorum
3. Evrak kayıt işlemi ekranı sade yapılırsa iyi olur. Çünkü bilgi girmediğim gereksiz alanlar var
4. Aynı evrakı çok kaydediyorum. Bunun için sabit bir taslak olursa iyi olur.
5. Evrak dosya kodu ve gideceği yer çok geç yükleniyor.



Şekil 2.2. Ankette kullanıcıların verdiği cevapların dağılımı

Anket sonucuna göre bu sorunlar ortaya çıkmıştır. Kullanıcıları ankette verdiği cevapları dağılım Şekil 2.2'de görüntülenmektedir. Bu sorunlar genel olarak başka bir deyimle; ekranların karmaşıklığı, ekranda ulaşılması istenen bölümler açık bir şekilde gösterilmemesi, benzer evrak bilgilerinin sürekli olarak girilmesi gibi sorunlardır. Ekranların sade olması karmaşıklığı azaltan en temel çözümdür. Ancak ekranlar sadeleşirken tek ekranda tüm bilgilerin de ayrıca sunulması gerekmektedir. 2005 yılında Dr. Cynthia Xin Zhang tarafından hazırlanan bir çalışmada bu mümkün hale gelmiştir. Ekranlarda sade renklerin tercih edilmesi, yönlendirici ikon resimlerin kullanılması, detaylı giriş ekranlarının ayrı bir küçük ekranda sunulması gibi öneriler sunulmuştur [12]. Yine aynı çalışmada ikon resim kullanımı giriş alanlarının ne için kullanıldığını ifade etmek için tercih edilen bir yöntem olmuştur.

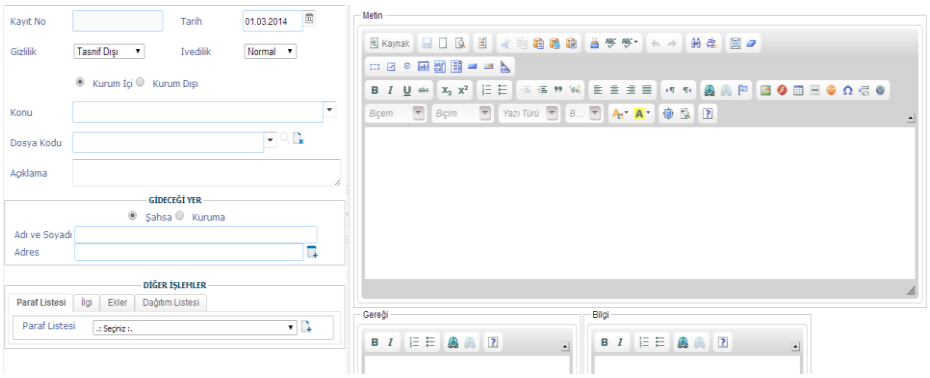
Hedef kullanıcı kitlesi sadece belli bir kuruma ait iç kullanıcılardan oluşmamaktadır. EBYS sisteminin farklı kategorilerde kullanıcıları bulunmaktadır.

EBYS ile ilgili Faaliyetlerinde Hedef kullanıcı kitlesi genel olarak;

1. Kurum içi personel
2. Kurum dışı personel
3. Diğer Kamu Kurum ve Kuruluşları
4. Resmi işler için başvuru yapan ilgililer ve özel kişilerden oluşmaktadır.

Belirlenen hedef kitle kullanıcılarıyla ekran sadeliği konusunda çalışmalar yapılmıştır. Kurum içi personel kullanıcıları Yükseköğretim Kurulu Personelleri, Kurum dışı personel kullanıcıları Üniversite personelleri, Diğer Kamu Kurum ve Kuruluşları kullanıcıları Dışişleri Bakanlığı personelleri, son olarak Resmi işler için başvuru yapan ilgililer ve özel kişiler Bilgi Edinme formu kullanan kullanıcılar kitle olarak belirlenmiştir. Bu kullanıcılar ile yapılan çalışma sonucunda kullanıcıların genel olarak talepleri; ekranda formların kendi içeriğiyle ilgili tüm işlemlerin aynı ekranda görmelerini sağlamaktır. Çünkü uygulamada formlar arasındaki ekran geçişleri karmaşıklığı artıran nedenler olarak görülmüştür [13].

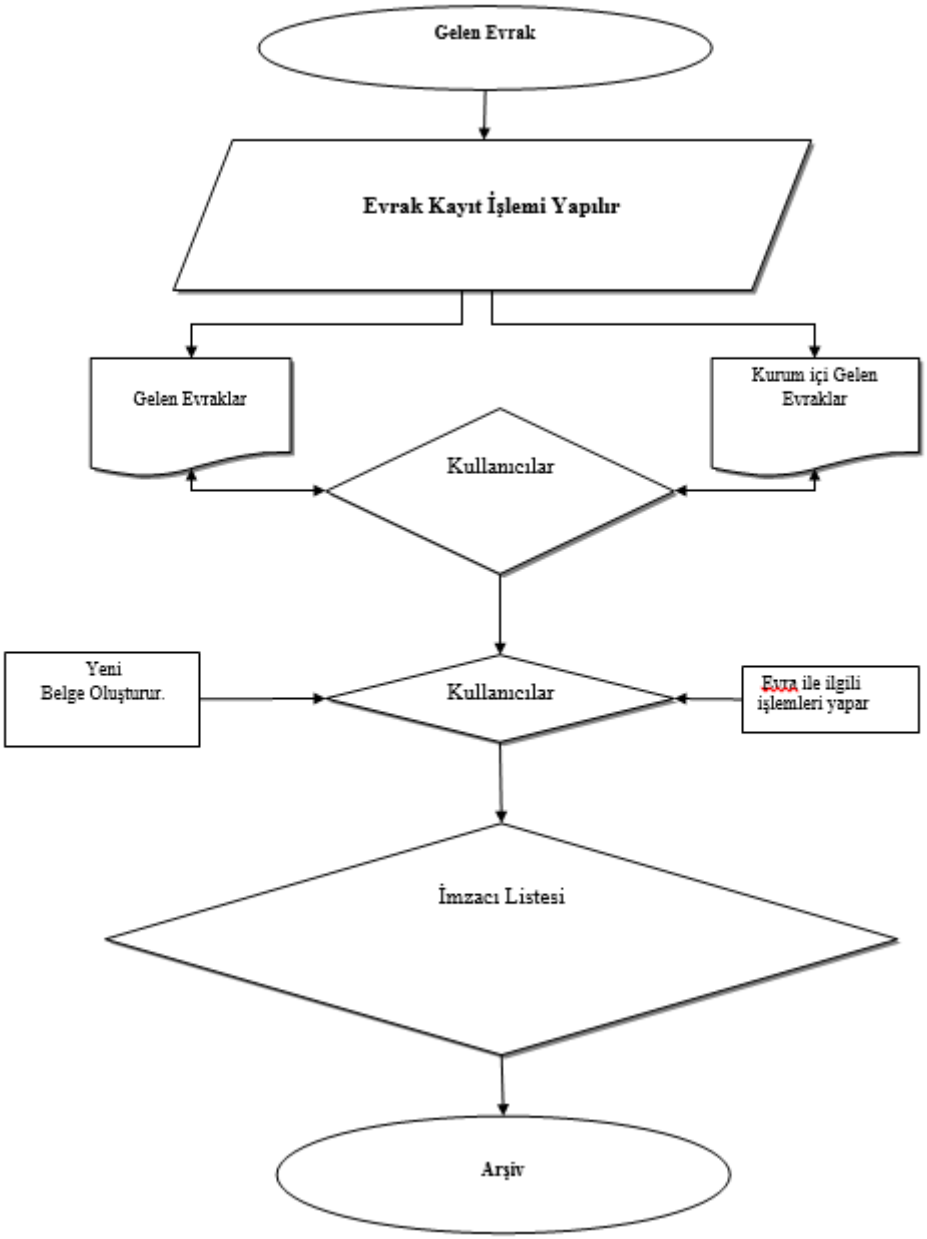
Geliştirme sürecinde bu aşamaya kadar elde edilen bilgiler sonucunda toplanan veriler için uygulama ekranı hazırlanmıştır (Şekil 2.3).



Şekil 2.3. Evrak kayıt örneği tasarımı

Uygulama form ekranı için kullanılan ara yüz tasarım teknolojisi Zk Frameworktur. Ekranda Metin, Gereği ve Bilgi alanı için Ck Editör teknolojisi kullanılmıştır. Teknolojiler ile ilgili detaylı bilgiler Geliştirme kısmında anlatılacaktır. Evrakların bilgi giriş alanları detaylı olarak ele alınacaktır.

Yazılım geliştirme sürecinde Çevik Metot yöntemi kullanıldığından dolayı evrak ile ilgili ilk ekran olan evrak giriş bilgileri kullanıcılar ile tasarlanmıştır. Tasarlanan bu ekran, evrak kayıt aşamasıdır. Evraklar kayıt edildikten sonra standartlar dahilinde evrak akışına girerler. Evrak akışları gelen evrak ve giden evrak için farklı olarak gerçekleştirilir (Şekil 2.4).



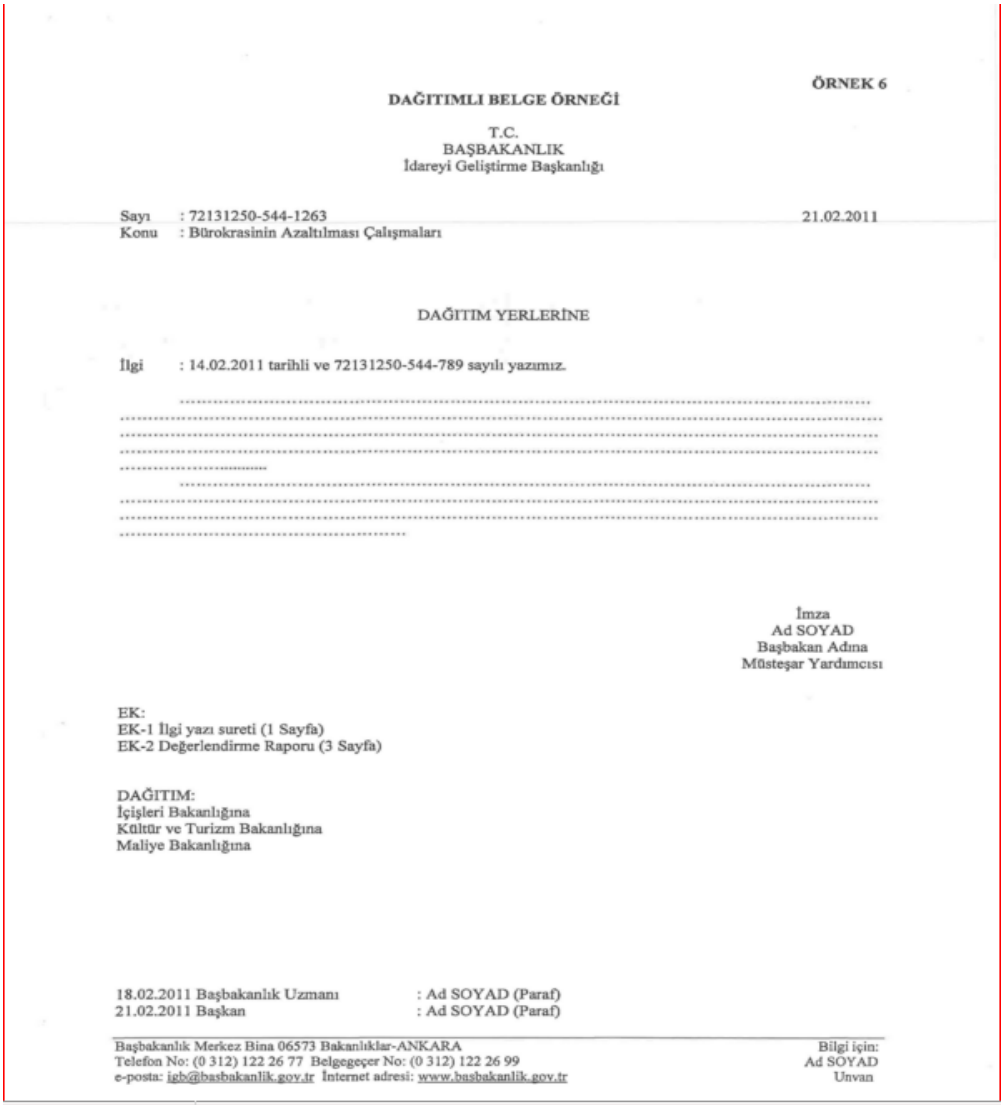
Şekil 2.4. Gelen evrak iş akışı

Dışarıdan gelen evrak bilgisi ilk olarak taranarak sisteme yansıtılır. Şekil 2.3'daki gibi ekran bilgileri alınarak sisteme kaydedilir. Kaydedilen evrak imza listesine sunulur.

İmza türleri;

İmza, Onay, Olur, Görüş, Paraf ve Koordinasyon'dur.

İmza türleri oluşturulan evrak belgesi üzerinde farklı konumlanır. İmza türleri için evrak belgesi örneği Şekil 2.5'de görüntülenmektedir [14].



ÖRNEK 6

DAĞITIMLI BELGE ÖRNEĞİ

T.C.
BAŞBAKANLIK
İdareyi Geliştirme Başkanlığı

Sayı : 72131250-544-1263 21.02.2011
Konu : Bürokrasinin Azaltılması Çalışmaları

DAĞITIM YERLERİNE

İlgi : 14.02.2011 tarihli ve 72131250-544-789 sayılı yazımız.

.....
.....
.....
.....
.....
.....
.....
.....
.....

İmza
Ad SOYAD
Başbakan Adına
Müsteşar Yardımcısı

EK:
EK-1 İlgi yazı sureti (1 Sayfa)
EK-2 Değerlendirme Raporu (3 Sayfa)

DAĞITIM:
İçişleri Bakanlığına
Kültür ve Turizm Bakanlığına
Maliye Bakanlığına

18.02.2011 Başbakanlık Uzmanı : Ad SOYAD (Paraf)
21.02.2011 Başkan : Ad SOYAD (Paraf)

Başbakanlık Merkez Bina 06573 Bakanlıklar-ANKARA
Telefon No: (0 312) 122 26 77 Belgegeçer No: (0 312) 122 26 99
e-posta: igb@basbakanlik.gov.tr İnternet adresi: www.basbakanlik.gov.tr

Bilgi için:
Ad SOYAD
Unvan

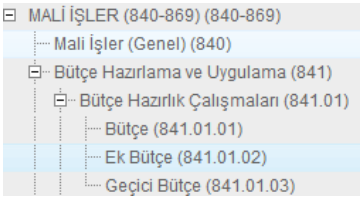
Şekil 2.5. Evrak belge örneği

Şekil 2.5'de görüldüğü gibi imza kısmı sağ tarafta metin kısmının altında bulunmaktadır. Paraflar evrakta alt kısımda bulunmaktadır. Şekil 2.3' da tasarlanan ekranda Paraf listesi sekmesi imza türlerinin eklenmesi için oluşturulmuştur.

Paraf listesi sekmesinde İmzacı listesi oluşturulurken her imza türü için Kullanıcı adı, Kullanıcı Rolü ve imza türü olması gerekmektedir. İmzacı listesi oluşturulurken Şekil 2.5'de imza türünün evrakta hangi kısımda bulunacağının dikkate alınması gerekmektedir [14].

Sonuçta evrak bilgisi alınırken imzacı listesinin standartlara uygun bir şekilde tasarlanması gerekmektedir. Çünkü evrak akışı imzacı listesine göre şekillenir. Akışlar EBYS sisteminde evrak için önemli aşamalardandır. Bu yüzden imzacı listesinin karışık olmayacak şekilde tasarlanması kullanıcı adaptasyonunu hızlandıracaktır. İmzacı listesi oluşturulurken, imzalar için eklenecek kullanıcıların ekran üzerindeki arama kısmının kolay bir şekilde tasarlanması gerekmektedir. Ayrıca kullanıcıya ait rol listesi ekranda listelenecek şekilde tasarlanmalıdır.

Evrak bilgilerinin bir alanı olan dosya kodu standart bir yapıya sahiptir. Standartlar Başbakanlık Devlet Arşivleri Genel Müdürlüğü tarafından belirlenmiştir. Dosya kodu bilgilerinin kullanıcı tarafından elle girişi engellenmelidir. Bilgilerin seçimlik alandan oluşması gerekmektedir. Ayrıca dosya kodu hiyerarşik bir yapıya sahip olduğundan dolayı ekranda ağaç yapısı modeli kullanılması kullanıcılar açısından da faydalı olacaktır. Hiyerarşik model yapılarında alt kısımlardaki bilgilerin bulunması zor olmaktadır. Çünkü hangi bilginin hangi başlık altında olduğu kullanıcı tarafından bilinmeyebilir. Bu konuda kullanıcılarla yapılan ortak çalışmada çözüm olarak sunulan fikir; ekran tasarlanırken arama alanı kısmının eklenmesidir. Aranacak kritere göre ekrandaki bilginin hiyerarşik olarak gösterilmesi gerekmektedir (Şekil 2.6) [15].



Şekil 2.6. Hiyerarşik ağaç modeli tasarımı

Ekrandaki veri giriş alanları için giriş türü şekli iyi tasarlanmalıdır. Evraka ait diğer ilgilerden konu alanının hem elle girişi, hem de seçimli alan şeklinde olması gerekmektedir. Çünkü kullanıcılar daha önce girdiği konu başlıklarının sistemde kayıtlı kalmasını talep ederler. Kullanıcıların kaydettiği evrak konu başlıkları en çok kullanılandan en az kullanılanlara göre listelenmektedir. Evrak kayıt numarası evrak sisteme ilk kaydedildiğinde uygulama tarafından otomatik olarak kayıt numarası alır. Kayıt numarası kullanıcı tarafından verilmemektedir. Şekil 2.3'da gösterildiği gibi elle bilgi girilemeyecek şekilde sadece okunabilir alan olarak tasarlanması gerekmektedir [10].

Evraka ilgi eklenirken kaydedilmiş daha önceki evraklar da ilgi olarak eklenir. Yükseköğretim Kurulu Evraklar biriminde gelen ve giden evraklar incelenirken evrak ilgileri somut olarak bulunup incelenir. Çünkü evrak ilgilerinin karışmaması gerekmektedir. Evrak kayıt sürecinde bu durumları göz önünde bulunduracak şekilde ekranların tasarlanması gerekmektedir. Sonuçta uygulamada ilgi ekleme işlemi yapılabilmesi için ekranlarda kayıtlı evraklar için arama motoru eklenmesine gerek vardır. Bu arama motoru aranan evrakın eklenebilmesini sağlamalıdır. Veya evrak kayıt bilgileri hatırlanıyorsa elle giriş de seçenek olarak sunulabilir. Eklenmiş ilgilerin kullanıcı tarafından belge olarak görüntülenmesi sağlanmalıdır. Çünkü kullanıcılar, evrakın başlığını doğru hatırlamamış olabilir. Bu yüzden evrakı belge olarak görüntüleyip içeriğinden emin olabilecektir. Kullanıcıların ekran üzerinde ilgi alanını hızlıca bulmalarını sağlamak için Şekil 2.3'da ilgi sekmesi tasarlanmıştır. Evrakın ilgi alanı giriş türüne benzer olarak evrak ekleri giriş türü alanı bulunmaktadır. Evrak ekleri sisteme PDF olarak yüklenip, mevcut kaydedilen evrak için belge olarak görüntülenmesi sağlanmalıdır. İlgi ekleme ve evrak eki ekleme yöntemlerinin benzerliği bulunmaktadır. Çünkü oluşturulan evrak başka evraka ait bilgiler eklenerek kaydedilir. Sonuçta içerik olarak benzer işleve sahip olan alanların bir arada bulunması kullanıcı için kolaylık sağlamaktadır [16].

Evrak gideceği yer bilgisi alınırken 2 farklı şekilde ele alınır; Şahıs ve kurum.
Şahsa gönderildiğinde adı, soyadı ve adres bilgisi alınır. Bu bilgiler elle giriş şeklinde olacağı için ekranda metin giriş alanı şeklinde tasarlanmalıdır. Kuruma gönderildiğinde Devlet Teşkilatı Veri Tabanı (DTVT) kodu ile veri tabanından

alınması gerekir. DTVT kodu kurumlar için standart bir kod sağlar. Sistemlerde entegrasyonu sağlayan bir yöntemdir [17].

2.1.2. Geliştirme

Geliştirme, Yazılım Geliştirme Süreçlerinde 2.aşamada bulunmaktadır. Analiz ve tasarım kısmında elde edilen bilgiler ile bu aşama gerçekleştirilir.
Bu uygulamada yazılım geliştirme için kullanılan teknolojiler;

- Zk Framework
- Spring
- Ibatis
- Itext Report
- Oracle Database
- Apache Tomcat Server
- Java Programlama Dili

2.1.2.1. Geliştirme Teknolojileri

Zk Framework

ZK, en genel tanımıyla Java ile yazılmış Ajax tabanlı, açık kaynak ve sunucuda çalışan bir web ve mobil uygulama çatısıdır. Olay ve bileşen tabanlıdır. Zengin kullanıcı ara yüzü geliştirmeye olanak sağlar ve bunu yaparken az programlama bilgisi yeterlidir. Spring, Ibatis gibi bir çok yapı ve/veya framework ile uyumludur. Birçok büyük kuruluş tarafından desteklenmekte ve kullanılmaktadır.
ZK ile uygulama geliştirirken Javascript ve Ajax bilgisi zorunluluğu yoktur. ZK kendi ZUML dilini geliştirmiştir. ZUML/XUL/XHTML bileşenlerini içermektedir. Böylece ZK ile zengin içerikli XUL/XHTML bileşenleri ile uygulama geliştirilmektedir. Bu bileşenler tetiklenen çeşitli olaylarla çok etkin hale getirilebilmektedir.

