ANKARA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

GÜVENLİ DOKÜMAN YÖNETİM SİSTEMİ

Ergin ÖZEKES

ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

ANKARA
2009

Her hakkı saklıdır

TEZ ONAYI

Ergin ÖZEKES tarafından hazırlanan "**Güvenli Doküman Yönetim Sistemi**" adlı tez çalışması 05.01.2009 tarihinde aşağıdaki jüri tarafından oy birliği ile Ankara Üniversitesi Fen Bilimleri Enstitüsü Elektronik Mühendisliği Anabilim Dalı'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Danışman **:** Doç.Dr. H. Gökhan İLK

Jüri Üyeleri **:**

Başkan **:** Doç.Dr. Fatih Vehbi ÇELEBİ
Ankara Üniversitesi Bilgisayar Mühendisliği Anabilim Dalı

Üye **:** Doç.Dr. H. Gökhan İLK
Ankara Üniversitesi Elektronik Mühendisliği Anabilim Dalı

Üye **:** Yrd.Doç.Dr. H. Alparslan ILGIN
Ankara Üniversitesi Elektronik Mühendisliği Anabilim Dalı

Yukarıdaki sonucu onaylarım.

Prof.Dr. Orhan ATAKOL
Enstitü Müdürü

ÖZET

Yüksek Lisans Tezi

GÜVENLİ DOKUMAN YÖNETİM SİSTEMİ

Ergin ÖZEKES

Ankara Üniversitesi
Fen Bilimleri Enstitüsü
Elektronik Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. H. Gökhan İLK

GDYS (Güvenli Doküman Yönetim Sistemi), çeşitli dosya formatlarında olan dokümanların güvenli ortamlarda sunulması, etkin kullanımı ve içerik kalitesinin artırılmasını amaçlayan yayın yönetim sistemidir. Akademik çalışmalar sonucu yapılan yayınlar GDYS sisteminin temel girdisini teşkil eder. Akademik yayınların bir bütün içinde yönetilip araştırma amaçlı sunulabilmesi, sistemin rahat yapılandırılması, doküman inceleme yeteneklerinin en üst düzeyde gerçekleştirilmesi amaçlanmıştır.

GDYS, gereksinim analizi yapılarak en modern yazılım tasarım aracı olan BMD (Birleşik Modelleme Dili) ile tasarlanmıştır. GDYS geniş kitlelere yayınları güvenli olarak sunulabilmesi için bir ağ uygulaması olarak Java yetkili ağ hizmeti olan Tomcat üstünde geliştirilmiştir. Güvenliğin en yüksek mertebede gerçeklenebilmesi için OpenLdap dizin hizmetinden faydalanılmıştır. Sistem yönetimi ve sistemin sağlaması gereken genel işlevselliğe ait veri kaynağı olarak açık kaynak kodlu ve güçlü veritabanı sistemi Postgresql kullanılmıştır. Ana parçalar belirlendikten sonra genel işlevsellik ortaya konarak GDYS ile hangi süreçlerin gerçeklenebileceği belirlenmiş ve sonra bu süreçler için ana veri yapılarına karar verilmiştir.

Gerekli veri tabanı yapısı oluşturulup optimize edilerek BMD ile tasarlanmış ağ ara katmanı uygulaması, tasarım şablonları kullanılması ile Tomcat ağ sunucu üstünde geliştirilmiştir. Geliştirme ortamı olarak seçilen açık kaynak kodlu Eclipse ile ara katman geliştirimi için Java dili, Postgresql veri tabanı üstünde yapılması gereken süreçler için ise YD/YSD (Yordamsal dil/Yapılandırılmış Sorgu Dili) kullanılmıştır.

Kırılması oldukça zor olan, yapılandırılması tanımlı yetkili kişilerce yapılabilen, her doküman için eski sürümlerinin tutulduğu ve güncel dokümanlara ait anahtar kelimeler ya da kelimelerin tamamı üstünden çeşitli aramaların yapılabildiği, kayıtlı kişilere ait bilgilerin görülebildiği ve onlara ait dokümanların takip edilip belirli şablonlar ve tanımlanan görevler ile dokümanlardaki kalitenin artırılabildiği, kişilerin toplantılarının takip edildiği ve tanımlanan yetkiler doğrultusunda sistemle etkileşebilen GDYS sistemi geliştirilmiştir.

Sonuç olarak bireylerin tümleşik bir sistem altında grup olarak çalışmaları durumunda birbirleri ile etkileştiklerinde bilgi paylaşımı sağlayarak, güvenli ve aynı zamanda bilirkişi denetiminin sağladığı tecrübeleri de kullanarak daha hızlı ve etkin akademik yayın üretmelerine imkân sağlayacak bir yazılım sistemi sunulmuştur.

Ocak 2009, 108 sayfa

Anahtar Kelimeler: Sürüm Kontrolü, Doküman Tipleri, Anahtar Kelime Arama, Güvenli Bağlantı, Yetkilendirme

ABSTRACT

Master Thesis

SECURE DOCUMENT MANAGEMENT SYSTEM

Ergin ÖZEKES

Ankara University
Graduate School of Natural and Applied Sciences
Department of Electronical Engineering

Supervisor : Assoc.Prof.Dr. H. Gökhan İLK

SDMS (Secure Document Management System) can be used to present different kinds of documents in a secure environment and boost the quality of content. The basic inputs of this system are the publications, that are produced at the end of an academic work. The purpose of the SDMS is to manage academic publications as a whole and at the same time delivers a tool which enables joint research rather than individual effort.

To configure the system easily and to provide search skills at the high level requirement analysis has been done by the most popular modelling tool which is UML. SDMS was developed on the Tomcat that is java enabled web server to publish the academic documents to a wider community. To provide security, at the high level, openldap server has been used. The data source required for system management and for general functionality is Postgresql database which is opensource and it is powerful. First the main functionality and the process of the SDMS was defined then main data structures were decided.

After the database structure was optimised, middleware web application was developed using the design templates and UML. To develop middleware application; java programming language was used. In order to develop database, side logic pl/sql language was used. Development environment used is Eclipse which is open source.

The resulting software is an unbreakable, secure system that enables academic staff to work in a joint manner. Configuration of this system can be done by any authorized person. Documents in this system can be configured and text search can be done over the current documents for any keyword. Information that belongs to users who are the registered users of that system can be queried. Documents which belong to registered users can be matured using tasks and templates which are assigned to them by a moderator. Users can interact with the SDMS system according to their priviledges. Meetings can be assigned and staff can search them according to keywords. SDMS was developed to mature academic publishing in a flexible secure managed environment.

January 2009, 108 pages

Key Words : Configuration Management, Document Types, Keyword Search, Secure Connection, Authentication

ÖNSÖZ ve TEŞEKKÜR

Bu sistem akademik çalışmaların meyvesi olan yayınların geliştirilip standartlar etrafında olgunlaştırılması için tasarlanmıştır.

Bu sistemi daha geniş çaplı kitlelere hitap etmesi için ağ uygulaması olması düşünüldü. Bilinir ki ağ uygulamalarının en büyük zayıflığı ise güvenliktir. GDYS sisteminde güvenlik üstünde makul olduğu kadarı ile durulmuştur. İlk giriş klasör hizmeti olan OpenLdap tarafından yapılmaktadır.

Aynı zamanda veri bütünlüğü sağlanması için kullanılan Postgresql veri tabanın da verimli çalışması gereklidir. Ağ uygulamalarının en verimli çalıştığı platformun da Linux olması ve sistemin düşük kapasiteli masa üstü bilgisayarlarda dahi koşturulabilmesi için Linux platformunun gücünden faydalanılmıştır.

Kullanıcının rahat kullanabilmesi için alışıldık taşınabilir programlardaki ara yüzlere yakın ve o ara yüzler kadar yetenekli bileşenleri sağlayan JBOSS işçatısı kullanılmıştır. En problemli geliştirme aşaması ise JBOSS işçatısının arkasındaki EJBG (Eşzamansız Java Betiği ve GİD) teknolojisine hâkim olmak olmuştur.

Veritabanı yapısı ve ağ uygulamasının ana denetim mekanizmasının tasarlanmasında BMD'nin ve tasarım şablonlarının gücünden faydanılmıştır.

Yapılan geliştirimler sonucunda örneği fazla olmayan bir sistem geliştirilmiştir.

Çalışmalarımı yönlendiren, araştırmalarımın her aşamasında bilgi, öneri ve yardımlarını esirgemeyerek akademik ortamda olduğu kadar beşeri ilişkilerde de engin fikirleriyle yetişme ve gelişmeme katkıda bulunan, çalışmalarım süresince maddi manevi desteklerini esirgemeyen, bilimsel yaklaşımı kendisinden öğrenmeye çalıştığım danışanım ve değerli hocam Doç.Dr. H.Gökhan İLK'e (Ankara Üniversitesi Mühendislik Fakültesi), çalışmalarım süresince birçok fedakârlıklar göstererek beni destekleyen eşim ve kızıma en derin duygularla teşekkür ederim.