XHTML: En genel tanımı XML sözdiziminin HTML içinde kullanımıdır. Amaç HTML dilinin tarayıcılardaki yorumlanma farklılıklarını ortadan kaldırmak, bununla birlikte HTML söz dizemindeki düzensizlik ve hatalardan kurtulmaktır. XHTML bir web standartıdır.W3C tarafından önerilmiştir.

XUL: XML kullanıcı ara yüzü tanımlama dilidir. Mozilla tarafından geliştirilmiştir. Firefox tarayıcısı XUL ile geliştirilmiştir [18].

ZUML: Zengin kullanıcı ara yüzleri tanımlamak için ZK tarafından geliştirilmiş XUL tabanlı bir işaretleme dilidir. XUL standartlarını kullanır ancak bazı eklemeler de yapılmıştır. ZUML aynı sayfada birbiri içinde farklı işaretleme dillerinin kullanılmasına izin verir. İçinde gömülü olarak Java yazılabilir ve bileşenler üzerinden deyim kullanılarak çeşitli işlemler yapılabilir [19].

Spring

Spring, kurumsal uygulamaları geliştirme süreçlerini kolaylaştırmayı amaçlayan bir uygulama geliştirme çatısıdır. Spring oldukça kullanışlı birçok özelliği bünyesinde barındırır. Bu özellikler, çatıyı oluşturan yedi farklı birim tarafından içerilir.

Bunlar;

- Spring AOP
- Spring ORM
- Spring DAO
- Spring Web
- Spring Bağlamı
- Spring Kabuğu
- Spring Web MVC

Bağımlılık iletimini sağlayan kabuk birimi, uygulama çatısının en temel birimidir. Bu birim, geliştiriciye çekirdek kozasının işlevlerini yönetme olanağı verir. Kabuğun içerdiği temel kavram olan "BeanFactory", çekirdek oluşumda tekilliğin sağlanması gerekliliğini ortadan kaldırıp, bağımlılıkların biçim ve tanımlarını eşlemeyi programlama mantığından dışlar.

Kabuk biriminin üstünde, çekirdeklere uygulama çatısı tarzında erişimi sağlayan, bağlam birimi bulunur. Bağlam birimi, özelliklerini çekirdek biriminden kalıtır. Bu özelliklere metin tabanlı ileti gönderme desteği gibi özellikleri de ekler.

DAO birimi, çaba gerektiren JDBC kodlama ve veri tabanı üreticisine bağımlı hata kodlarını ortadan kaldıran JDBC soyutlama katmanını sağlar. JDBC paketi, sadece özel ara yüzleri gerçekleştiren sınıflara değil, tüm POJO'lara da birim işlem yönetimi gerçekleştirmek üzere programlama ve tanımlama yöntemleri sunar.

ORM birimi, JDO, iBatis ve Hibernate de dahil olmak üzere, popüler nesne-ilişkisel eşleme API'lerine bütünleştirme katmanı sunar. Geliştirici, ORM birimini kullanarak, tüm bu nesne-ilişkisel eşleyicileri Spring'in diğer tüm imkânlarıyla birlikte kullanabilir.

Spring'in AOP birimi, yönelim tabanlı programlama gerçekleştirimi olan AOP politikası sunar. Bu politika sayesinde, geliştirici kodunda bağımlılık yaratacak eşlemelerden uzak, işlevsel tanımlamalar yapabilir. Kod düzeyi ara-veri (meta-data) işleviyle, tüm davranışsal veriler koda dahil edilebilir. Spring'in Web birimi, Servlet dinleyicileri ve web tabanlı uygulama bağlamını kullanarak bağlamlara ilk değer atama gibi temel web tabanlı bütünleşim özelliklerini sağlar. Ayrıca bu birim, Spring'in WebWork ya da Struts ile birlikte kullanıldığı durumlarda bütünleşimi de gerçekleştirir.

Spring'in Web MVC birimi, web uygulamaları için Model-Görünüm-Denetim gerçekleştirimini sağlar. Spring'in MVC gerçekleştirimi, yalnızca basit bir gerçekleştirim değil, Spring çatısının tüm özelliklerine (örneğin; geçerlilik denetimi) taban olan, model ile etki alanı arasında kesin ayrımı sağlar [20].

iBatis

iBatis çatısı, Java ve .NET geliştiricilerine veriye ulaşmada yardımcı olmak amacıyla hazırlanmış açık kaynak bileşenlerinden oluşan bir yapıdır. İlişkisel veri tabanına erişim sırasında genel olarak kullanılan yol JDBC yaklaşımıdır. Fakat bu yöntem okunurluğu az, gereğinden fazla şişmiş ve iş mantığı üzerinde bir etkisi olmayan

karmaşık kodlar ortaya çıkarmaktadır. SQL Maps, bu yaklaşımla birlikte gelen fazla kodları azaltmayı sağlar [21].

Itext Report

Java ve C# için PDF dokümanı oluşturma kütüphanesidir. Bu kütüphane kodlarıyla PDF belgeleri üzerinde değişiklikler yapılabilir. Jasper Report gibi raporlama araçlarında PDF görüntüleme için kullanılır.

Oracle Database

Oracle firması tarafından geliştirilen Veri tabanı Yönetim Sistemidir. Kendi içerisinde yönetilebilir bir yapıya sahiptir. Bu çalışmada yazılım için Oracle ürünlerinden sadece veri tabanı kısmı kullanılmıştır.

Apache Tomcat Server

Tomcat, Servlet barındırıcı ve Java Server Page (Java sunucu sayfası) uygulama programıdır. Servlet barındırıcıları iki kısımda ele alınabilir.

Stand-Alone Servlet barındırıcı: Bunlar web sunucuların önemli bir kısmını oluştururlar. Tek bir program vardır ve gelen tüm istekleri karşılar. Servlet barındırıcı tek başına da kullanılabilir ama statik sayfalardaki performansı asla popüler web sunucular kadar iyi olamaz. Tomcat tarafından kullanılan varsayılan moddur.

Inprocess Servlet barındırıcı: Java barındırıcı uygulaması ve web sunucu eklentileri birleşimi olan servlet barındırıcılardır. Web sunucu eklentisi, web sunucunun adres uzayında bir Java Sanal Makinesi açar ve Java Barındırıcının çalışmasını sağlar. Bir in-process barındırıcı çoklu-çoğullama tek işlemcili sunucular için uygundur ve çok iyi performans sağlamaktadır fakat sağlamlık konusunda sınırlamalar vardır [22].

Java Programlama Dili

Java Programlama dili şu anda dünyadaki en popüler programlama dillerinden biri haline gelmiştir. Java SUN bilgisayar şirketince orijinal olarak elektrikli ev araçlarının (mikrodalga fırınları, buzdolapları, televizyonlar, uzaktan kumanda cihazlari vs.) birbiriyle haberleşmesini sağlamayı amaçlayan bir proje içerisinde

1991 yılında geliştirilmeye başlanmıştır. Orijinal adi bu dilin yaratıcıları James Gosling, Patrick Naughton, Chis Wartdh, Ed Frank ve Mike Sheridan tarafından Oak olarak konulan programlama dili daha sonra bu isimde başka bir programlama dili olduğu keşfedilince o anda bir kahvehanede kahve içen programlama gurubu tarafından kahve markasından esinlenerek Java olarak değiştirilmiştir. Akıllı elektronik ev araçları pazarı SUN gurubunun tahminlerinden çok daha yavaş bir gelişme gösteriyordu. Bu yüzden Java dili projesi ticari bir geliştirme projesi olarak büyük olasılıkla iptal edilmekteydi. 1993 Yilinda "World Wide Web" büyük bir atılım göstererek bütün dünyaya yayılmaya başladı. Java'nın Dinamik Web sayfaları hazırlamadaki büyük potansiyelini gören SUN şirketi projeyi bu tarafa yönlendirdi ve bu Java'ya yeni bir canlılık ve yaşama umudu sağladı. Mayıs 1995 de SUN Java'yı büyük bir konferansta tanıttı. Program iş dünyası tarafından derhal büyük bir ilgiyle karşılandı. Java modern bilgisayar dünyasının ses, grafik işlem, haberleşme gibi ihtiyaçlarına cevap verebilen ve ticari gayeler için hazırlanan bir program dili olarak daha önceki bilgisayar dillerinin hiç birinin kapsayamadığı özellikleri içermekteydi. Bunun yanı sıra dil komut yapısı olarak C++ diline çok yakın olması da öğrenilmesini kolaylaştırıyordu. SUN Java'yı "World Wide Web" de kullanmak isteyen herkese ücretsiz olarak sundu. Java internette yayınlanmasının ardından çok büyük bir patlama yaşadı. 1997 ye gelindiğinde dünyadaki bütün bilgisayar okullarında temel bilgisayar dili olarak gösterilmeye başlandı. Dünyada şu anda hala en çok kullanılan bilgisayar dili olan C++ dilinin yapılan hataları tam olarak denetlememesi programın çalışma hızını arttırma yönünden iyi bir özellik olsa da profesyonel programcılar dışında kullanılmasını sınırlandırıcı bir etki yapıyordu. Java ise bütün hataları bildiren yapısı ve modern bilgisayarın bütün fonksiyonlarına ulaşabilen kütüphaneleriyle programcıların çok daha kolaylıkla öğrenebileceği bir dildir. Burada hemen şunu da belirtelim. C dili hızlı çalışma amacı birinci planda tutularak yaratılmış bir dildir. Java'da ise emniyet ilk planda yer almıştır. Hız açısından düşünüldüğünde Java C (ve C++) diliyle rekabet edemez. Zaten program derleyicisi de C++ dilinde yazılmıştır. Java'nın diğer önemli bir temel özelliği Nesneye yönelik bir dil olmasıdır. Nesneye yönelik diller, nesneleri gerçek dünyadakine daha benzer bir yapıda tanımlayarak anlaşılmalarını kolaylaştırırlar. Nesneleri gerçek dünyadaki gibi masa, sandalye, bilgisayar, gerçek gaz gibi

tanımlayarak programlamak insan beyninin anlaması açısından çok daha kolaydır [23].

2.1.2.2. Uygulama Yazılım Yapısı

Yazılımda 3 katmanlı mimari yapı kullanılmıştır. Sunum, İş ve Veri katmanları katmanlı mimari yapısını oluşturmaktadır. Sunum katmanında Zk Framework kullanılmıştır. Her ekran için ZUL uzantılı bir dosya oluşturulmuştur. Yazılıma ait dizin dosyaları hiyerarşik bir yapıya sahiptir. Ekran tasarımları kısmını oluşturan dosya dizinleri, kategoriler şeklinde oluşturulmuştur. Evrak İşlemleri altında evraklara ait dosyalar, Kullanıcı işlemleri altında kullanıcı dosyaları bulunmaktadır. Bu dosyalar yazılımda ekran görüntülerini oluşturmaktadır. İstekler bu ekranlardan alınır iş katmanına ulaştırılır. Ekranlar bileşenlerden oluşur. Her bileşen bir yapıyı temsil eder. Örneğin tablo yapısı, buton yapısı, metin giriş yapısı gibi bileşenler mevcuttur. Sonuçta ekran bileşenleri bir araya gelerek ekran görüntüsünü oluşturmaktadır. Yazılım için ilk ekran giriş ekranı dosyasıdır.

Sunum katmanında giriş ekranı için giriş.zul dosyası eklenmiştir. Bu dosya yazılım giriş ekranı için tasarım oluşturmaktadır. Ekrandan bilgilerin girilmesi için Kullanıcı adı ve Şifre metin kutuları bileşen olarak eklenmiştir. Metin kutularından alınan istekler iş katmanına aktarılır ve sonraki işlemler gerçekleştirilir. İş katmanında giris.zul dosyası için GirisWindow.java klasörü oluşturulmuştur. İş katmanında her zul uzantılı ekran için bir tane Java dosyası bulunmaktadır. Bu Java dosyalarında son ek olarak "Window" eki bulunmaktadır. GirisWindow.java dosyası Java programlama dilinden oluşmaktadır. Bu dosyada kullanılan kütüphaneler Zk Framework kütüphanesine ait bileşenlerdir. Sunum katmanında kullanılan bileşenlerin aynısı GirisWindow.java dosyasında değişkenler ile kullanılabilir. Bu değişkenler iş katmanında kontrollerin sağlanmasını sağlayacaktır. Örneğin sunum katmanında Zk Framework ile oluşturulan Textinput bileşeninin tanımı GirisWindow.java klasöründe getfellowifany yordamıyla bir değişkene değeri aktarılabilir. Bu yordamda yine Zk Framework kütüphanesiyle eklenen bir metottur. Değişkenlerden alınan veriler kontrol için kullanılacaktır. İstekler veri katmanına

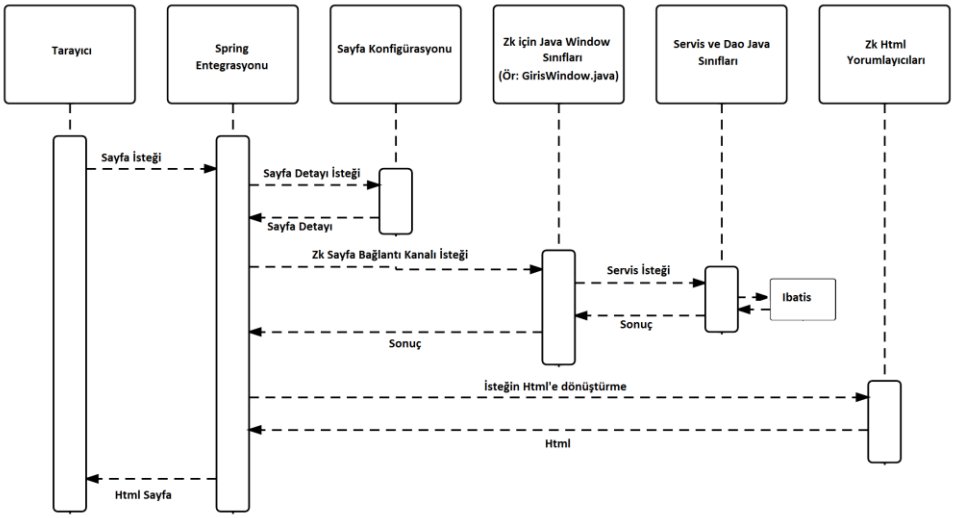
ulaştırılmadan önce bu katmanda işlemlerin kontrolleri yapılacaktır. Örneğin kullanıcı şifre politikasına uygun olmayan bir şifre girilirse iş katmanındaki kontrollere takılır. Kontrol algoritmaları Java programlama dili ile kodlanmıştır. İş katmanında işlemler tamamlandığında, istekler artık veri katmanına aktırılabilir. Veri katmanı için görev yapan dosyalar GirisService.java ve GirisDao.java dosyalarıdır. GirisService.java dosyasına gelen istekler kontrolleri yapıldıktan sonra GirisDao.java dosyasında istekler veri tabanında tablolarda işlenir.

İstekler 4 farklı şekilde ele alınır;

- Verilerin veri tabanından ekrana yansıtılması
- Verilerin ekrandan veri tabanına kaydedilmesi
- Verilerin ekrandan veri tabanına güncellenmesi
- Verilerin ekrandan veri tabanından silinmesi

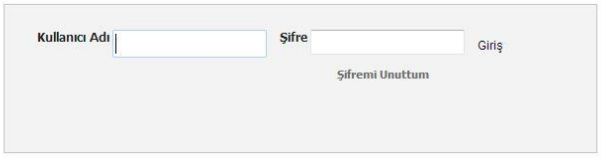
Yazılımda Veri katmanının son kısmında Ibatis ve Spring teknolojileri kullanılmıştır. Ibatis teknolojisi ile veri tabanı tablo verileri, Java nesnesi halinde elde edilerek iş katmanına aktarılır. Ibatis konfigürasyonu Spring teknolojisi ile sağlanır. Spring teknolojisi 3 katmanda da işlem yapar. Spring teknolojisi, uygulama ile ilgili gerekli yapıların konfigürasyonu sağlar. Örneğin GirisService.java dosyasında GirisDao.java nesnesinin değerlerininin yüklenmesini sağlar. Bunun yanında ayrıca GirisDao.java dosyasında kullanılan Ibatis XML dosyalarının yüklenmesini sağlayarak yazılım için kod yapılarını hazır hale getirir.

Sistem katman yapısı genel olarak Şekil 2.7’ de görüntülenmektedir. Şekil çizimi için materyal olarak Microsoft Visio kullanılmıştır.



Şekil 2.7. Sunucu istemci etkileşim diyagramı

Uygulamadaki tüm süreçler Şekil 2.7'deki işlem sırasını gerçekleştirmektedir. Uygulama için ilk giriş ekranı tasarlanmıştır (Şekil 2.8).



Şekil 2.8. Kullanıcı girişi ekranı

Uygulama arka planında kullanıcı rol yapısı bulunmaktadır. Giriş yapan kullanıcıların rolüne göre uygulama menüleri listelenir.

2.1.2.3. Kullanıcı Rol Yapısı

Kullanıcı rol yapısı farklı bilgilerin bir araya gelmesinden oluşmaktadır. Her kullanıcının bir veya birden fazla rolü bulunabilmektedir. Bu durumda oluşan bilgiler; Kullanıcı bilgisi, Rol bilgisi ve Kullanıcı Rolü ilişkilendiren Kullanıcı Rol bilgisidir. Ayrıca her rolün işlemleri bulunmaktadır. Bu işlemler, yetki olarak tanımlanmıştır. Bundan dolayı yetki bilgisi ve rol yetkisi bilgisine de ihtiyaç vardır. Sonuçta Kullanıcı Rol yapısı; kullanıcı, rol, yetki, kullanıcı-rol ve rol-yetki bilgilerinden oluşmaktadır. Bu yapı dikkate alınarak kullanıcılar sisteme giriş yapar ve yetkilerine göre ekranda menüler listelenecektir. Bu bilgileri elde etmek için Oracle Veri tabanı sisteminde tablolar tasarlanmıştır.

Tablolar;

- T_KULLANICI
- T_ROL
- T_MENU
- T_KULLANICI_ROLLER
- T_MENU_ROLLER

Yazılımda oluşturulan menü bilgileri aşağıdaki gibidir;

- Gelen Evrak Ekle
- Giden Evrak Ekle
- Taslaklar
- Gelen Evraklar
- Gönderilen Evraklar

2.1.2.4. Evrak Kayıt

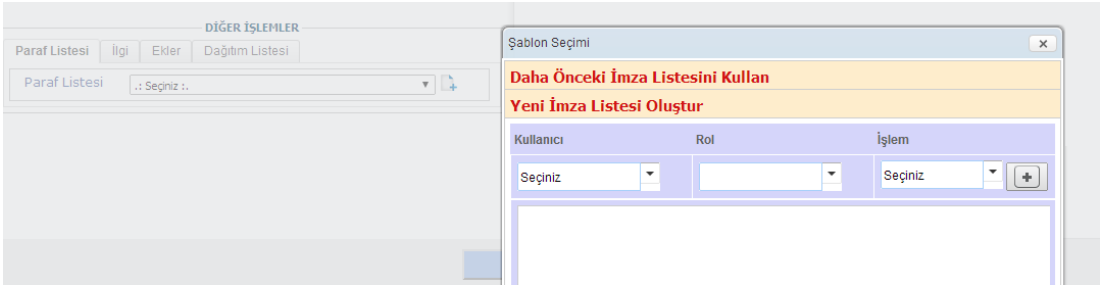
Evrak kaydedilirken uygulama veri katmanında evrak, evrak belge, evrak imzacı listesi, evrak ilgi(Evrak ilgisi varsa), evrak eki(Evrak eki varsa) ve evrak gideceği yerlerin listesi bilgileri, 6 farklı tabloya kaydedilmiştir.

Tablolar;

- T_EVRAK
- T_EVRAK_BELGE
- T_EVRAK_IMZA_LISTESI
- T_EVRAK_ILGI
- T_EKLER
- T_EVRAK_HEDEF

Evrak tablosuna, Şekil 2.3'de sol kısımda gösterilen bilgiler kaydedilmiştir. Evrak tablosu kayıt işleminde ana tablodur. Ayrıca bu tablo evrak ile ilgili meta verileri tutmaktadır. Evrak belge tablosu Şekil 2.3'de sağ tarafta gösterilen metin bilgisi kaydını tutmaktadır. Evrak belge hali oluşturulurken bu tablodan okunan metin bilgileri, belge üzerinde metin kısmını oluşturur. Evrak ile ilgili ekranda çoklu alan olarak eklenen bilgiler için ayrı tablolar oluşturulmuştur. Çünkü tabloların kayıt sayısı kullanıcıya bağlıdır. Sonuçta veri tabanı normalizasyonu açısından iyi bir tasarım oluşturulması gerekiyor. Bundan dolayı evrak imzacı listesi, evrak ilgi, evrak ekleri ve evrak gideceği yer gibi çoklu kayıt yapılan alanlar için ayrı tablolar oluşturulmuştur.

Analiz ve tasarım kısmında evrak imzacı listesinin yapısı ele alınmıştır. Bu yapıya uygun olarak uygulamada Şekil 2.9'daki gibi ekran hazırlanmıştır.

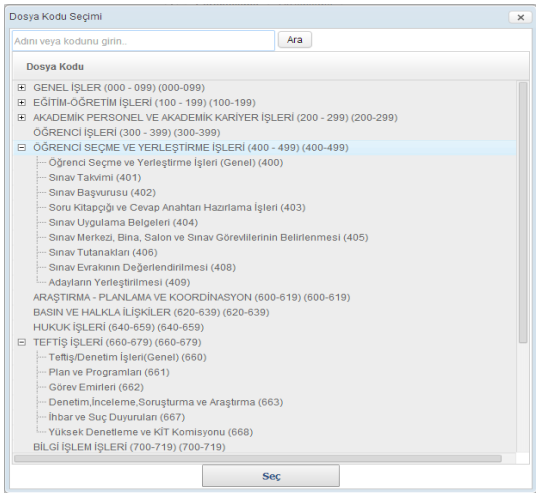


Şekil 2.9. Evrak imzacı listesi açılır ekranı

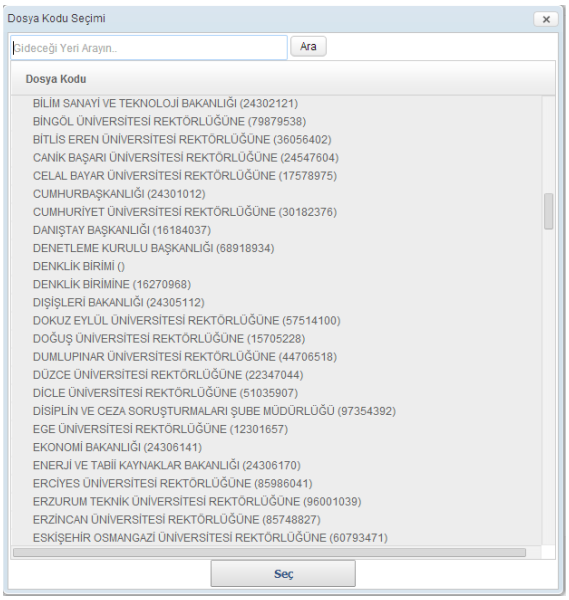
İmzacı listesi oluşturulurken ilk olarak kullanıcı seçimi yapılır. Kullanıcı seçiminden sonra kullanıcıya ait rol listesi ekrana gelir. İlgili rol seçildikten sonra imza türü seçilir ve imzacı listesine eklenir. Açılır listeler için kayıtlar değer ve değer etiketi olarak yüklenir. Bu tür veri yüklemeleri için sonu Window ile biten Java kontrol sınıflarının üst sınıfı olan ErpWindow Java sınıfında ortak olarak kullanılan bir yordam oluşturulmuştur. Bu yordama Zk Framework kütüphanesine ait Listbox ekran bileşeni ve yüklenecek veri yapısı parametre olarak eklenir. Kullanıcı, Rol ve İmza listesi işlemi için bu yordam ortak olarak kullanılmıştır.

Şekil 2.3 evrak bilgilerinin kaydı için tasarlanmıştı. Bu ekran üzerinde imzacı listesi oluşturulurken Şekil 2.9'da imza listesi için ayrı açılır ekran eklenmiştir. Ekranda oluşturulan imzacı listesi, evrak kaydedilirken veri tabanında evrak imza listesi tablosuna kaydedilir. Veriler liste halinde olduğu için tabloda çoklu kayıt gerçekleşir.

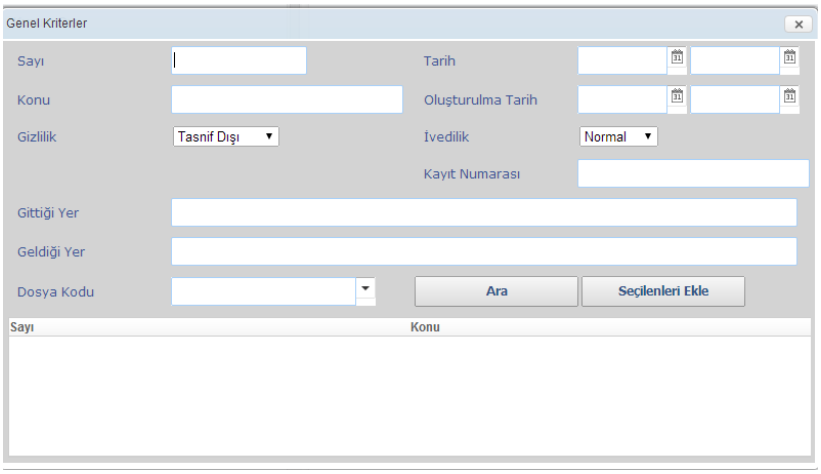
Evrak imzacı listesi gibi evrak ilgi, evrak gideceği yerler ve evrak dosya kodu alanları için de ayrı açılır ekranlar tasarlanmıştır. Kullanıcı, Rol ve İmza listesi gibi Dosya kodu ve Evrak gideceği yer bilgilerinin veri yüklemesi değer ve değer etiketi şeklinde olmuştur. Benzer şekilde bu ekranlarda da ErpWindow Java sınıfında bulunan ortak yordamlar kullanılmıştır.



Şekil 2.10. Evrak dosya kodu ağaç yapısı ekranı

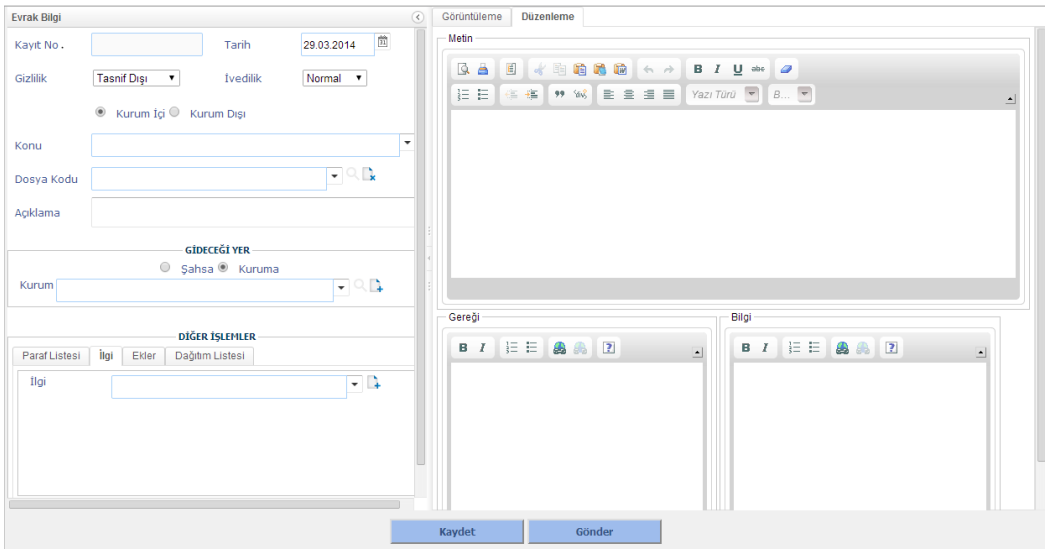


Şekil 2.11. Evrak gideceği yer ağaç yapısı ekranı



Şekil 2.12. Evrak ilgi ekranı

Ana ekran olarak Şekil 2.3'deki gibi verileri temsil eden ekran hazırlanmıştı. Şekil 2.3'deki ekran iyileştirilerek Şekil 2.13'daki gibi son halini almıştır.



Şekil 2.13. Evrak bilgi girişi tüm ekran

Ekranda evrak kayıt ile ilgili tüm bilgiler için giriş alanları tasarlanmıştır. Giriş alanlarının hangi tabloya kaydedileceğinden bahsedilmiştir. Evrak bilgileri kayıt işlemi Şekil 2.7'daki yazılım işlevini gösteren mantıkta kaydedilmiştir. Dinamik model yapısı olan Java kütüphanesine ait HashMap modeli kullanılarak veriler veritabanına aktarılmıştır. Uygulama geliştirmenin genel yapısında model olarak HashMap nesnesi kullanılmıştır. HashMap nesnesi Java programlama diline ait tanımlanmış bir kütüphane sınıfıdır. Bu sınıf veri yapıları grubuna girmektedir. Veri yapısı olarak esnek halde tasarlanmıştır. Veri eklendiğinde bellek alanı otomatik olarak artar veya veri eksiltildiğinde otomatik olarak azalacatır. HashMap kütüphanesinin en büyük avantajı değişken ismiyle indekslenmiş olmasıdır. Değerler indeks ile değil isimler ile tutulmaktadır. Veri tabanından gelen bilgiler isimler ile geldiğinden bu çalışmada verimli hale gelmiştir. HashMap'in dezavantajı içerisinde tutulan veri setleri sıralı değildir.

Evrak kayıt işlemi tamamlandıktan sonra ayrıca evrakın belge hali kaydedilir. Kayıt edilen belge, dizinde kriptolu şekilde bulunmaktadır. Evrak belgesi yazılımda kriptolama algoritması tarafından çözülebilir. Belge için kripto yöntemi şifre tabanlıdır. Öncelikle kripto için bir şifre belirlenir. Ondan sonra kripto işlemi

gerçekleştirilir. Kullanılan algoritma MD5 ve DES algoritmalarıdır. Algoritmalar açık kaynak kütüphanelerden elde edilmiştir. MD5 ve DES algoritmaları kriptolama işlemlerinden kullanılan algoritmalardır [24,25].

Evrak kaydedilirken oluşan belge, ekranlarda görüntüleme şekli Ergonomik Tasarım başlığı altında ele alınacaktır.

2.1.2.5. Evrak İş Akışı

Evrak iş akışı uygulamada dinamik bir yapıda oluşturulmuştur. Dinamik akışı oluşturan yapı, evrak için oluşturulan imzacı listesi ile mümkün hale gelmiştir. Evrak kaydedilirken Şekil 2.14'deki gibi bir imza listesi oluşur.

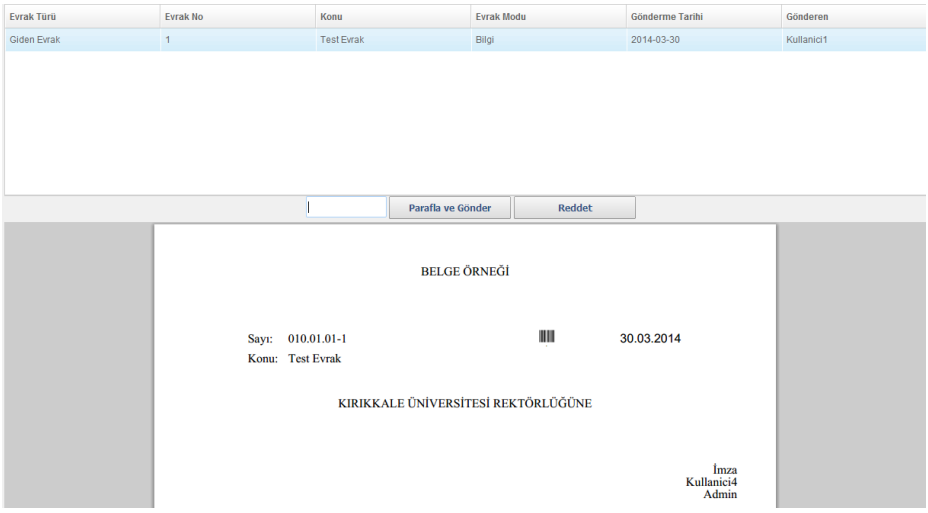
Şekil 2.14. Eklenen imzacı listesi ekranı

İş akışının başlangıcı Şekil 2.13'de Gönder butonu ile başlatılır. Gönder butonu uygulamada evrak için tekrar güncelleme yaparak gönderim işlemini gerçekleştirir. Çünkü ekran üzerinde değişiklik yapılmış ve kaydet butonuna basılmamış olabilir. Güncelleme işleminden sonra evrak iş akışı için kayıt bilgilerini bulunduran T_EVRAK_GONDER tablosu ile başlamış olacaktır.

Tablo;

- T_EVRAK_GONDER

Evrak gönder tablosu, imzacı listesinde bulunan kullanıcıların işlemlerini gerçekleştirirken akış bilgilerini kayıt eden tablodur. Kullanıcılar imza işlemini yaptıklarında tabloya bir kayıt eklenir. Tabloda, eklenen imza kaydı için önceki imza kaydına bağlı olduğunu gösteren tablo sütunu mevcuttur. Evrak iş akışı Şekil 2.13'de görüldüğü gibi listenin en başında bulunan Kullanici1 ile başlar. Kullanici1 uygulamaya giriş yaptığında Gelen Evraklar menüsüne tıkladığında Şekil 2.15'deki gibi ekran açılacaktır.



Şekil 2.15. Evrak listesinde evrak belge görüntüsü

Şekil 2.15'de de görüldüğü gibi kullanıcıya hangi imza türü ile geldiği buton ile ifade edilmiştir. Parafla ve Gönder butonu tıklandığında akışta bulunan sıradaki kullanıcıya evrak gönderilecektir. Akışta bulunan kullanıcılar kendisine gönderilen evraklardan birini seçtiğinde Şekil 2.15'deki gibi ekran açılacaktır. Ekranda Parafla ve Gönder butonu tıkladığında evrak gönder tablosuna ve evrak imzalama bilgilerini tutan tabloya birer kayıt eklenir.

Evrak imza bilgisini tutan tablo;

- T_EVRAK_SIGNS

Parafla ve Gönder butonu ayrıca dizinde bulunan evrak belgesi üzerinde dijital imza işlemini gerçekleştirir. Dijital imza işlemi özel anahtar ve sertifikaları içeren Java kütüphanesine ait JKS özel imzalama dosyası ile yapılmaktadır. Her kullanıcı için özel oluşturulmuş bir JKS dosyası bulunmaktadır. JKS dosyaları komut satırı ile oluşturulmaktadır [26].

Komut satırı ile oluşturulan imza, yazılımın kendi içerisinde kullandığı dijital imza yöntemidir. Bu yöntemle birlikte yazılıma ait özel bir imza sistemi kullanılmış olacaktır.

Parafla ve Gönder butonu yanında Reddet butonu bulunmaktadır. Bu butona tıklandığında akışta gerçekleştirilmiş imzalar veri tabanında silinir ve ayrı bir tabloya yedeği alınır.

Bu tablo;

- T_EVRAK_SIGNS_DELETED

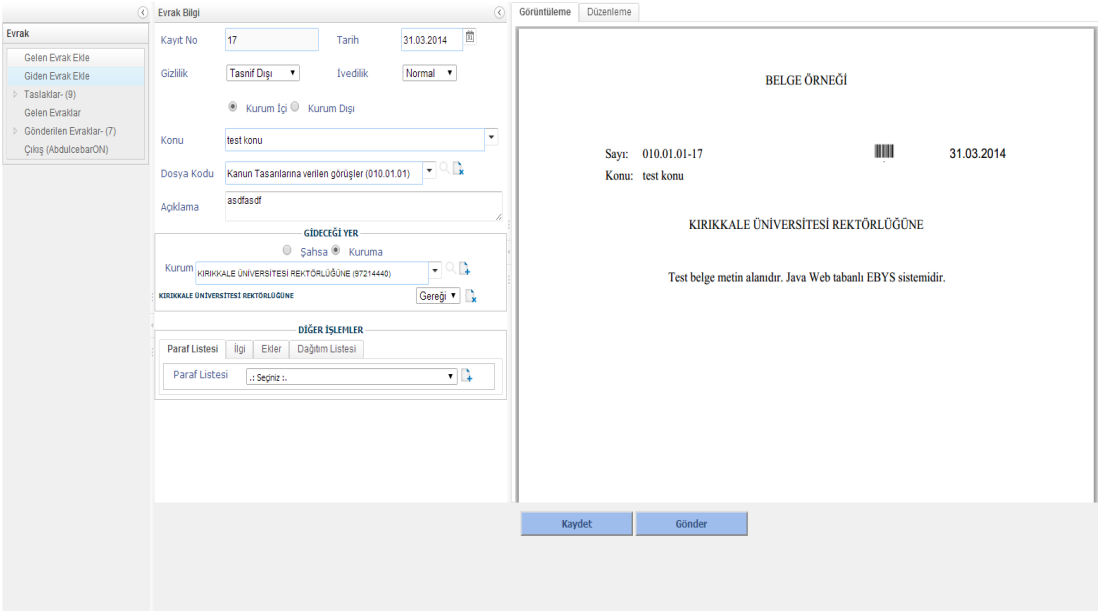
T_EVRAK_SIGNS_DELETED tablosuyla evrak iş akışında bulunan kullanıcılardan birinin evrakı reddetmesi ile akışta daha öncesinde gerçekleştirilmiş imzalarının veri tabanında tamamen silinmesinin önüne geçilmiştir.

Evrak imza akışı tamamlandığında akışta bulunan her kullanıcı evrakın son durumunu görüntüleyebilmelidir. Elektronik Belge Yönetim Sistemlerinde en büyük sorunlardan bir tanesi de evraklarla ilgili son durum bilgilerin görüntülenmemesidir. Ekranlarda bilgi sunumu çoğaldıkça karmaşıklık ona göre artmaktadır. Çözüm olarak ekranlarda ergonomik tasarımlar oluşturulmuştur.

2.1.2.6. Ekranlardaki Ergonomik Tasarım

Analiz ve tasarım bölümünde ekranların ara yüzü tasarımı hakkında bilgiler ele alınmıştır. Evrak kayıt ekranında kullanıcının evrak kaydı ile ilgili tüm bilgilere ulaşması gerekir. Şekil 2.13'da tasarlanan ekran bu yapıdadır. Ana ekrandan

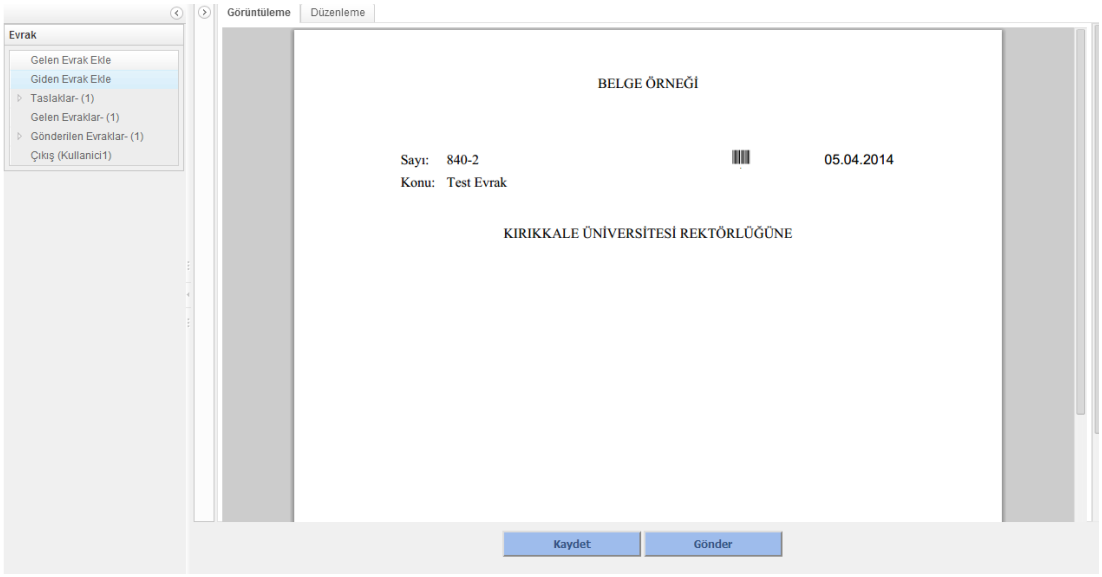
ayrılmadan dosya kodu, gideceği yer bilgileri, evrak ekleri ve evrak ilgileri ana ekran üzerinden açılır ekran oluşturularak yazılım geliştirilmiştir. Ekran üzerinde kaydedilen bilgiler ile ekranda evrakın belge hali görüntülenerek kullanıcıya doküman olarak görüntülenmesi sağlanmıştır (Şekil 2.16).



Şekil 2.16. Ana menüler ve evrak giriş ekranı

Belge dosya dizininde kriptolu olduğundan dolayı dosya kripto çözümlemesine ihtiyaç duyulmaktadır. Kripto algoritmaları için daha önce belirlenmiş şifre ile dosya çözümlenmesi sağlanabilir. Bu şekilde evrakın belge halinde görüntülenmesi sağlanmıştır. Evrak görüntülenmesi kullanıcılar tarafından talep edilen bir yöntemdir. Yükseköğretim Kurulu Elektronik Belge Yönetim Sistemi uygulamasında kullanıcıların talep ettiği evrak kayıt sistemi, girilen verilere karşılık evrak belge şeklinin oluşmasıdır. Sonuçta girilen evrak bilgilerinin belge hali oluşmuş ve kullanıcı, bilgilerini doğru girdiğine emin olacaktır. Ekranda belge yüklenirken öncelikle bilgisayar dizininde geçici bir klasörde kriptosu çözülmüş halde kaydedilir. Belge ekrana yüklendikten sonra geçici klasörde silinecektir.

Ekranlar genişletilebilir ve daraltılabilir şekilde tasarlanmıştır. Şekil 2.16'da gösterilen ekran daraltılarak Şekil 2.17'deki gibi yapılabilir.