Ergin ÖZEKES
Ankara, Ocak 2009

İÇİNDEKİLER

ÖZET..... i
ABSTRACT..... ii
ÖNSÖZ ve TEŞEKKÜR..... iii
KISALTMALAR DİZİNİ (TÜRKÇE - İNGİLİZCE)..... vi
ŞEKİLLER DİZİNİ..... ix
ÇİZELGELER DİZİNİ..... x
1. GİRİŞ..... 1
1.1 Kuramsal Temeller 5
1.1.1 BMD..... 6
1.1.2 Tasarım şablonları..... 6
1.1.3 Kimlik doğrulama ve yetkilendirme..... 7
1.1.3.1 HDEP..... 8
1.1.4 Uygulama servisi..... 9
1.1.5 EJBG..... 9
1.1.6 Veri tabanı..... 10
1.1.7 Sürümleme 11
1.1.8 Örnek sistemler..... 12
1.1.8.1 Knowledge Tree 12
1.1.8.2 Alfresco..... 13
1.1.8.3 Documentum..... 13
1.1.8.4 Main Pyrus DMS..... 13
1.1.8.5 OpenKM..... 14
2. MATERYAL VE YÖNTEM..... 15
2.1 GDYS Sistemi Analizi ve Tasarımı..... 15
2.1.1 BS (Birleşik Süreçler) yazılım geliştirme süreci..... 17
2.1.1.1 Başlangıç aşaması..... 18
2.1.1.1.1 İş kullanım vakaları..... 18
2.1.1.1.1.1 Kullanıcı girişi iş kullanım vakası 20
2.1.1.1.1.2 Doküman yükleme iş kullanım vakası...... 21
2.1.1.1.1.3 Kullanıcı ekleme iş kullanım vakası..... 23
2.1.1.3 Detaylandırma aşaması..... 23
2.1.1.4 Yapılandırma aşaması..... 26
2.1.1.5 Geçiş aşaması..... 27
2.1.2 GDYS sistemi veri tabanı tasarımı..... 28
2.1.2.1 Fazlalık ve veri aykırılığı..... 30
2.1.2.2 Fonksiyonel bağımlılık..... 31
2.1.2.3 Fonksiyonel bağımlılık notasyonu..... 31
2.1.2.4 Bileşik determinant..... 32
2.1.2.5 Tam fonksiyonel bağımlılık..... 32
2.1.2.6 Kismi fonksiyonel bağımlılık..... 32
2.1.2.7 Geçişsel bağımlılık..... 32
2.1.2.8 Fonksiyonel bağımlılık sonuç aksiyomları (Armstrong aksiyomları)..... 32
2.1.2.8.1 Yansıma aksiyomu..... 32
2.1.2.8.2 Artırma aksiyomu..... 33
2.1.2.8.3 Geçişlilik aksiyomu..... 33

2.1.2.8.4 Yalancı (Pseudo) geçişlilik aksiyomu..... 33
2.1.2.8.5 Birleşme aksiyomu..... 33
2.1.2.8.6 Ayrışma aksiyomu..... 33
2.1.2.9 Fonksiyonel bağımlılığın kapalı kümesi..... 33
2.1.2.10 Kapsama..... 34
2.1.2.11 Normalizasyon..... 35
2.1.2.11.1 Normalizasyonun amacı..... 36
2.1.2.11.2 Normalizasyon adımları..... 36
2.1.2.11.2.1 Birinci normal form (1NF) 38
2.1.2.11.2.2 İkinci normal form (2NF) 38
2.1.2.11.2.3 Üçüncü normal form (3NF) 38
2.1.2.11.2.4 Boyce–Codd normal form (BCNF) 38
2.1.2.11.2.5 Beşinci normal form (5NF) 38
2.1.2.11.2.6 Alan (Domain)/Anahtar (Key) normal form (AA/NF)..... 38
2.1.2.11.3 Normalizasyon dezavantajları..... 39
2.1.2.12 Denormalizasyon..... 39
2.1.2.12.1 Denormalizasyon Temel Tipleri..... 39
2.1.3 GDYS sistemi ağ tasarımı..... 43
2.1.3.1 JSY..... 43
2.1.3.2 JSY yaşam döngüsü..... 46
2.1.3.2.1 Seyir düzeltme (Restore view)..... 46
2.1.3.2.2 İstek değerlerini uygulama (Apply request values)..... 46
2.1.3.2.3 Süreç doğrulama (Process validations)..... 47
2.1.3.2.4 Model değerlerini güncelleme (Update model values)..... 47
2.1.3.2.5 Uygulama çağırma (Invoke application)..... 47
2.1.3.2.6 Cevap verme (Render response)..... 47
2.2 GDYS Genel Yapısı..... 52
2.2.1 Tomcat ağ sunucusu..... 53
2.2.1.1 Tomcat kurulum klasörleri ve mimarisi..... 54
2.2.1.2 Ağ uygulaması mantıksal yapısı..... 59
2.2.1.3 MSD 61
2.2.1.3.1 Model..... 61
2.2.1.3.2 Seyir (View)..... 61
2.2.1.3.3 Denetleyici (Controller)..... 62
2.2.1.4 JGS platform, uygulama sunucusu ve ağ kabı..... 62
2.2.1.5 JYKS..... 65
2.2.1.6 JİDA..... 72
2.2.2 OpenLDAP sunucusu..... 74
2.2.3 JVTB ve PostgreSQL sunucusu..... 81
3. BULGULAR..... 92
3.1 Test Vakaları..... 92
3.1.1 Cactus..... 92
3.1.2 Elle yapılan test vakaları..... 93
3.2 GDYS Sistemi Test Vakaları..... 94
3.3 Test Vaka Sonuçları..... 103
4. SONUÇ ve TARTIŞMA..... 104
KAYNAKLAR..... 106
ÖZGEÇMİŞ..... 108