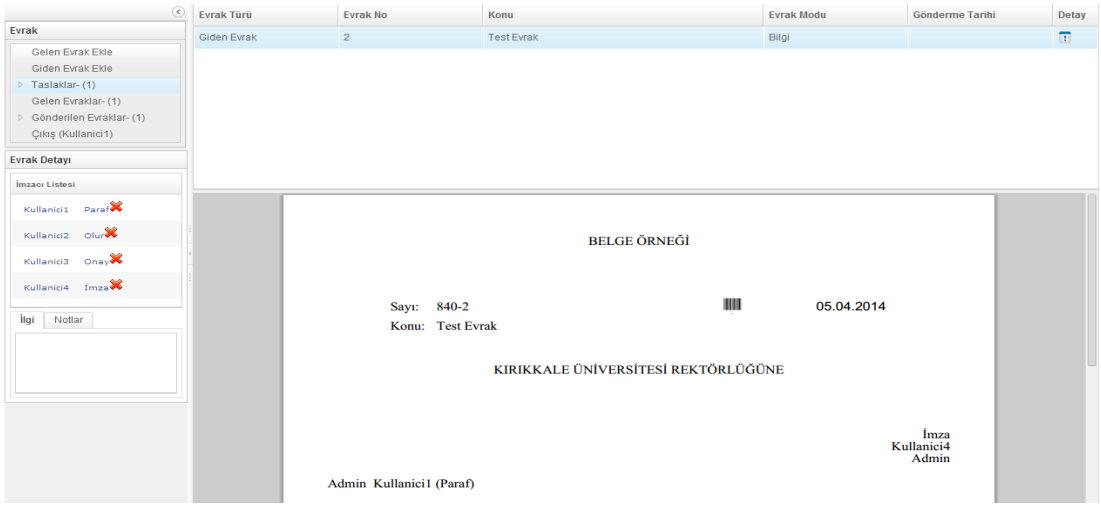
Şekil 2.17. Evrak belge görüntüsü

Şekil 2.17'de gösterilen evrakın tam ekran olarak görüntülenmesi sağlanmıştır. Aynı şekilde menüler de daraltılarak ekran daha da genişletilebilecektir (Şekil 2.18).

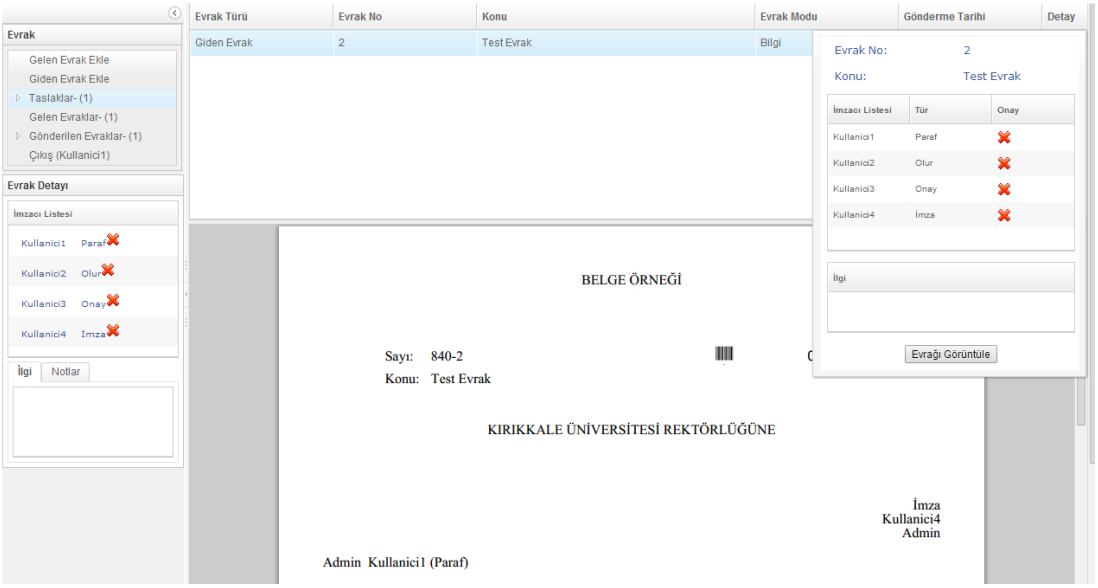
Şekil 2.18. Büyütülmüş evrak belge görüntüsü

Ekranların daraltılıp genişletilmesi kullanıcı açısından kullanım kolaylığı sağlamaktadır. Şekil 2.16'da gösterilen ekranın sol üst kısımlarında ekran esnekliğini sağlayan yönlendirme çubukları bulunmaktadır. İşletim sistemlerinde kullanılan masaüstü uygulamaları genellikle bu şekilde tasarlanmaktadır. Ekran esneklikleri, iş çatısı olarak kullanılan ZK Framework kütüphaneleri ile sağlanmıştır. Bu işlemler tamamen istemci tarafında yapılmaktadır. Dolayısıyla sunucu tarafında iş yükü daha az olacaktır.

Evrak ile ilgili kayıt işlemi tamamlandıktan sonra evrak taslaklara kaydedilir. Taslaklar menüsünden evrak tekrar yüklenip, kullanıcı evrakta kaldığı yerden devam edebilecektir (Şekil 2.19, Şekil 2.20)



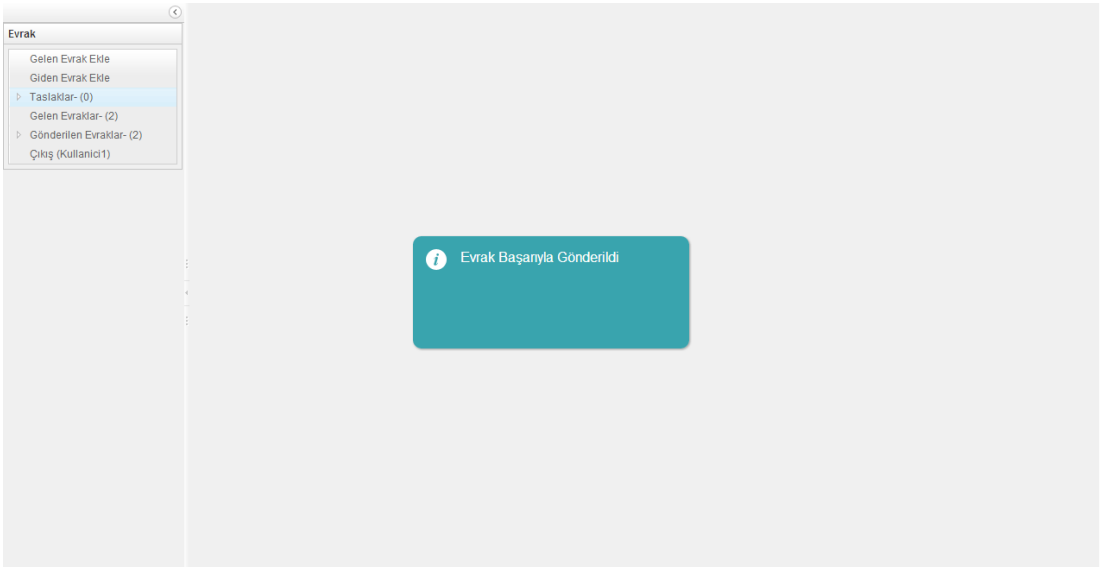
Şekil 2.19. Taslak halinde evrak bilgisi ve belge görüntüsü



Şekil 2.20. Evrak bilgisine ait açılır ekran görüntüsü

Şekil 2.19'da gösterilen ekranda sağ üst kısımda evrak detay bilgisini gösteren küçük bir ikon bulunmaktadır. İkona tıklandığında Şekil 2.20'deki gibi sağ üst tarafta açılan bilgi amaçlı küçük bir ekran açılacaktır. Açılan küçük ekran üzerinde evrak görüntüle butonu mevcuttur. Bu butona tıklandığında Şekil 2.16'daki ekran açılacaktır. Böylece taslak olarak bulunan evrakta güncelleme işlemine devam edilmesi sağlanmıştır. Ekranın herhangi bir kısmına tıklandığında açılmış olan küçük ekran tekrar kapanacaktır. Burada amaçlanan durum, ekranlarda temel bilgilerin açık olarak verilmesi, detay bilgilerin ise küçük ikonlar ile sunulmasıdır.

Yazılımda yapılan her işlem sonrası için kullanıcıya işlemiyle ilgili bilgi mesajı verilmelidir. Örneğin; evrak kaydı yapılırken 'Evrak Başarıyla Kaydedildi' şeklinde mesaj açılacaktır. Mesaj bilgi kutuları tamamen zaman duyarlı notifikasyonlar ile yapılmıştır. Şekil 2.16'da evrak gönder butonu ile evrak gönderim işlemi yapıldığında Şekil 2.21'deki gibi bilgi notifikasyonu açılacaktır.



Şekil 2.21. Ekran notifikasyon örneği

Ekrandaki notifikasyon 2 saniye bekledikten sonra mesaj bilgi kutusu kapanacaktır. Sonuçta ekran mesajlarını kapatmak için ayrı bir müdahale işlemi yapılmasına gerek kalmayacaktır.

Genel olarak ele alınan ergonomik tasarımlar araştırma verileri ve kullanıcıların geri beslemeleri ile oluşturulmuştur [10,27].

2.1.2.7. Veri Madenciliği ve Bulanık Arama Yöntemi

Kayıt işlemlerinde metin sınıflandırılması için veri madenciliği kullanılmıştır. Kayıt işlemi gerçekleştirildiğinde kayda benzer yapılar sistem tarafından önerilmiştir. Örneğin evrak konu başlığı girildiğinde yazılımda arka planda sınıflandırma mantığı ile oransal bir yapı oluşturulur. Konu başlığı bu oransal yapıya en uygun gideceği yerleri önermiştir. Yazılımda metin sınıflandırma algoritması için "TextClassifier" Java sınıfı kullanılmıştır. Bu sınıfta Weka MNB teoremi ele alınmıştır. Weka MNB teoremi kelimeler üzerinde sınıflandırma yapmaktadır. Kelimelerin sayısını tutan bir liste olması gerekir. Bu liste matematiksel olarak p(w,c) şeklinde tanımlanabilir. "w"

sembolü kelimeyi gösterir. ”c” sembolü kelimenin sınıfını gösterir. Eşitlik 1 kelimeler sayacını gösteren bir vektördür. Eşitlik 2 sınıf aralığını gösterir. N değeri ele alınacak kelime sayısı, M değeri ele alınacak sınıf sayısını gösterir.

$$w = [w_1, ..., w_n]1 \quad (2.1)$$

$$c : 1 \leq c \leq M \quad (2.2)$$

Bayes teoremi üretken ortak dağılım çarpanları için sınıf önceliği formülü p(c), sınıf koşulu formülü için p(w|c) kullanır. Sonuçta p(w, c) = p(c)p(w|c) şeklinde bir tanım olacaktır. Naive Bayes sınıflandırması ek varsayımını kullanır ve sınıf koşullu olasılıklar bu durumdan bağımsızdır. Bu durumlar dikkate alındığında $p(w|c) \propto \prod np(Wn, n|c)$ formülü ele alınır. Sonraki aşamada MNB teoremi 2 parametreyi ele alır. Bu parametreler sınıf koşullu olasılıklar ve Multinominal dağılımdır. Bu durumda oluşacak formül $p(w_n, n|c) = p(n|c)^{w_n}$ şeklinde olacaktır. Özetle ele alınacak MNB için son formül Eşitlik 3 gibi olacaktır [28].

$$p(\boldsymbol{w}, c) = p(\boldsymbol{w}|c)p(c) \propto p(c) \prod_{n=1}^{N} p(n|c)^{w_n} \quad (2.3)$$

Genel olarak bu formülleri içeren Weka Java Kütüphanesi kullanılmıştır. “TextClassifier” Java sınıfı ile Weka Kütüphanesi kullanılarak yazılım için uygun hale getirilmiştir. Bu Java sınıfında yordamlar bulunmaktadır. Yordamlar formüllerde kullanılan değişkenler için değerleri oluşturur. Öğrenme algoritmaları sınıfına giren Weka MNB yöntemi bilgileri işlemek için kendisine özel “arff” uzantılı dosyaları okuyabilmektedir. Arff dosya formatı diğer metin dosyalarından biraz farklıdır. Bu uygulama için Arff dosya formatı şu şekildedir;

```
@ATTRIBUTE evrakKonu string
@ATTRIBUTE gidecegiYer{PERSONEL DAİRE BAŞKANLIĞI,
            BİLGİ İŞLEM DAİRE BAŞKANLIĞI,
            İDARİ MALİ İŞLER DAİRE BAŞKANLIĞI}
@DATA
"35.madde", PERSONEL DAİRE BAŞKANLIĞI
"Öyp", PERSONEL DAİRE BAŞKANLIĞI
```

"Bilişim Kadro", PERSONEL DAİRE BAŞKANLIĞI
"İlanlar", PERSONEL DAİRE BAŞKANLIĞI
"Maaş", PERSONEL DAİRE BAŞKANLIĞI
"Öyp", BİLGİ İŞLEM DAİRE BAŞKANLIĞI
"Bilişim Kadro", BİLGİ İŞLEM DAİRE BAŞKANLIĞI
"Yazılım", BİLGİ İŞLEM DAİRE BAŞKANLIĞI
"Donanım", BİLGİ İŞLEM DAİRE BAŞKANLIĞI
"Donanım", İDARİ MALİ İŞLER DAİRE BAŞKANLIĞI
"DMO", İDARİ MALİ İŞLER DAİRE BAŞKANLIĞI
"Maaş", İDARİ MALİ İŞLER DAİRE BAŞKANLIĞI

Dosya örneği evrak bilgilerinin konuya göre gideceğe yer bilgisini tutmaktadır. Dosyada üst kısımda kural ve tanımlar bulunmaktadır. Bu tanımlara uygun olarak veri seti alınır. Veri seti yine aynı dosyada tanımlanır [29]. Veri setlerine göre yüzdelik sonuçlar alınır. Yüzdelik en yüksek olan bilgi ihtimali en yüksek olan bilgidir. Buna benzer olarak yazılımda veriler veri tabanından alınarak uygun bir format oluşturulmuştur. Evrak ile ilgili girilen alanlar tanım kısmını oluşturmaktadır. Belli bir evrak konusu sürekli olarak belli bir gideceğe yere göre kaydedilirse yüzdelik artacaktır. Bu durumda kayıt konu başlığı girildiğinde kullanıcıya gideceğe yer için oranı en yüksek olan kaydı önermiştir. İmza listesi önerisi, metin alanı için yazı önerisi, evrak için ilgi belge ekleme önerisi gibi kayıt işlemlerinde de aynı mantık kullanılmıştır.

Yazılımda ekranlarda arama işlemleri için bulanık arama yöntemi kullanılmıştır. Normal arama yöntemlerinde aranan kelimeler veya cümleler hedefte bulunan kayıtlar içinde benzer harfler ve harf sıralarını içeriyorsa listelemeyi gerçekleştirecektir. Ancak bulanık arama yönteminde yüzdelik benzerlik dikkate alınarak kayıtlar getirilmektedir. Bulanık arama yöntemi için kullanılan algoritma Levenshtein Distance yöntemidir. Eşleştirilecek kelimelerin birbirine olan uzaklık bilgisi alınmaktadır. Bu mesafe azaldıkça benzerlik artmaktadır. Levenshtein Distance yöntemi Eşitlik 4'te gösterilmiştir.

$$\operatorname{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \operatorname{lev}_{a,b}(i-1,j) + 1 \\ \operatorname{lev}_{a,b}(i,j-1) + 1 \\ \operatorname{lev}_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases} \tag{2.4}$$

Eşitlik 4'de a ve b kelimelerin uzaklıkları dikkate alınarak benzerlik yüzdeliği elde edilmektedir [30]. Bu yöntemin kullanılmasının amacı aranacak bilginin tam olarak hatırlanmaması durumunda, benzer kelimeler kullanılarak aramayı gerçekleştirmektir. Birçok bilimsel çalışmada bu yöntem kullanılmıştır. Norveç'te lehçelerin farkını ele almak için Levanshtein Distance yöntemi kullanılmış ve olumlu sonuçlar alınmıştır [31].

2.1.3. Test

Yazılım test aşaması, yazılım geliştirme yaşam döngüsünde önemli aşamalarından birisidir. Etkin test yapılması oldukça önemlidir. Yazılım testi, önceden belirtilen gereksinimleri karşılayıp karşılamadığını, doğru çıktıyı üretip üretmediğini kontrol eden yapıdır. Çok sayıda test durumu ve test stratejisi hazırlanır, bu testlerin hepsi belirli hedefler (tüm sorunları kaldırmak, yazılımın hatasız çalışmasını sağlamak ve en iyi çıktıyı elde etmek) için uğraşır. Çok sayıda test tekniği ve metodolojisi bulunur. Yazılım test metodolojisi, yazılım testi tekniklerinden farklıdır.

2.1.3.1. Yazılım Test Metodları

2 çeşit test metodu bulunmaktadır.

Bunlar;

- Kara Kutu (Black-Box) Testi
- Beyaz Kutu (White-Box) Testi

Kara Kutu Testi

Kara Kutu testlerini iki gruba ayırabiliriz, bunlardan biri kullanıcı gereksinimi duymayan testler, diğeri ise kullanıcının gerekli olduğu testlerdir.

Kullanıcı Gereksinimi Duymayan Test Metotları

Fonksiyonellik Testi: Fonksiyonel gereksinimlerini test etmek için yazılmıştır. Eğer uygulama beklendiği gibi davranırsa, testlerin yazılmasına sıralı bir şekilde devam edilir.

Stres Testi: Çok sayıda veri girişi, büyük nümerik değerler, çok sayıda sorgu olduğu zamanlarda kullanılır ve uygulamanın dayanıklılığını belirler.

Yükleme Testi: Bir sistemin performansını derecelendirmek veya ağır yükler ve çok sayıda veri girişinde sistemin hangi noktada çakılacağını tespit etmek için kullanılır.

Ad-Hoc Testi: Geçerli bir test oluşturulmadan önce kullanılmalıdır. Bu test diğer testlerin kapsama alanını ve diğer testlerin süresinin belirlenmesinde yararlı olur.

Araştırma Testi: Ad-Hoc testine benzer ve uygulamayı öğrenmemizi sağlar. Uygulama hakkında ön bilgimiz olur.

Kullanılabilirlik Testi: Kullanıcı dostu testi olarak da bilinir. Eğer kullanıcı ara yüzü önemli bir yere sahipse ve ihtiyaçlar belirli bir kullanıcı kitlesine göre belirleniyorsa test başarılı olur.

Duman Testi: Mantık testi olarak da bilinir. Bu test, uygulamanın büyük testlere hazır olup olmadığını belirler ve küçük testleri başarı ile geçtiğini gösterir.

Yenilenme Testi: Uygulamanın herhangi bir hataya karşı ne kadar sürede eski haline geleceğini test eder. Sistem gereksinimlerine göre, tip ve yenileme hızı belirlenir.

Seviye Testi: Uygulamanın etkinliği ile ilgilenir. Uygulama tarafından çok büyük veri işlemi yapılırken, sistemin uç limitlerini kontrol eder.

Kullanıcının Gerekli Olduğu Testler

Kullanıcı Kabul Testi: Kullanıcının sistemi test edip, gereksinimleri karşılayıp karşılamadığının incelenmesini sağlar.

Alfa Testi: Kullanıcı geliştirme merkezine çağrılır. Geliştiriciler programı kullanır ve program hakkında not alır ya da kullanıcılar işlemleri gerçekleştirir.

Beta Testi: Kullanıcılara beta versiyon dağıtılır ve test etmesine izin verilir. Kullanıcılar sistemi inceler, herhangi bir sorun bulduğunda, geliştiriciye bildirir.

Beyaz Kutu Testi

Beyaz kutu testi Uygulamanın kodunu temel almaktadır. Kodun koşullarını, alanlarını ve açıklamalarını temel alır. Beyaz Kutu testi, cam, açık kutu, temiz kutu olarak adlandırılmaktadır. Bu testte, testi yapan kişi sorunlu kısmı bulmak için kodu incelemelidir.

Beyaz Kutu Test Metodları;

Birim Test: Çalışan bir kod parçası ya da modül için, geliştiriciler tarafından gerçekleştirilir. Düşük seviyede işlem gerçekleştirir.

Statik ve Dinamik Analiz: Statik analiz kodu sıralı bir şekilde inceler ve hataları araştırır. Dinamik analiz, kodun çalışmasını ve çıktıyı analiz eder.

Açıklama Kapsamı: Açıklamaların test edilmesiyle ilgilenir. Her bir açıklama en az bir kez test edilir. Tüm açıklamaların sorun yaşamadan çalıştığını garanti altına alır.

Sınıf Kapsamı: Hiçbir yazılım uygulaması sürekli olarak kodlanmaz, bazı noktalarda, kodun dağılışı incelemeli ve belirli bir fonksiyonu çalıştırılmalıdır. Tüm sınıfların doğrulanmasına yardım eder ve uygulamanın anormal davranış göstermesini engeller.

Güvenlik Testi: Sistemin izinsiz erişimler, kod bozulması, k gibi olaylardan nasıl korunacağı ile ilgilenir. Karmaşık test metotları gerekir.

Değişim Testi: Belirli bir hata düzeltildikten sonra yapılan bir testtir. Hangi kodun, stratejinin geliştirmeye yardımcı olacağını belirlemeyi sağlar.

Bazı testler hem Beyaz Kutu testi hem de Kara Kutu testi kapsamında incelenebilir. Bunlara örnek verecek olursak;

Fonksiyonellik Testi: Kodla birlikte fonksiyonelliği test eder.

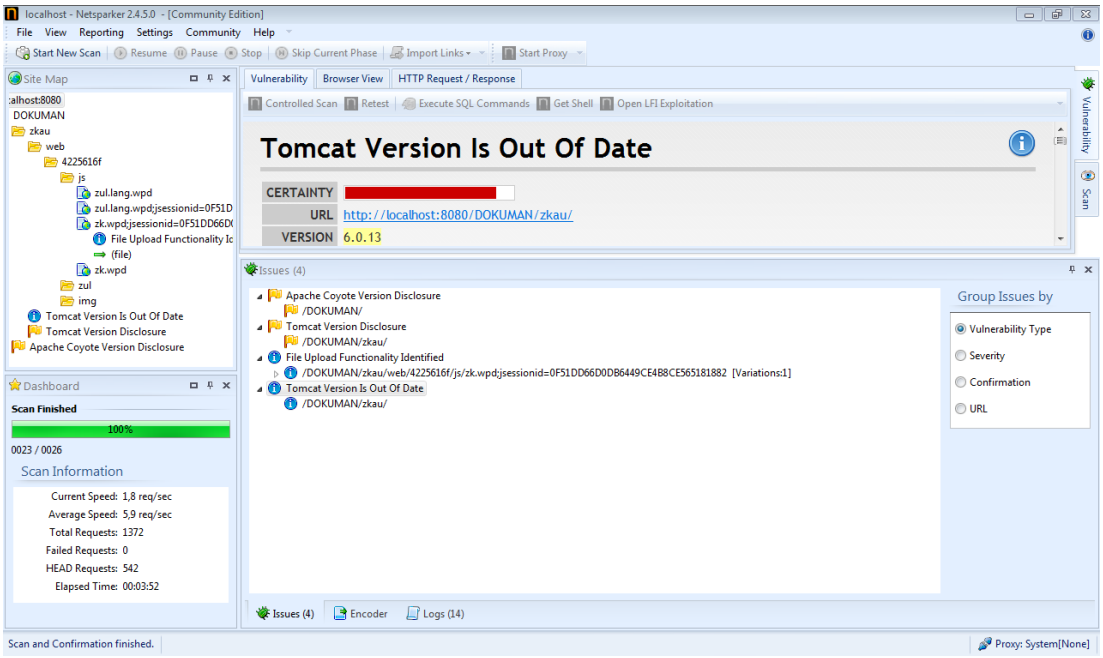
Birleştirme Testi: Uygulamaya yeni eklenen kod ile işbirliği yapar.

Performans ve Yükleme Testi: Kodun hangi kısmının sistemin kaynaklarını yönettiğini ve performansı belirler [32].

2.1.3.2. Uygulamada Kullanılan Test Yazılımları

Uygulamanın ihtiyacı karşılayıp karşılamadığını anlayabilmek için sistemi kullanan kullanıcılar ile sürekli ekran girişleri ile bilgi sunumu yapılmıştır. Çevik metodu kullanıldığından dolayı aslında geliştirme sürecinde sürekli olarak ihtiyaç testi yapılmıştır. Bu yüzden bu aşamada ihtiyaç testleri ele alınmamıştır. Değerlendirmeye alınan testler test performans ve güvenlik testleridir.

Güvenlik testi aşamasında yazılımın kod güvenliği ele alınmıştır. Kodlarda açık olup olmadığını anlamak için 3.parti yazılım kullanılmıştır. Bunun için NetSparker yazılımı kullanılmıştır [33].
NetSparker ile EBYS uygulaması testi yapılmıştır (Şekil 2.22).



Şekil 2.22. NetSparker sonuç ekranı

Şekil 2.22'de çok kritik yazılım eksiklikleri kırmızı renkle, daha az kritik eksiklikler sarı renkle gösterilmektedir. Uygulama için 1 kritik eksiklik bulunmuştur. Zk Framework'e ait 'zkau' dizini saldırı için uygun olabilir. Bu güvenlik açığında kurtulmak için sunucu tarafında 'zkau' dizini ile ilgili işlem yetkisi kaldırılmıştır. Bu yüzden bu kısımda yapılacak saldırılar sonuçsuz kalabilecektir. Sarı ile gösterilen uyarılar Şekil 2.22'de görüntülenmektedir. Uyarılar 'zkau' ile ilgili olduğundan dolayı sorun olmayacaktır.

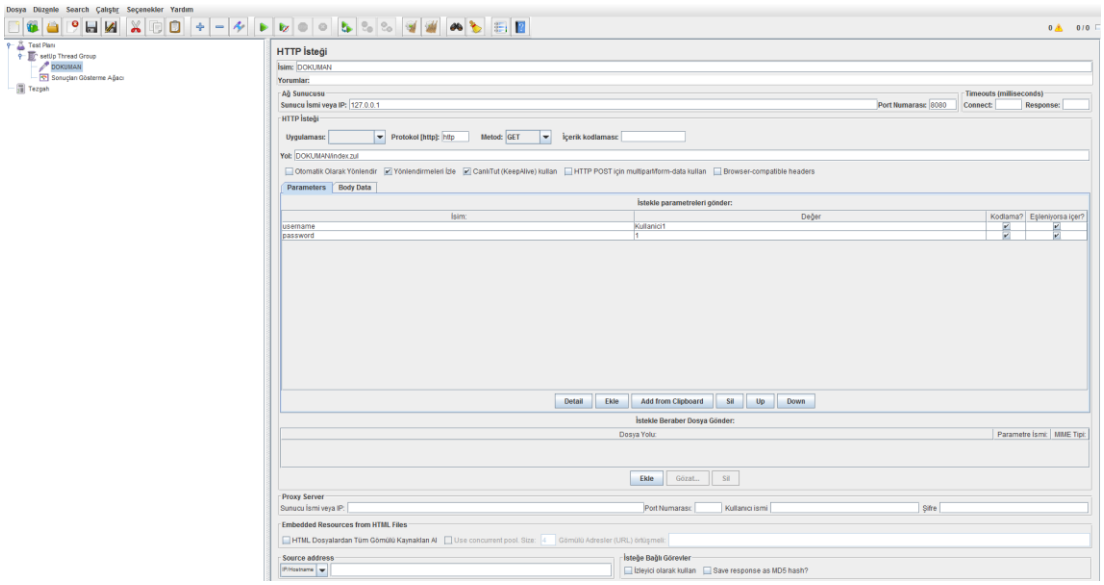
Farklı olarak 2 uyarı bulunmaktadır;

- Uygulama dizin ismi açık olarak kullanılması
- Kullanılan uygulama sunucusunun eski versiyonda olması

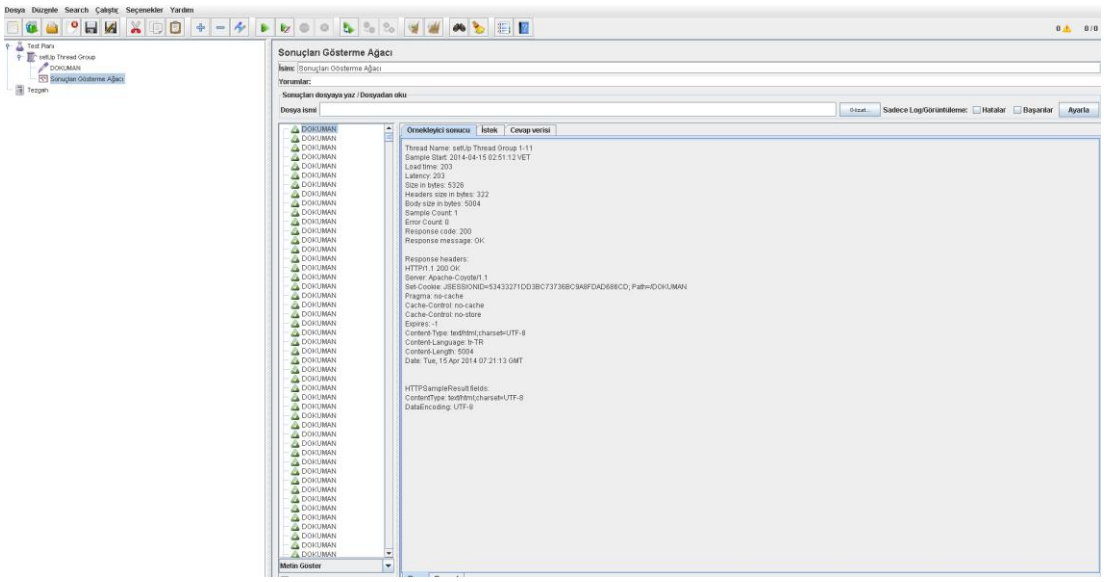
Uygulama dizin ismi geliştirme safhasında kullanıldığından dolayı sorun olmayacaktır. Çünkü uygulama sunucuda çalıştırıldığında dizin ismi kaldırıp belli bir alan adı ile erişimi sağlanacaktır. Uygulama sunucusu geliştirme safhasında iken

versiyonu eski olabilir. Yazılım yayınlandığında, uygulama sunucusu en yeni versiyonu ile kullanılacağından dolayı bu sorun da halledilmiş olacaktır.

Yazılımda stres testi gerçekleştirilmiştir. Stres testi için kullanılan yazılım aracı Apache JMeter'dir. Stres testi gerçekleştirmek için öncelikle uygulamanın web ortamında çalışabilir durumda olması gerekir. Uygulama web adresi için lokal IP adresi veya lokal alan adı, uygulama sunucusu portu ve uygulama dizininin belirlenmesi gerekir. Stres testi için iş parçacıklarının yani Java Thread'lerin tanımlanması gerekir. Apache JMeter'de iş parçacığı sayısı 50 olarak belirlenmiştir. İş parçacıkları tanımlandıktan sonra uygulamanın alan adı veya IP adresi, portu ve uygulama dizin ismi eklenir. Tanımlamalar bittikten sonra test işlemi başlatılır (Şekil 2.23, Şekil 2.24).



Şekil 2.23. JMeter uygulama ayarı için ekran görüntüsü



Şekil 2.24. JMeter test sonuçları görüntüsü

Şekil 2.23'da yazılım için gerekli ayarlar yapılmıştır. Şekil 2.24'de yazılım test sonuçları görüntülenmektedir.

Stres testi sonuçlarına göre yazılım detayları;

- Ekranda yüklenme için geçen süre 0,2 saniye
- Ekran yüklenme boyutu 5326 byte
- Başlıkların boyutu 322 byte
- Ekran ana bileşen boyutu 5004 byte
- Hata sayısı 0
- Ele alınan iş parçacığı sayısı 50
- Yanıt kodu 200

Bu sonuçlara göre yazılım başarılı bir testten geçmiştir.

2.1.4. Kurulum

Uygulama için gerekli 3 tane kurulum vardır. Bunlar; Veri tabanı kurulumu, Yazılım Kurulumu ve Uygulama Sunucusu Kurulumudur.

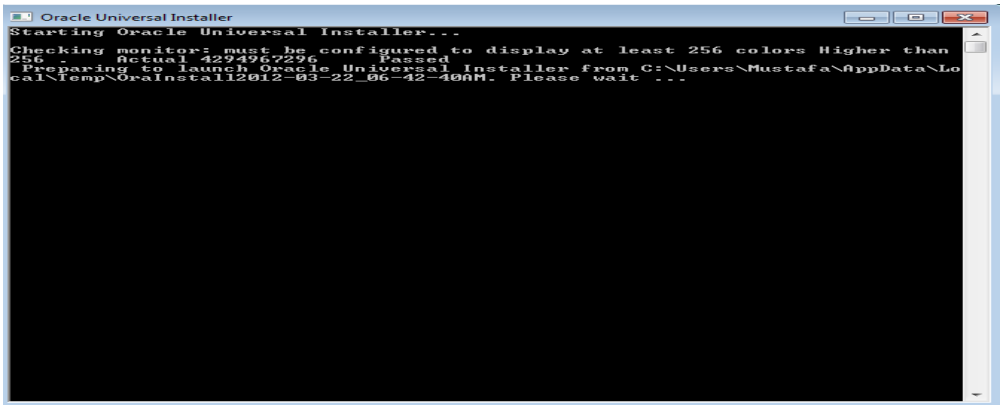
2.1.4.1. Veri tabanı Kurulumu

Elektronik belge yönetim sistemi uygulaması için kullanılan veritabanı Oracle veri tabanıdır. Oracle veri tabanı için farklı sürümler mevcuttur. Uygulama için kullanılan Kişisel Sürümdür.

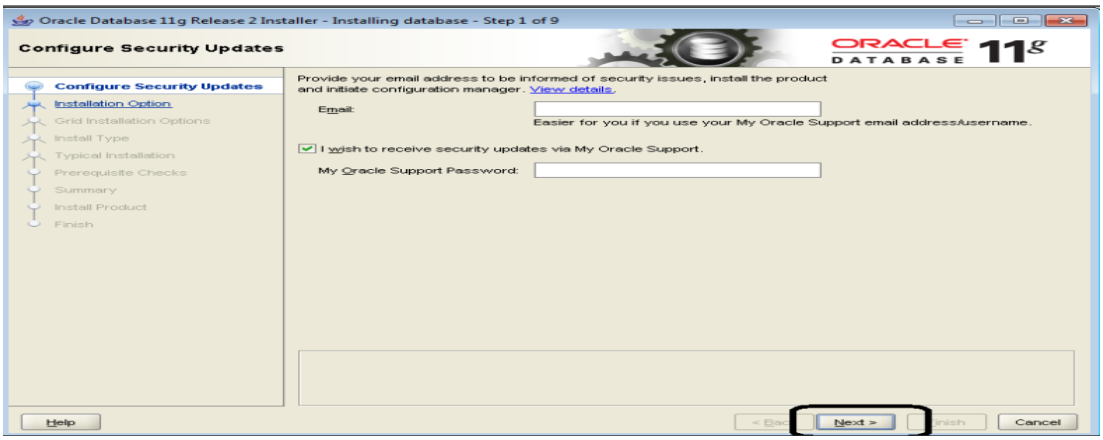
Sürüm çeşitleri;

- Kişisel Sürüm
- Standard Sürüm
- Kurumsal Sürüm

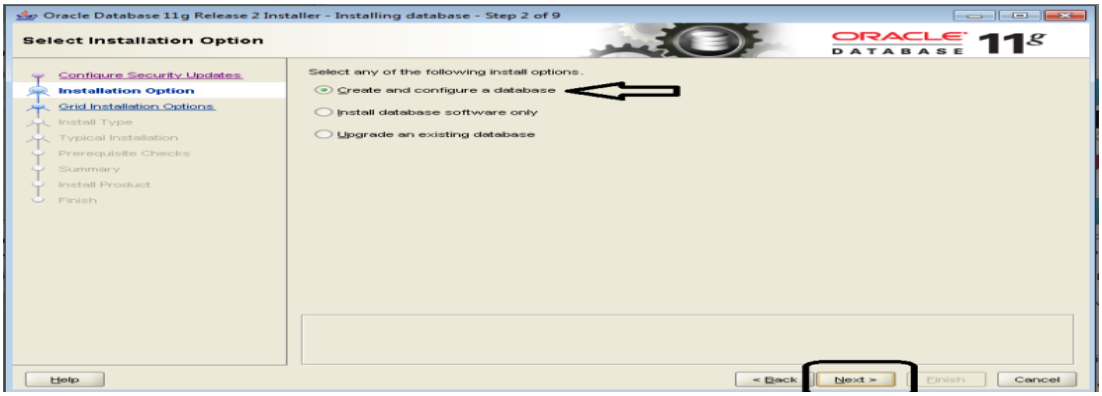
Oracle Veri tabanı sistemi kendi internet web sayfasından indirilebiliyor. Veri tabanı sistemi bilgisayara indirildikten sonra kurulum aşamaları başlatılır. İndirilen dosya üzerinden aşamaları sırasıyla Şekil 2.25- Şekil 2.33 arasında gösterilmektedir.



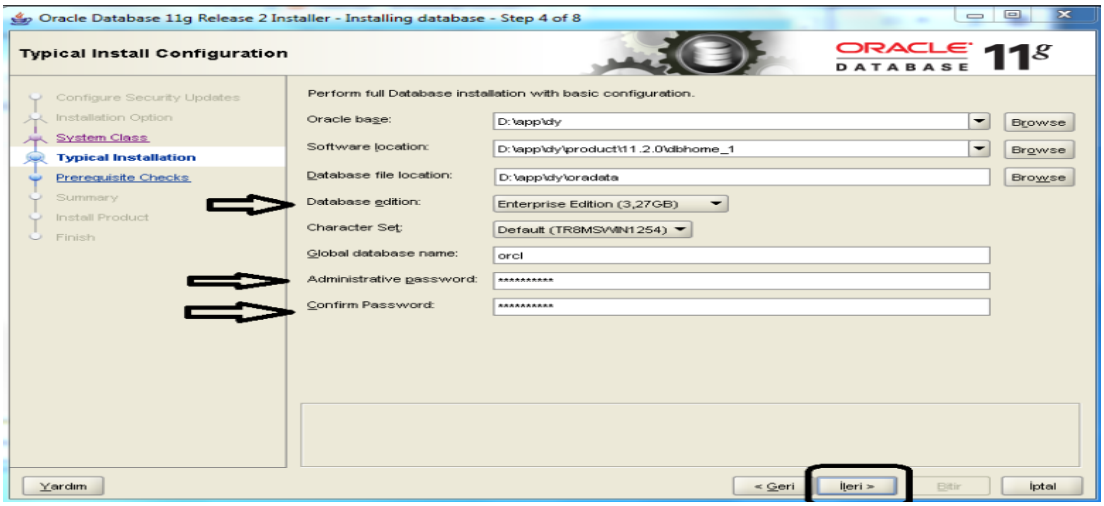
Şekil 2.25. Oracle ilk yüklenme görüntüsü



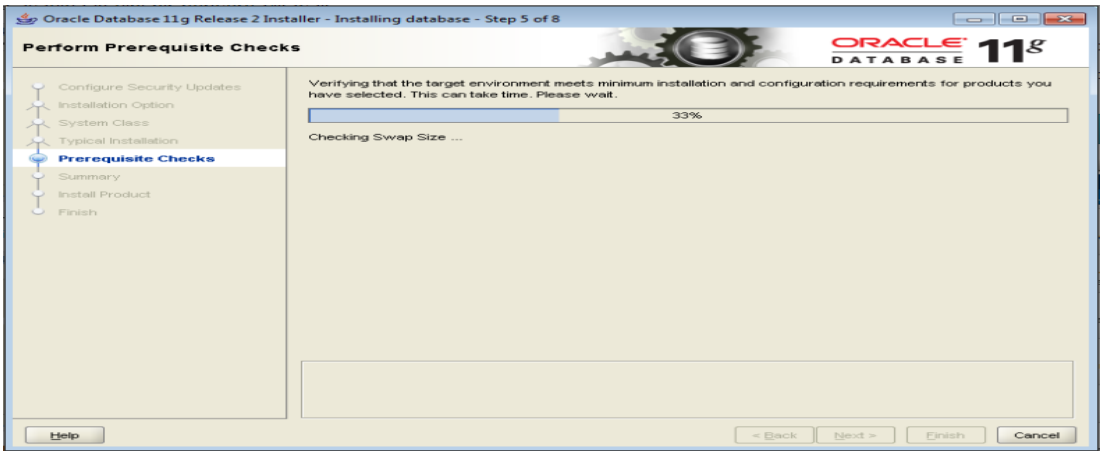
Şekil 2.26. Oracle yüklenme anında üyelik için istenen bilgi ekranı



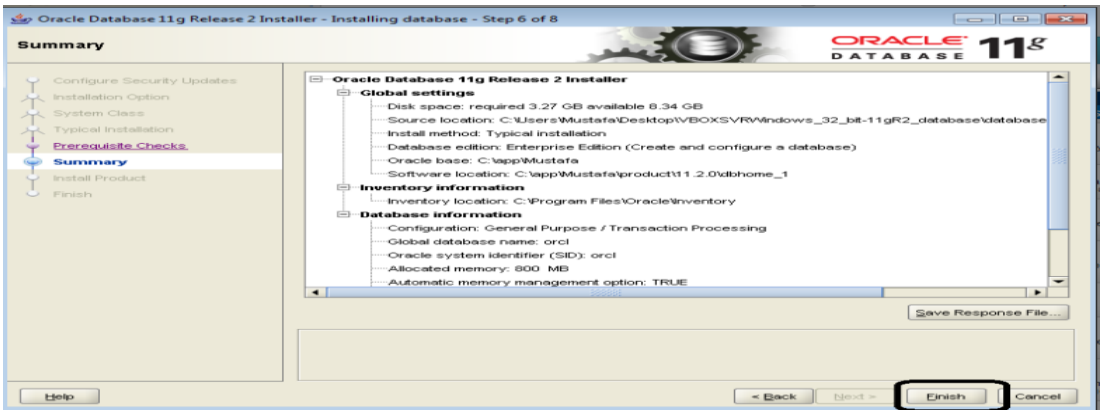
Şekil 2.27. Veritabanı kurulum seçeneği için ekran görüntüsü