KISALTMALAR DİZİNİ (TÜRKÇE - İNGİLİZCE)

GDYS	Güvenli Doküman Yönetim Sistemi
DYS	Doküman Yönetim Sistemi
JSS	Java Sunucu Sayfası
JSY	Java Sunucu Yüzleri
GGL	Gnu Genel Lisansı
GİY	Girişim İçerik Yönetimi
EDYS	Elektronik Doküman Yönetim Sistemi
OUBS	Ortak Unix Baskı Sistemi
JGS	Java Girişim Sürümü
JSV	Java Standard Sürümü
DR	Doküman Resimleme
AİY	Ağ İçerik Yönetimi
BMD	Birleşik Modelleme Dili
JYKS	Java Yetkilendirme ve Kimlik Denetimi Servisi
HDEP	Hafif Dizin Erişim Protokolü
HVDP	HDEP Veri Değişim Protokolü
UPA	Uygulama Programlama Ara yüzü
GİD	Genişletilebilir İşaretleme Dili
EJBG	Eşzamansız Java Betiği ve GİD
VTYS	Veri Tabanı Yönetim Sistemi
İVTYS	İlişkisel VTYS
VMD	Veri Muamele Dili
VTD	Veri Tanımlama Dili
YSD	Yapılandırılmış Sorgu Dili
YD/YSD	Yordamsal Dil/YSD
$C_H	Tomcat kurulum klasörü
KİD	Kablosuz İşaretleme Dili
GJF	Girişim Java Fasulyesi
JMS	Java Mesajlaşma Servisi
JYE	Java Yönetim Eklentileri
JMU	Java Muamele UPA
JİDA	Java İsimleme ve Dizin Ara yüzü
JVTB	Java Veri Tabanı Bağlantısı
GÇJU	GİD Çözümleme için Java UPA
GKJP	GİD Kayıtları için Java UPA
KJYK	Kaplar için Java Yetkilendirme Servisi Sağlayıcı Kontratı
UYÇ	Uzak Yordam Çağrısı
GUJU	GID temelli UYÇ için Java UPA
SYUP	Servis Yönelimli Uygulama Programlama
JSEU	Java için SYUP Eklentileri UPA
JKEİ	Java Kabı Etkinleştirme İşçatısı
TKDM	Takılabilir Kimlik Denetimi Modülü
AİS	Alan İsmi Sistemi
NDS	Novell Dizin Sistemi
UTB	Uluslararası Telekomünikasyon Birliği
USO	Uluslararası Standartlar Organizasyonu