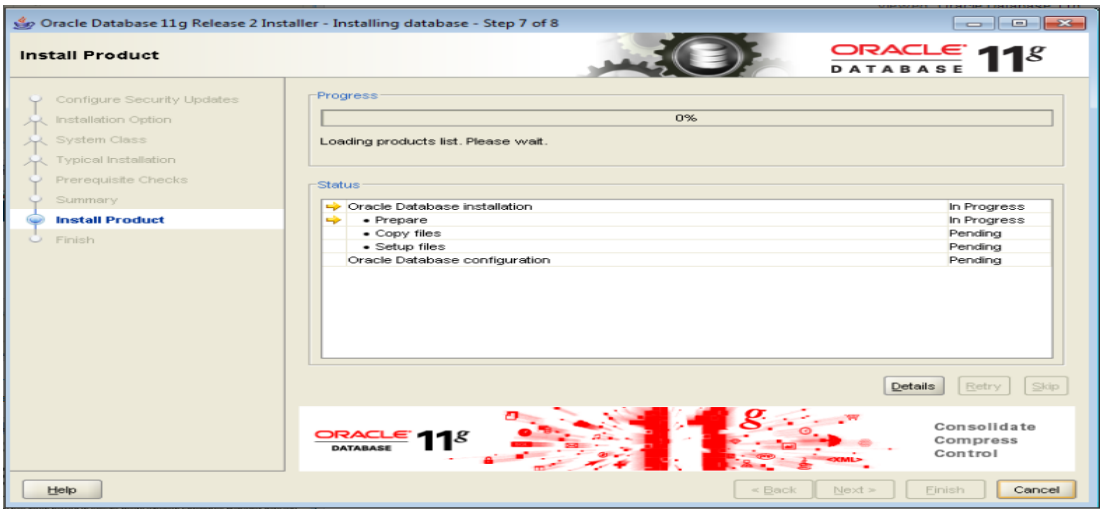
Şekil 2.28. Veritabanı kurulum dizini ve ilgili seçenekler görüntüsü



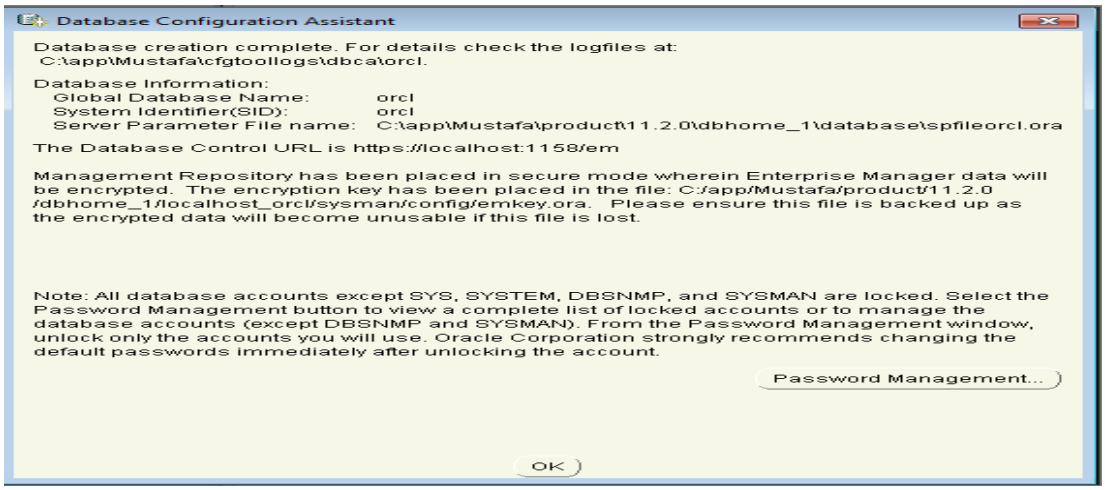
Şekil 2.29. Veritabanı yüklenme anındaki son ayarlama ekran



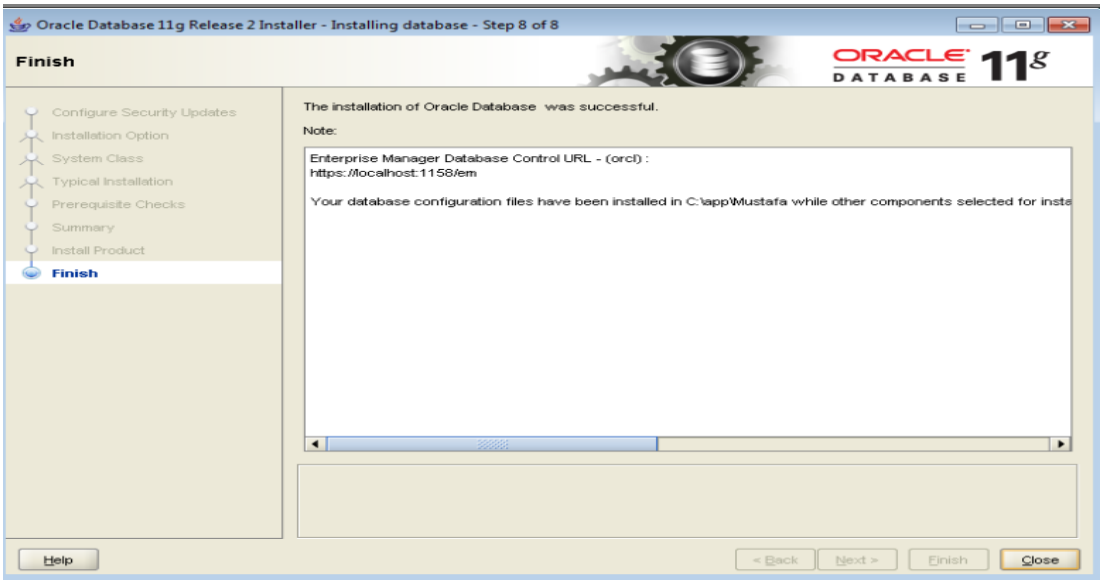
Şekil 2.30. Veritabanı kurulumu öncesi yüklenecek bilgiler için rapor ekranı



Şekil 2.31. Veritabanı kurulum ekranı



Şekil 2.32. Veritabanı kurulumu mesaj ekranı



Şekil 2.33. Veritabanı kurulumu bitiş ekranı

Kurulum aşamasında bazen seçimlik alanlar olabiliyor. Şekil 2.26'da e-posta adresi alanı istemektedir. Bu alan zorunlu değildir. Şekil 2.27'deki gibi ilk seçim yapılır. Bu seçimle birlikte veri tabanı hem kurulacak hem de konfigürasyonu yapılabilecektir. Şekil 2.28'de veri tabanı kurulum dizini seçilir. Alt kısımda veri tabanı sistem

kullanıcısı için şifre belirlenecektir. Geri kalan kısımlarda ilerleme butonuyla kurulum tamamlanmış olacaktır. Veri tabanı kurulumundan sonra yazılım için veri tabanı kullanıcısı tanımlanması gerekir. Tanımlanan veri tabanı kullanıcısı ile giriş yapıldıktan sonra yazılım için tablolar oluşturulacaktır. Tablo tanımları Geliştirme bölümünde açıklanmıştır.

2.1.4.2. Yazılım Kurulumu

Yazılım kurulum kısmı diğer kurulumlara göre daha kolay bir aşamadır. Yazılım kod geliştirmesi tamamlandıktan sonra yayınlanması için tüm kod bileşenleri ve kütüphaneleri ile bilgisayarda kurulum dosyası oluşturulur. Yazılım Java web uygulaması olduğundan dolayı kurulum dosyası war uzantısı şeklinde olur. Elde edilen yazılım kurulum dosyası daha sonra uygulama sunucusunda çalıştırılmak üzere bilgisayar ortamında kaydedilir. War uzantılı dosyalar java web yazılımlarının sıkıştırılmış dosya halidir. Dosyalar sıkıştırılmadan da alınabilir. Ancak yazılımın tek dosya olması açısından war uzantılı dosyalar faydalı olacaktır.

2.1.4.3. Uygulama Sunucusu Kurulumu

Elektronik Belge Yönetim sistemi web tabanlı olduğundan dolayı uygulama sunucusuna ihtiyaç duymaktadır. Uygulama sunucusu yazılımın yayınlanması için gerekli iş parçacıklarını bulundurur. Bu iş parçacıkları sayesinde aynı anda birden fazla istemciye hizmet sağlar. Çeşitli uygulama sunucuları bulunmaktadır. Bu test çalışmasındaki yazılım için tercih edilen uygulama sunucusu Apache Tomcat'tir. Uygulama sunucusu Java web uygulamaları için çalışma ortamı sağlamaktadır. Web tabanlı yazılım kurulum dosyaları sunucuda derlendikten sonra internet ortamında erişime açılır. Erişim için IP adresi veya alan adı kullanılabilir.

Apache Tomcat uygulama sunucusu kendi web sayfasından indirilebilir. Dosya yapısı webapp, bin, conf, lib, logs, temp ve work'tur. Yazılım kurulum dosyası webapp dizininde tutulmaktadır. Yazılımın çalışması için bilgisayarda Java JDK

kütüphanesi yüklü olması gerekmektedir. Ayrıca Java JDK için bilgisayar ayarlarında ortam değişkenlerinde JAVA_HOME dizini tanımlanmalıdır [34].

Uygulama sunucusu kendi dizininde bin klasörü altında 'startup.bat' dosyası ile çalıştırılmaktadır. Sunucu başlatıldıktan sonra yazılım sunucu IP'si veya erişim adresinden hizmet sağlamaya başlayacaktır.

2.2. Uygulama yapısı

Yazılım iskelet yapısını detaylı görmek için diyagramlar hazırlanmıştır. Diyagramlar 2 farklı şekilde ele alınmıştır.

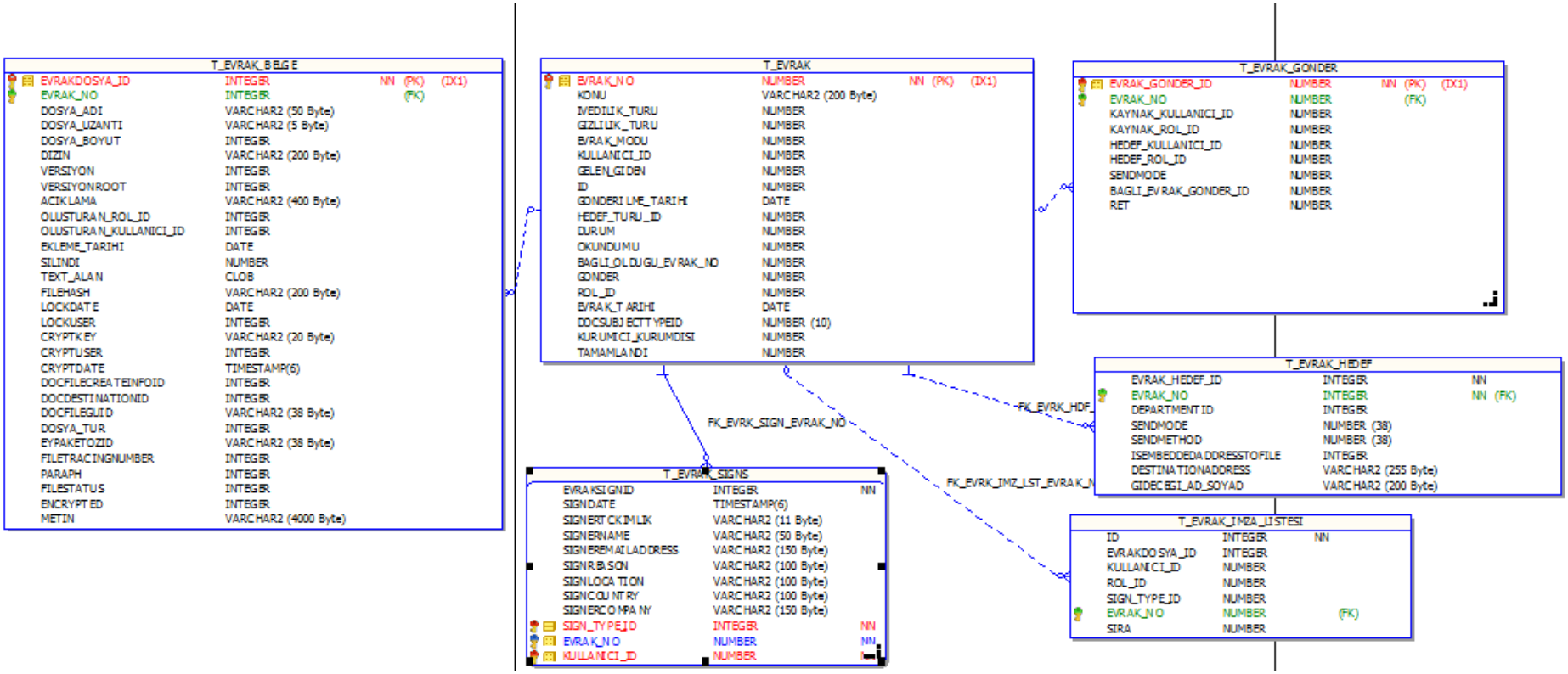
Bunlar;

- ER Diyagram
- UML Diyagram

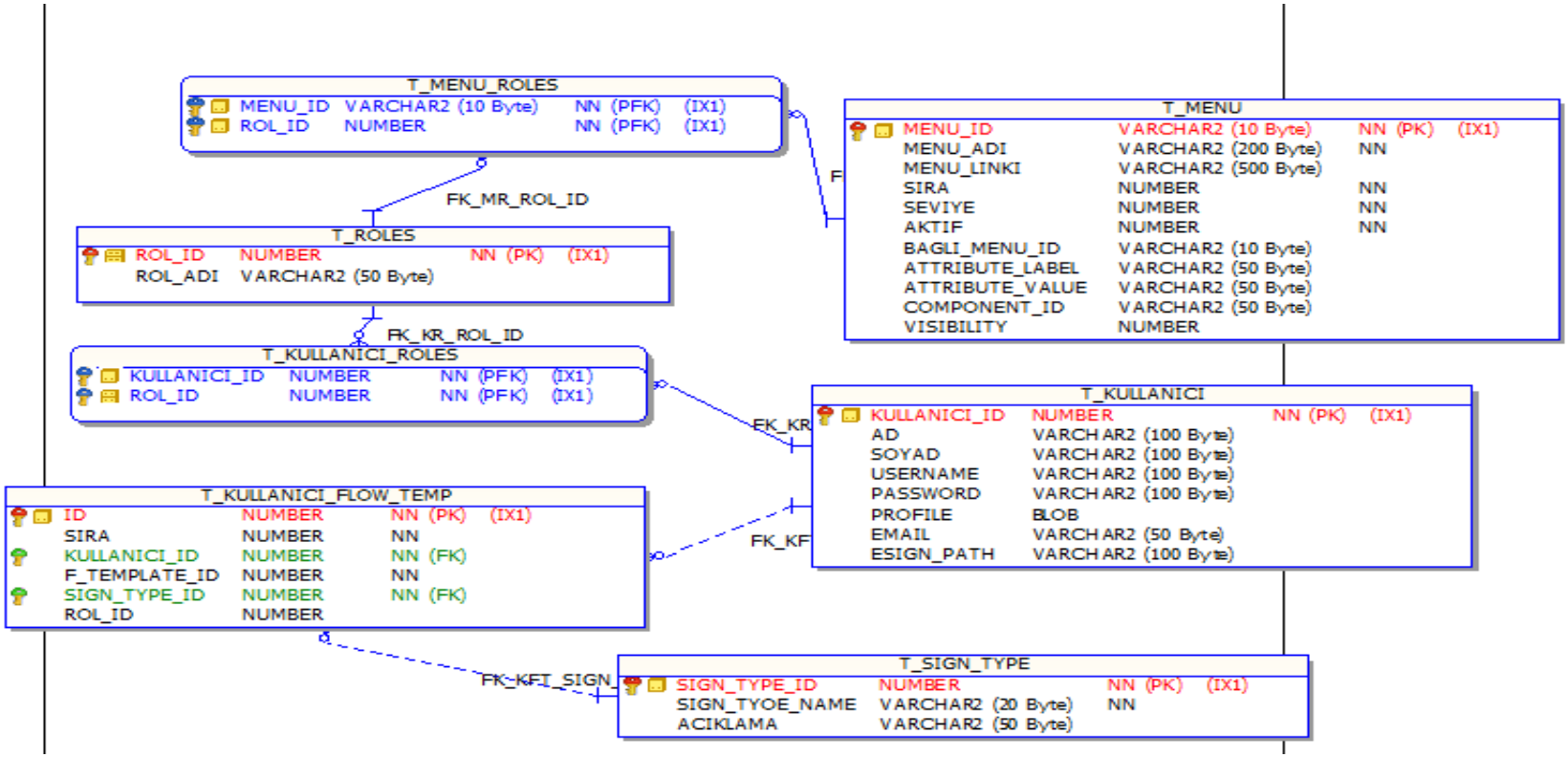
2.2.1. ER Diyagram

Türkçesi varlık-ilişki şemasıdır. Bu şemada, veri tabanı içinde bulunan bütün tablo ve ilişkiler belirtilir. Veri tabanının genel yapısı buradan kolaylıkla anlaşılabilir. Büyük sistemler için oldukça gereklidir. Bu sayede sisteme hâkimiyet sağlanması daha kolaydır. Diyagramlar herhangi bir veri tabanı sistemine özel değildir. Diyagramı hazırlayabilmek için yazılım araçları vardır. Bu sistem için kullanılan araç Toad aracıdır. Toad ile tabloların birincil anahtarı, ilişkili anahtar, indeks gibi özellikleri de gösterilebilmektedir.

EBYS yazılımının veri tabanı sistemini genel olarak özetlemek için 2 tane ER Diyagram oluşturulmuştur. Evrak yapısı için ER diyagram ve Kullanıcı yapısı için ER diyagram. Evrak yapısı için ER diyagram Şekil 2.34'de ve Kullanıcı yapısı için Er diyagram Şekil 2.35'de gösterilmektedir.



Şekil 2.34. Evrak tablosu ER diyagramı



Şekil 2.35. Kullanıcı tablosu ER diyagramı

ER diyagramlarında tablolara ait anahtarların belirlenmesi gerekmektedir. Anahtar eklenmiş tablolardan veriler daha hızlı şekilde elde edilir. Çünkü anahtarlar ayrıca tablolar için indeks yapısını oluşturur. Anahtarlar tablolar üzerinde görüntülenmektedir (Şekil 2.34 ve Şekil 2.35).

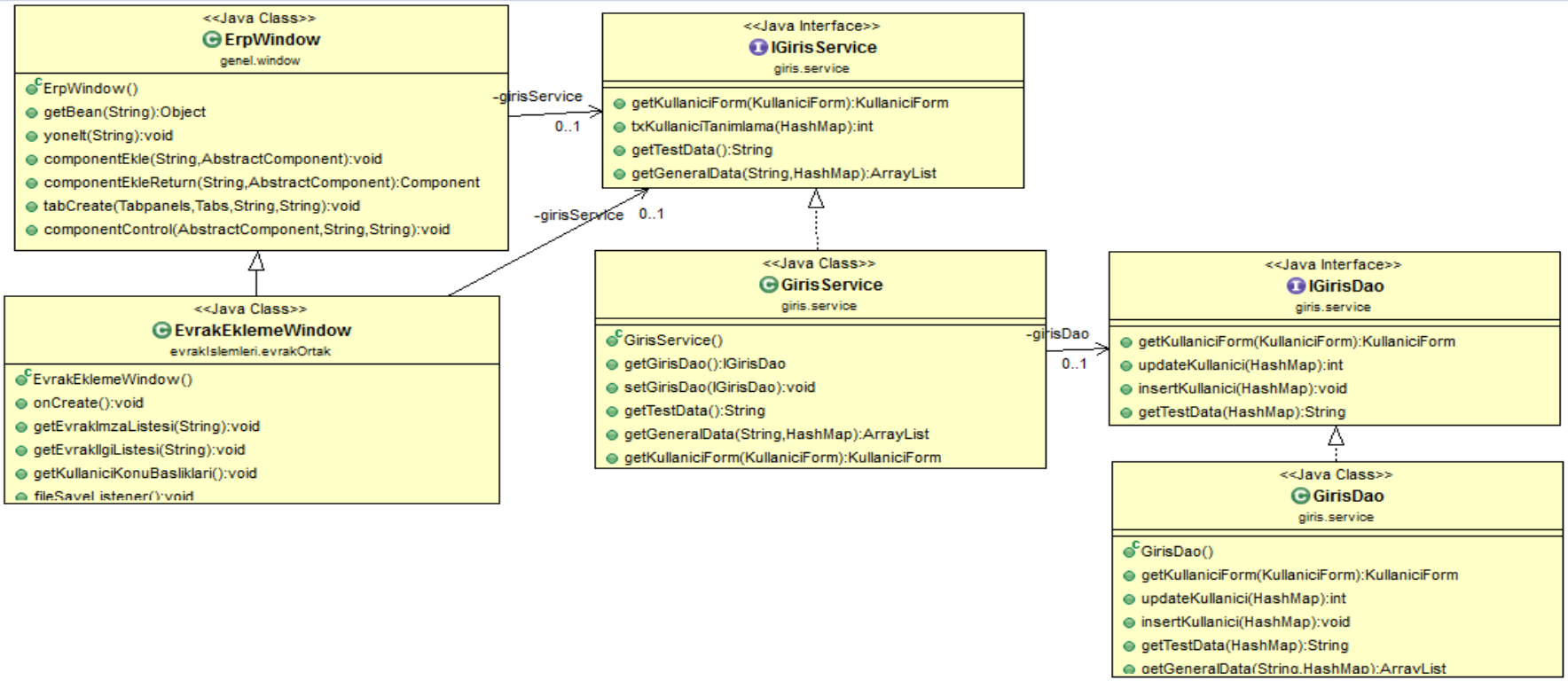
2.2.2. UML Diyagram

UML Türkçe karşılığı olarak "Birleşik Modelleme Dili" şeklinde adlandırılabilir. UML bir programlama (ya da yazılım geliştirme) dili değildir daha çok yazılım geliştiricileri tarafından kullanılır [35].

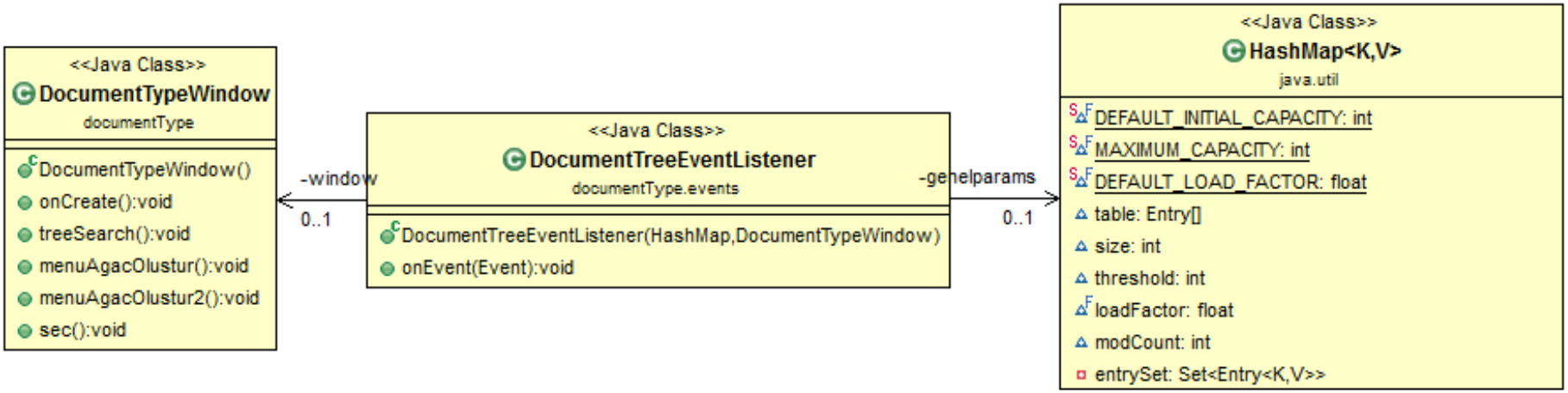
Kullanıcıların veri tabanı ile etkileşiminde kullanılan diyagramların hepsi aynı mantıkta tasarlanmıştır. Bunun için evrak kayıt işlemi örnek olarak ele alınabilir. Veri tabanı işlemlerinde kullanılan sınıflar diyagramda gösterilebilir ve yazılımdaki veri tabanı işlemlerinde benzer diyagramlar oluşturulabilir (Şekil 2.36).

Diğer veri tabanı işlemlerinde diyagramda değişen tek durum Java kontrol sınıfının değişmesidir. ‘EvrakEklemeWindow’ yerine başka kontrol sınıfı düşünülebilir.

Yazılımda açılan ağaç bileşenli ekranlarda her kayıt için bir olay aksiyon tanımlanmıştır. Olay tabanlı diyagramları evrak dosya kodu örneğinde gösterebiliriz. Evrak dosya kodu, ekranda ağaç yapısında açıldığını daha önce ele alınmıştır. Açılan ağaç yapısında kayıtlar seçildiğinde altındaki kayıtlar listelenir. Dolayısıyla kayıtlar için bir olay tanımlanmıştır. Bu yapı için kullanılan UML Sınıf diyagramı Şekil 2.37’da görüntülenmektedir.

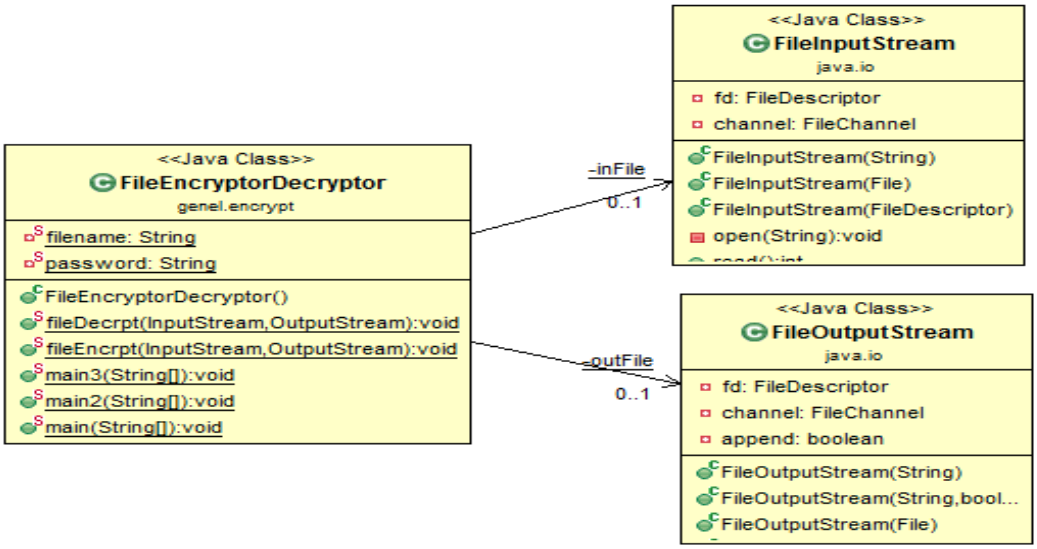


Şekil 2.36. Yazılım genel alt yapısı sınıf diyagramı



Şekil 2.37. Olay tabanlı modeller için sınıf diyagramı

Dosya dizininde kaydedilen evrak kriptolanıyordu. Kripto işlemlerinde kullanılan Java sınıflar Şekil 2.38'de gösterilmiştir.



Şekil 2.38. Evrak belge şifreleme için sınıf diyagramı

Yazılımda gerçekleşen veritabanı işlemleri, olay güdümlü işlemler ve dosya kriptolu işlemleri için diyagram örnekleri verilmiştir.

3. ARAŞTIRMA BULGULARI

Bu çalışmada EBYS yazılımı benzer EBYS sistemlere göre iyileştirilerek son halini almıştır. İyileştirmeler genel olarak güvenlik ve kullanılabilirlik tarafında olmuştur. Bunların yanında yazılımın kendisine ait özgün yapıları da bulunmaktadır.

3.1. Güvenlik Yöntemi Bulguları

Bu yazılımda dosya kriptolama işleminde farklı yöntemler kullanılmıştır. Evrak dosya dizinde kripto hali bulundurulur. Bu kripto işlemi yazılımda kod tarafında gerçekleştirilir. Şifreleme algoritması MD5 ve DES yöntemleridir. İçişleri Bakanlığı Elektronik Belge ve Yönetim Sisteminde kullanılan yöntem bu çalışmadan farklı bir şekilde ele alınmıştır. Orda kullanılan şifreleme algoritmaları HSM cihazları ile sağlanmaktadır. HSM cihazları kendi içerisinde hazır bulunan şifreleme algoritmaları ile kripto işlemini gerçekleştirir. Bu çalışmada kullanılan kripto işlemleri için güvenlik kodu gibi bir şifre ile başlatılır. Bu şifre evrak oluşturan kullanıcı tarafından da elle girişi yapılarak kriptolu hale getirebilir. Bu yöntemle kullanıcı etkisiyle kripto işlemi başlatılmış olacaktır.

Dosya kripto işlemlerinde güvenlik seviyeleri bulunmaktadır. Bu seviyeler normal, 1.derece ve 2.derecedir. Çalışmada kullanılan kripto düzeyi 2.seviyededir. Bazı EBYS sistemlerinde normal düzey kullanılır. Örnek olarak Devlet Meteoroloji İşleri Genel Müdürlüğü'nde kullanılan EBYS sisteminde belge güvenliği normal düzeydedir [36,37].

Çeşitli şifreleme algoritmaları bulunmaktadır. Şifreleme algoritmaları 2 ana sınıfta gruplanır.

Bunlar;

- Simetrik Şifreleme Algoritması
- Asimetrik Şifreleme Algoritması

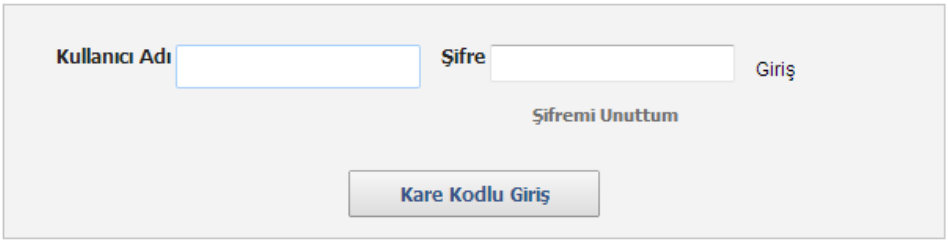
Simetrik Şifreleme Algoritması: Veriyi şifrelemek ve şifreli veriyi çözmek için her iki tarafında bildiği ortak-tek bir anahtar kullanılır.

Asimetrik Şifreleme Algoritması: Veriyi şifrelemek ve şifreli veriyi çözmek için 2 farklı anahtar kullanılmaktadır. Bunlar public key (açık anahtar) ve private key (gizli-kapali anahtar)'lerdir. Asimetrik şifreleme algoritmalarında veriyi alan taraf sadece kendisinin bildiği bir private key ve diğer kişiler – kurumlarca bilinebilen bir public key oluşturur. Gönderen veriyi, alıcının public key'i ile şifreler. Alıcıya ulaşan şifreli veri, alıcının private key'i kullanılarak çözülür. Private key bilinmediği sürece şifrelenmiş verinin hiçbir değeri yoktur. Çünkü belirli bir kişinin-kurumun public key'i ile şifrelenmiş veri sadece o kişinin-kurumun private key'i ile çözülebilir [38].

DES ve MD5 algoritmaları simetrik şifreleme algoritmaları grubuna girer. Yazılımda bu algoritmalar kullanılmıştır. Yazılımda kullanılan yöntem farklı olarak şifre bilgisi kod kısmında tutulduğundan evrakı kullanacak kullanıcıların şifreyi bilmelerine gerek kalmayacaktır.

Yazılım saldırıları genellikle kullanıcı giriş ekranları üzerinde gerçekleşir. Bu güvenlik sorununu halletmek için yazılımlarda çeşitli güvenlik yöntemleri ele alınmıştır. Yöntemlerden bir tanesi kullanıcı adı ve şifre dışında ayrıca ekranda değişken değerli bir kod alanı eklenmesidir. Bu yöntemle robot saldırılarının önüne geçilmiştir. Çünkü kod alanı yanlış girildikçe ekranda farklı bir kod üretilir. Bu yöntemden farklı olarak yazılımla giriş güvenliğini sağlamak için yöntem ele alınmıştır; Kare kodlu giriş ekranı.

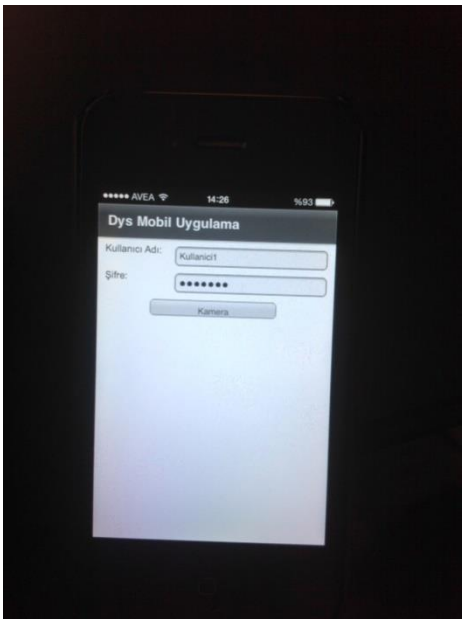
Kare kodlu giriş ekranı ile ilgili tanım yapılmıştı. Bu yöntemle ekranda kullanıcı adı ve şifre girilmez. Bunun için mobil cihazlara yazılıma ait mobil uygulama programı yüklenmesi gerekmektedir. Mobil uygulama ekranında kullanıcı adı ve şifre kaydedilir. Bundan sonra yapılacak tek işlem web uygulama ekranında kare kodlu giriş seçeneği ile kare kod ekranının açılmasıdır (Şekil 3.1, Şekil 3.2).



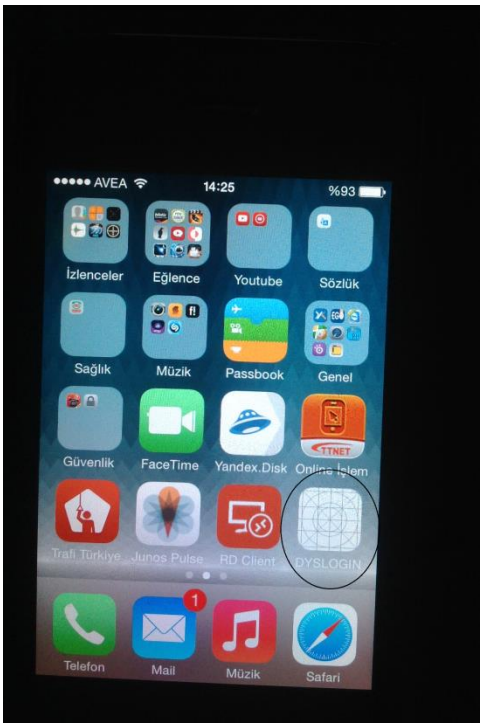
Şekil 3.1. Kare kod seçenekli giriş ekranı

Şekil 3.2. Kare kod örneği

Kare kod ekranı açıldıktan sonra mobil uygulama kamerasıyla kare kod okutularak sisteme giriş yapılabilir (Şekil 3.3, Şekil 3.4).



Şekil 3.3. Mobil uygulama EBYS giriş ekranı



Şekil 3.4. Mobil uygulama ikon sembolü

Kare kod doğrulama seçeneği kullanıcı giriş ekranlarında bu yöntem henüz kullanılmıştır. Türkiye İş Bankası tarafından alış veriş siteleri için uygulanmıştır. Bankanın mobil uygulamasına kredi kartı bilgileri kaydedildikten sonra bankayla anlaşmalı alış veriş siteleri satış ekranlarında kare kod üretilir. Üretilen kod İş Bankası mobil uygulama ile okutulur ve işlem gerçekleştirilir [39].

Web uygulamalarında kare kodun kullanıcı girişi için ilk kez kullanılması yazılıma özgünlük kazandırmıştır. Bu yöntemle EBYS sistemi diğer EBYS sistemlerine göre güvenliği daha da artırmıştır.

3.2. Kullanılabilirlik Yöntemi Bulguları

Yazılımların kullanıcılar tarafından kısa sürede benimsenmesi gerekir. Bunun en etkili yöntemi yazılımların kullanıcı dostu şeklinde tasarlanmasıdır. Kullanıcı dostu ekranlar nasıl tasarlanır?

Bu sorunun cevabını alt başlıklar halinde incelenebilir. Ekran tasarımları için farklı yaklaşımlar ele alınmıştır [40].

Temel yaklaşımlar;

- Kontrol Edilebilirlik
- Öğrenile bilirlik
- Yardım Edilebilirlik
- Görünürlük
- Etkinlik
- Memnuniyet

Bu yaklaşımlar ile birlikte sonuçlara Anketler ve Kullanılabilirlik testi ile ulaşılmıştır.

Anketler, genellikle kullanıcıların web sayfaları hakkında görüşlerini almak için kullanılmaktadır. Günümüzde, anket tasarımı başlı başına bir çalışma konusudur.

Anketlerde, sonuçlarının güvenilir olması için geçerliliği kanıtlanmış soruların kullanılması gerekmektedir. Literatürde kullanılan kullanıcı memnuniyeti ile kullanıcı görüşlerini ortaya çıkaran birçok anket bulunmaktadır. Geliştirme kısmında Yükseköğretim Kurulu EBYS sistemi ile ilgili anket durumu sunulmuştur.

Kullanılabilirlik testi, gerçek kullanıcılar ve gerçek görevler ile yapılan ve ara yüzlerin değerlendirilmesinde kullanılan en yaygın ve en temel kullanılabilirlik mühendisliği metodudur. Bu metodun amacı, ara yüz ile kullanıcı arasındaki etkileşimi incelemek ve ürünün kullanılmasında engel oluşturan kullanılabilirlik problemlerini tespit etmektir. Kullanıcı ara yüz arasındaki etkileşim gerçek ortamda yapılan gözlem ve ölçümlerle incelenir. Bu çalışmada Çevik Metot yazılım geliştirme sürecinde sürekli olarak kullanılabilirlik testi sağlamıştır.

Gazi üniversitesinde yapılan bir araştırma sonucunu ele alalım. Üniversitenin Merkez Kütüphane web sitesi daha önce yapılan bir çalışmada kullanılabilirlik testi ile değerlendirilmiş ve yaşanan kullanılabilirlik problemlerinin çözümü ile yeni bir tasarım geliştirilmiştir. Bu çalışmada ise;

Anket yöntemi ile;

- Tasarım Değişiklikleri
  Mevcut tasarım ile yeni tasarımın karşılaştırılması
- Kullanıcı Memnuniyeti
  Mevcut tasarım ile yeni tasarımın kullanıcı memnuniyetinin ölçülmesi

Ankette kullanılan sorular;

- Bilgi organizasyonun anlaşılır olması
- Kolay kontrol edebilme
- Kolay öğrenme
- Yardım mesajlarının yeterli olması
- Site kapasitenin yeterli olması
- Hataların kolay ve hızlı düzeltilmesi
- Görsel tasarımdan memnun kalma
- Yazı karakterlerinin okumayı kolaylaştırması

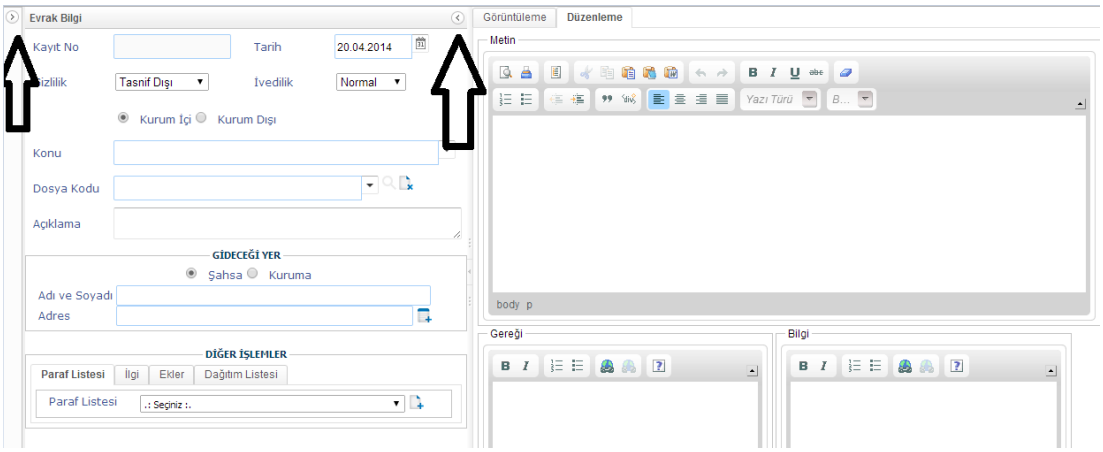
- Renklerin okumayı kolaylaştırması
- Görevlerin etkili bir şekilde yerine getirilmesi
- Birbirini takip eden sayfaların anlaşılması
- Bilgiye kısa sürede ulaşım
- Sitenin kullanımdan memnun olma

Anket sorularında kullanılan cevaplar;

- Kesinlikle katılıyorum
- Katılıyorum
- Kararsızım
- Katılmıyorum
- Kesinlikle katılmıyorum

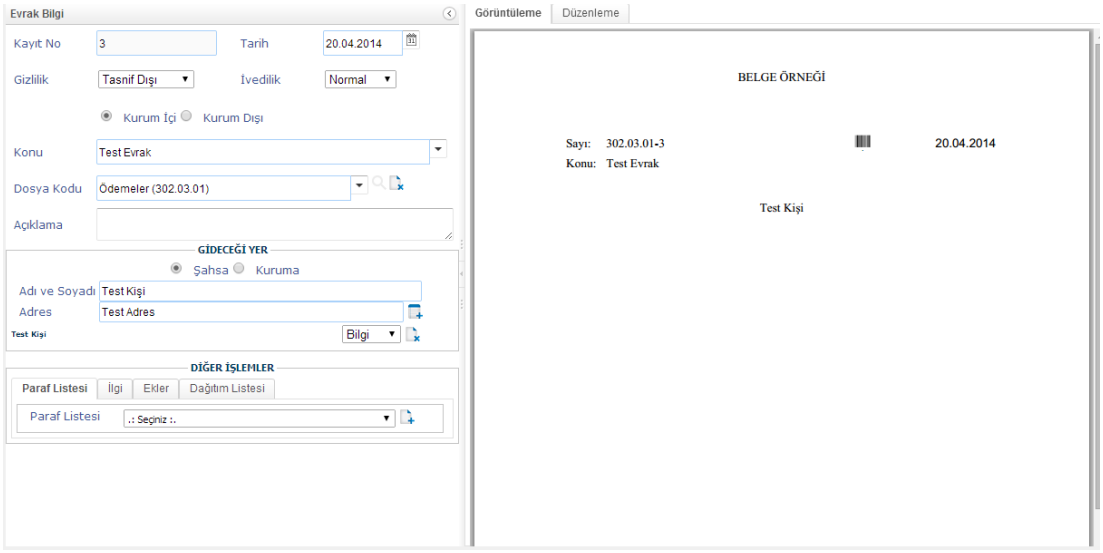
Bu araştırma ile elde edilen sonuçla tasarımlar için daha etkili bir hale gelmiştir. Kullanılabilirlik ile ilgili araştırma bulguları dikkate alınarak EBYS sistemi tasarlanmıştır. Çünkü bugüne kadar piyasada bulunan EBYS sistemlerindeki temel sorun kullanılabilirlik sorunudur. Adalet Bakanlığı ve Havelsan Kurumları tarafından geliştirilen EBYS sistemleri 3 yıl boyunca kullanıcı direnci sorunu yaşanmıştır. Bu tür sorunlar ile karşılaşılmaması için tasarım üzerindeki araştırmalar dikkate alınmıştır.

Yazılım çalışmasında ekran kullanılabilirliğini artırabilmek için ekranların küçültülüp büyütülmesi, bilgi girişi yapılırken evrak belge halinin görüntülenmesi gibi durumlar kullanışlılığı artırmıştır. Mevcut işlemde ihtiyaç duyulmayan ekranlar küçültülerek ekran sadeliğini sağlanmıştır (Şekil 3.5, Şekil 3.6).



Şekil 3.5. Ekran boyutlandırma çubukları

Ok işaretleri ile gösterilen yönler ekranların küçültülüp büyütülmesini sağlamaktadır.



Şekil 3.6. Evrak bilgi giriş ve belge ekranı

Şekil 3.6'da evrak belge halinin görüntülenmesi sağlanmıştır. Turmap firması önerisi ve Yükseköğretim Kurulu kullanıcıların yaşadığı problemler dikkate alınarak bu sorunlar üzerinde durulmuş ve yazılımda bu sorunlara çözümler sunulmuştur.

3.3. Algoritma Yöntemi Bulguları

EBYS alanında hizmet veren yazılımların kod yapısında genel olarak model yapısı bulunmaktadır. Model yapısı geliştiriciler tarafından Java veya başka programa dilleri ile tanımlanan nesnelerdir. MVC olarak tanımlanan iş çatılarında M harfinin temsil ettiği tanımdır. Bu nesneler tamamen yazılıma özeldir. Yani her geliştirici kendi yazdığı koda göre şekillendirmektedir. Sonuçta Modeller elle tanımlandığı için dezavantajlar oluşturmaktadır. Tanımlanan Model yapılarının dezavantajı yazılım esnekliğini azaltma ve değişiklik durumlarında koda müdahale durumunu artırmaktadır. Geliştiriciler tarafından tanımlanan modeller şu an bu alanda çalışan EBYS yazılımlarında kullanılmaktadır.

Üzerinde araştırma yapılan EBYS sistemleri genel olarak;

- Yükseköğretim Kurulu EBYS
- Dış İşleri Bakanlığı EBYS
- Adalet Bakanlığı EBYS
- Havelsan EBYS
- İçişleri Bakanlığı EBYS
- DMİ Genel Müdürlüğü EBYS
- İlgili diğer Lisansüstü tez araştırmaları

Bu EBYS sistemlerinde geliştiriciler tarafından kodlanan model yapıları bulunmaktadır. Bu çalışmada model yapıları tamamen Java kütüphanesine ait HashMap nesnesi ile sağlanmıştır. Bu nesne yazılımda model görevini görmektedir. HashMap nesnesinin en büyük avantajı model yapısının dinamik bir yapıya sahip olmasıdır. Veri tabanında nesne bilgileri elde edildiğinde veya veriler veri tabanına kaydedildiğinde, HashMap nesnesi taşıyabilecek veriler kadar bellekte alan

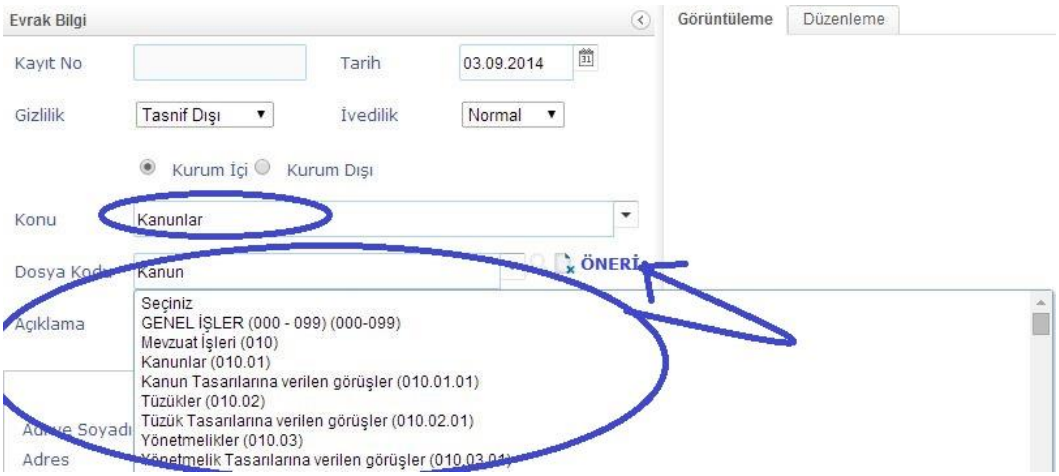
ayırmaktadır. Geliştirilen kodlarda esneklik sağladığından kod değişikliğini azaltmaktadır.

Model-View-Controller (MVC), kullanıcıya yüklü miktarda verinin sunulduğu karmaşık uygulamalarda veri ve gösterimin soyutlanması esasına dayanır. Böylece veriler ve kullanıcı ara yüzü birbirini etkilemeden düzenlenebilir. MVC, bunu controller adı verilen ara bileşenle, veri gösterimi ve kullanıcı etkileşiminden, veri erişimi ve iş mantığını çıkarma suretiyle çözmektedir. Uygulamada bu yapı esas alınarak sistem tasarımı yapılmıştır [41].

3.4. Veri Yönetimi Bulguları

Bu çalışmada yazılım için veri madenciliği kullanılmıştır. Veri madenciliği algoritmalarından Weka MNB yöntemi kullanıldığından bahsedilmiştir. Doküman yönetimi konusunda yapılan bir çalışmada Weka metin sınıflandırma yöntemi daha önce Çin'de doküman tasnifi için kullanılmıştır. Bu çalışmada farklı olarak Weka yöntemi Elektronik Belge Yönetim sisteminde öneriler için kullanılmıştır [42]. Weka metin sınıflandırma yönteminin uygulamalarda sağladığı faydaları bulunmaktadır. Bunlar genel olarak; optimizasyon, performans yönetimi, veri analizi ve yönetimi, zaman tasarrufu gibi faydalardır [43,44].

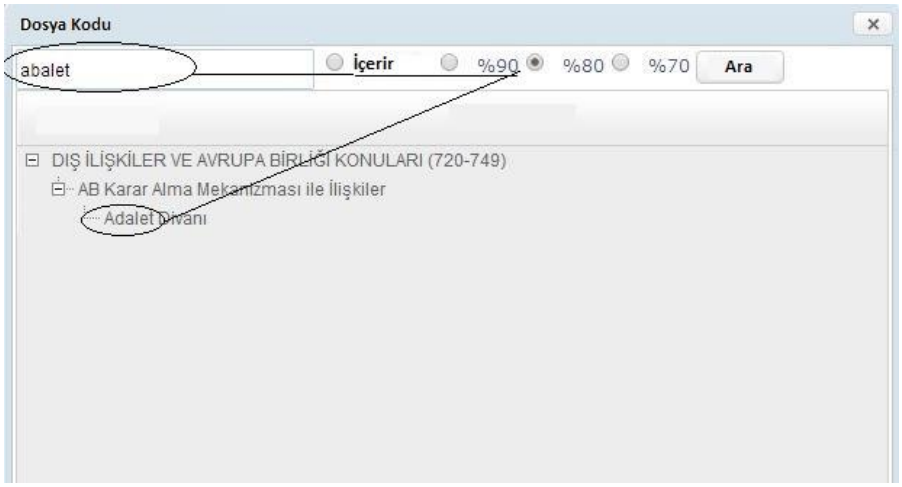
Yazılımda MNB yönteminin kullanımı evrak kayıt işlemlerindeki kullanımı gösterilmiştir (Şekil 3.7).



Şekil 3.7. Evrak kayıt ekranında öneri için Weka MNB yönteminin kullanımı

Şekil 3.7'de gösterilen "Proposal" butonu ile evrak için dosya kodları önermiştir. Bu durumda kullanıcı ilgili bilgileri daha hızlı bir şekilde elde etmiş olacaktır.

Yazılımda Levenshtein Distance yönteminin kullanımından bahsettik. Bu yöntem çalışmada arama kısımlarında kullanılmıştır. Şekil 10 gösterilen arama ikonlarına tıklandığında küçük arama ekranları açılacaktır. Bu ekranlarda Levenshtein Distance yöntemi kullanılmıştır (Şekil 3.8).



Şekil 3.8. Arama ekranında Levenshtein Distance arama benzerlik kullanımı

Şekilde kayıtlar içerisinde geçen metin ifadeleri aramada listelenmektedir. Kayıtlar hiç bulunmazsa kullanıcı benzerlik seçenekleri seçerek ekranda arayacağı kritere göre benzerlik yüzdelerini seçebilir. Sonuçta Levenshtein Distance yöntemi kullanıcılara kayıt aramalarında yardımcı olabilmektedir.

4. SONUÇ VE TARTIŞMA

Bu çalışmada Web tabanlı Elektronik Belge Yönetim Sistemi yazılım geliştirmesini ele alınmıştır. Yazılım geliştirme sürecine uygun olarak yazılım gelişimi gerçekleştirilmiştir. Yazılıma benzer işleve sahip daha önceki yazılımlarında yaşanan sorunlar bu çalışmada ele alınmış ve daha sorunsuz bir ürün oluşturulmuştur. Yazılıma benzer işleve sahip diğer yazılımları kullanan kullanıcıların tecrübelerinden faydalanılmıştır. Tecrübeli kullanıcılar sistem tasarımında etkin bir rol almıştır. Ele alınan ihtiyaçlar ekranlarda en sade haliyle sunulmuştur. Sunulan verilerin yönetimi güvenlikli bir yapıyla sağlanmıştır. Yazılımda ele alınan yöntemler çalışmaya özgünlük sağlamıştır.

Yazılımda ekranlar işlevsel hale getirilmiş ve özellikle karmaşıklığa neden olan detaylar daha sade sunulmuştur. Ekran geçişleri bu duruma örnek verilebilir. Ekran geçişleri yazılımda kullanılmamıştır. Ekran geçişleri yazılım kullanım kolaylığını engellemektedir. Dolayısıyla bu süreçte kullanıcı direnci daha az olabilmektedir. Sayıştay verilerine göre EBYS sistemleri için kullanıcı direnci çok fazla yaşanmaktadır [45]. Ayrıca ekranlarda evrak için belge görüntülenmesi için ayrı bir ekran geçişinde sunulmamaktadır. Mevcut ekranda girilen bilgilere karşılık belge hali görüntülenebilmiştir. Ergonomik tasarım kısmında detaylı olarak ele alınmıştır.

Yazılımda kullanılan teknolojilere bağlı olarak esneklik sağlanmıştır. Özellikle dinamik model yapısının kullanılması yazılım değişikliklerinde etkili olabilmiştir. Ayrıca yazılım dış entegrasyonlara da esnektir. Başka sistemlerinden Web Servis ile veri alış verişi sağlayabilmiştir. Örneğin mobil kare kod uygulaması web servis ile yazılımda kullanılan veritabanına bağlanabilmiştir.

EBYS sistemlerinde güvenlik unsuru temel gereksinimler arasında bulunmaktadır. Sistemde işlem gören evraklara ait bilgiler güvenli bir yapı ile sağlanması gerekir. Evraklar yazılımda belge haline dönüşebilmektedir. Belgeler açık bir dosya halinde tutulması güvenlik açığını oluşturmaktadır. Ayrıca sistemlerde başka güvenlik açıkları da oluşabilir. Kullanıcı bilgilerinin başka bir dış kullanıcı tarafından ele

geçirilip sisteme giriş yapması gibi sorunlar yaşanabilir. Bu çalışmada diğer güvenlik açıkları yanında ayrıca bu tür güvenlik açıklarına da çözüm sunulmuştur. İlk geliştirilen EBYS sistemlerinde güvenlik konusu daha az kapsamlı ele alınmıştı.
Bu çalışmada kullanılan teknolojiler günümüzde en çok tercih edilen teknolojileri arasındadır. Yazılım kurumsal halde tasarlanmıştır. Bu tasarım işlemi Spring Framework ile sağlanmıştır [46]. Spring iş çatısının özellikle tercih edilmesinin nedeni yazılım için kurumsal bir yapı sağlayarak daha işlevsel bir yapı oluşturmasıdır.

Bu çalışmada veri madenciliği kullanılmış ve bulanık arama yöntemiyle farklı yapılar eklenmiştir. Ele alınan algoritmalardan Weka MNB ve Bulanık arama yöntemleri yazılımda kullanılmış olup etkili veri yönetimine katkı sağlamış ve doküman yönetim sistemi çalışmasında öğrenme algoritmaları kullanılması sağlanmıştır.

Karabük Üniversitesinde ISITES 2014 uluslar arası bilimsel etkinliğinde bu çalışmanın sunumu gerçekleştirilmiştir. Sunumda genel olarak çalışmanın faydaları ve yeniliklerinden bahsedilmiştir [47].

KAYNAKÇA

[1] Fırat Üniversitesi Elektronik Belge Yönetim Sistemi,

https://ebysyardim.firat.edu.tr (Erişim Tarihi: 12.02.2014).

[2] Mohammada, M. N., Sulaimana, N., Muhsin, O.A., A novel intrusion detection system by using intelligent data mining in weka environment, 1237–1242. Procedia Computer Science, Selangor, Malaysia, 2011.

[3] Zhong, X., The Research And Application of Web Log Mining Based on the Platform Weka, 4073–4078. School of Computer Science, Jiaying University, Meizhou, Guangdong, China, 2011.

[4] Yadava, A.K., Malik, H., Chandela, S.S., Selection of most relevant input parameters using WEKA for artificial neural network based solar radiation prediction models, 509–519. Renewable and Sustainable Energy Reviews, Delhi, India, 2014.

[5] Haldar, R., Mukhopadhyay, D., Levenshtein Distance Technique in Dictionary Lookup Methods, An Improved Approach, Calcutta, India, 2011.

[6] Kao, C.H., Liu, S.T., Development of a Document Management System for Private Cloud Environment, 424–429. Procedia - Social and Behavioral Sciences, Taipei, Taiwan, 2013.

[7] Jones, S., eGovernment Document Management System: A case analysis of risk and reward, 396–400. Information Technology Department, Conwy County Borough Council, UK, 2012.

[8] Acm Software, Çevik Yazılım Geliştirme "Agile", http://www.acm-software.com/Pdf/AboutAgile.pdf (Erişim Tarihi: 15.02.2014).

[9] Akgün, AKGÜN Doküman Yönetim Sistemi,

http://www.akgunyazilim.com.tr/akgun-dokuman-yonetim-sistemi-icin-is-zekasi/ (Erişim tarihi: 01.03.2014).

[10] *Yükseköğretim Kurulu Ebys, Ankara, 2013.*

[11] Esen, M., *Yükseköğretim Kurulu Kullanıcı Anketi, Ankara, 2013.*

[12] Shneiderman, B., Plaisant, C., Designing the user interface 4 th edition, Pearson Addison Wesley, USA, 2005.

[13] Turmap Yazılım.

[14] *Resmi Yazışma Kuralları,* Başbakanlık, Ankara ,2010.

[15] Zk Live Demo,

http://www.zkoss.org/zkdemo/tree/multiple_selection

(Erişim Tarihi: 15.030.2014).

[16] Rules to build user friendly menus, Rules to build user friendly menus, http://freeplane.sourceforge.net/wiki/index.php/Rules_to_build_user_friendly_menus (Erişim Tarihi: 16.03.2014).

[17] Başbakanlık, http://dtvt.basbakanlik.gov.tr/AnaSayfa.aspx

(Erişim Tarihi 20.03.2012).

[18] XUL, https://developer.mozilla.org/en-US/docs/XUL

(Erişim Tarihi: 09.11.2013).

[19] ZK Framework Hakkında, http://www.kamuranyilmaz.com/zk/2013/09/05/zk-framework (Erişim Tarihi: 17.10.2013).

[20] Dayıbaş, O., Spring java/j2ee Uygulama Çatısı & Spring AOP, 1-25. Ankara, 2014.

[21] Akdoğan, A., iBatis, Ankara, 2006.

[22] Tomcat nedir,

http://www.serefakyuz.com/2011/06/tomcat-nedir-nasl-kurulur.html (Erişim Tarihi: 13.05.2014).

[23] Niçin Java Programlama Dili,

http://web.sakarya.edu.tr/~umit/panel/dosya/hafta1b.pdf (Erişim Tarihi: 13.05.2014).

[24] DES, http://tr.wikipedia.org/wiki/DES (Erişim Tarihi: 12.02.2014).

[25] MD5, http://tr.wikipedia.org/wiki/MD5 (Erişim Tarihi: 10.02.2014).

[26] Configuring Java CAPS for SSL Support, http://docs.oracle.com/cd/E19509-01/820-3503/ggfen/index.html (Erişim Tarihi: 21.01.2014).

[27] ModalPopup Kontrolü ile Kullanıcı Dostu Mesaj Pencereleri Çıkarmak, http://social.msdn.microsoft.com/Forums/tr-TR/2053d7bb-53c6-47f0-9fdb-ce99d37b0ef1/modalpopup-kontrol-ile-kullanc-dostu-mesaj-pencereleri-karmak?forum=aspnettr (Erişim Tarihi: 21.02.2014).

[28] Puurula, A., Combining Modifications to Multinomial Naive Bayes for Text Classification, 114-125. Information Retrieval Technology, Lecture Notes in Computer Science, New Zealand, 2012.

[29] Çavuşlar, M., Eren, K., Utku, M.,, Web Sayfalarından Bilgi Çıkarımı: Web Sitelerinden Bilgi Çıkarımı Yapan bir Uygulama ve Cümle Sınıflaması, Akıllı Bilgi Erişimi, NAMIK KEMAL ÜNİVERSİTESİ, Çorlu, 2013.

[30] Soukoreff, R.W., MacKenzie, I.S., Measuring Errors in Text Entry Tasks: An Application of the Levenshtein String Distance Statistic, 319-320. Measuring Errors in Text Entry Tasks, Extended Abstracts of CHI 2001, Toronto, Ontario Canada, 2001.

[31] Gooskens, C., Heeringa, W., Perceptive evaluation of Levenshtein dialect distance measurements using Norwegian dialect data, Language variation and Change, Norveç, 2004.

[32] Küçük, A.S.,Yazılım Testi Metodolojileri,

http://www.iztim.com/Blog/YazilimTeknolojisi/YAZILIM-METODOLOJI (Erişim Tarihi: 28.03.2014).

[33] What is Netsparker?,

https://www.netsparker.com/netsparker/ (Erişim Tarihi: 28.03.2014).

[34] JavaTM SE 6 Sürümünün Microsoft Windows Yüklenmesi,

http://www.baskent.edu.tr/~tkaracay/etudio/ders/prg/java/ch01/jdk_install.htm (Erişim Tarihi: 12.08.2013).

[35] UML, http://tr.wikipedia.org/wiki/UML (Erişim Tarihi: 16.08.2012).

[36] *Elektronik Belge Yönetim Sistemi e-içişleri projesi, İçişleri Bakanlığı Bilgi İşlem Daire Başkanlığı*, Ankara, 2013.

[37] *DMİ Genel Müdürlüğü EBYS Web Modülü Kullanım Kılavuzu, 2011, Ankara.*

[38] Tubitak Ulusal Bilgi Güvenliği Kapısı,

https://www.bilgiguvenligi.gov.tr/guvenlik-teknolojileri/sertifika-sertifika-olusturma-sertifika-turleri.html (Erişim Tarihi: 03.12.2013).

[39] İş Bankası Parakod uygulaması,

http://www.isbank.com.tr/content/TR/Bizi_Taniyin/Bizden_Haberler/Detay/Is_Bankasinin_Parakod_uygulamasi_ile_faturalar_artik_cep_telefonundan_odeniyor-562-3190.aspx (Erişim Tarihi: 13.10.2012).

[40] Kılıç, El., Güngör, Z., Web Site Tasarımlarında kullanılabilirlik değerlendirme yöntemlerinin önemi, Gazi Üniversitesi, Ankara, 2012.

[41] MVC, http://tr.wikipedia.org/wiki/Model-View-Controller

(Erişim Tarihi: 27.04.2014).

[42] Han, P., Wang, D., Zhao, Q., The research on Chinese document clustering based on WEKA, IEEE, Guilin, Nanjing, China, 2011.

[43] Bouckaert, R.R., Frank, E., Hall, M.A., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H., WEKA—Experiences with a Java Open-Source Project, *Journal The Journal of Machine Learning Research, 2533-2541. Hamilton, New Zealand, 2010.*

[44] Talia, D., Trunfio, P., Verta, O., Weka4WS: A WSRF-Enabled Weka Toolkit for Distributed Data Mining on Grids, 309-320. Edinburgh, 2006.

[45] Önaçan, M.B.K., Medeni,T.D., Özkanlı, Ö., Elektronik Belge Yönetim Sistemi (EBYS)'nin faydaları ve kurum bünyesinde EBYS yapılandırmaya, 1-23. Sayıştay Dergi, Ankara, 2012.

[46] Spring Framework'ün Faydaları, http://www.akilliyazilim.org/spring-framework-dersleri/spring-nedir-neden-spring.html
(Erişim Tarihi: 27.04.2014).

[47] On, A., Barışçı, N., Web Tabanlı Elektronik Belge Yönetim Sistemi, *ISITES 2014- AKADEMIK PLATFORM*, KARABÜK, 2014.