DSA	Dizin Servisi Ajanı
DKA	Dizin Kullanıcısı Ajanı
DEP	Dizin Erişimi Protokolü
GAA	Göreceli Ayrıştırma Adı
SSA	Servis Sağlayıcısı Ara yüzü
SGT	Sistem Gereksinim Tanımlamaları
MSD	Model Seyir Denetleyici
AVTB	Açık Veritabanı Bağlantısı
ATD	Arayız Tanımlama Dili
UMÇ	Uzak Metot Çağrısı
İNTA-UMÇ	İnternet Nesne Talebi Aracıları Arası Protokolüne Uzak Metot Çağrısı
TDP/İP	Taşıma Denetim Protokolü/İnternet Protokolü
AMTP	Atlamalı Metin Taşıma Protokolü
BKY	Birleştirilmiş Kaynak Yerleştiricisi
BKÇ	Birleştirilmiş Kaynak Çağrımı
VTN	Veri Taşıma Nesnesi
VEN	Veri Erişim Nesnesi
AMİD	Atlamalı Metin İşaretleme Dili
BGO	Bütünleşik Geliştirme Ortamı
JSM	Java Sanal Makine
JSP	Java Server Page
JSF	Java Server Faces
GPL	Gnu Public License
ECM	Enterprise Content Management
EDMS	Electronic Document Management System
CUPS	Common Unix Printing System
J2EE	Java Enterprise Edition
J2SE	Java Standard Edition
DI	Document Imaging
WCM	Web Content Management
UML	Unified Modelling Language
JAAS	Java Authentication and Authorisation Service
LDAP	Lightweight Directory Access Protocol
LDIF	LDAP Data Interchange Format
API	Application Programming Interface
AJAX	Asynchronous JavaScript and XML
DBMS	Data Base Management System
RDBMS	Relational Database Management Systems
DML	Data Manipulation Language
DDL	Data Definition Language
SQL	Structured Query Language
PL/SQL	Procedural Language/Structured Query Language
$C_H	Tomcat kurulum klasörü
WML	Wireless Markup Language
XML	Extensible Markup Language
EJB	Enterprise JavaBeans
JMS	Java Messaging Service

JMX	Java Management Extensions
JTA	Java Transaction API
JNDI	Java Naming and Directory Interface
JDBC	Java Database Connectivity
JAXP	Java API for XML Parsing
JAXR	Java API for XML Registries
JACC	Java Authorization Service Provider Contract for Containers
RPC	Remote Procedure Call
JAX_RPC	Java API for XML-based RPC
SOAP	Servise Oriented Applicatipn Programming
SAAJ	SOAP Attachments API for Java
JAF	JavaBeans Activation Framework
PAM	Pluggable Authentication Modules
DNS	Domain Name System
NDS	Novell Directory Service
ITU-T	International Telecommunications Union
ISO	International Standards Organization
DSA	Directory Service Agents
DUA	Directory User Agents
DAP	Directory Access Protocol
RDN	Relative Distinguished Name
SPI	Service Provider Interface
UP	Unified Proccess
SRS	System Requirement Specifications
ODBC	Open Database Connectivity
IDL	Interface Definition Language
RMI	Remote Method Invocation
RMI_IIOP	Remote Method Invocation to Internet Inter-ORB Protocol
TCP/IP	Transport Control Protocol/Internet Protocol
HTTP	HyperText Transport Protocol
URL	Unified Resource Locator
URI	Unified Resource Invocator
DTO	Data Transfer Object
DAO	Data Access Object
HTML	HyperText Markup Language
IDE	Integrated Development Environment
JVM	Java Virtual Machine

ŞEKİLLER DİZİNİ

Şekil 2.1 Çağlayan yazılım geliştirme mimarisi..... 16
Şekil 2.2 İteratif yazılım geliştirme mimarisi..... 17
Şekil 2.3 BS'nin zaman içinde her bir adımında gerçeklenen temel süreçlerin Dağılımı..... 17
Şekil 2.4 GDYS sistemi iş kullanım vakaları diyagramı..... 19
Şekil 2.5 Genel etkileşim süreci..... 20
Şekil 2.6 Kullanıcı girişi etkinlik diyagramı..... 21
Şekil 2.7 Doküman yükleme etkinlik diyagramı..... 22
Şekil 2.8 Kullanıcı ekleme etkinlik diyagramı..... 23
Şekil 2.9 Yükleme kullanım vakasına ait sekans diyagramı..... 24
Şekil 2.10 Yükleme kullanım vakasına ait işbirlik diyagramı..... 25
Şekil 2.11 GDYS sistemi sınıf diyagramı..... 26
Şekil 2.12 GDYS sistemi bileşen diyagramı..... 27
Şekil 2.13 GDYS sistemi kurulum diyagramı..... 28
Şekil 2.14 Veritabanı tasarımı genel aşamaları..... 29
Şekil 2.15 Normalizasyon adımları..... 37
Şekil 2.16 GDYS sistemi veri modeli..... 42
Şekil 2.17 Dört katmanlı JSY uygulama yapısı..... 43
Şekil 2.18 JSY yapı taşları..... 44
Şekil 2.19 JSY yaşam döngüsü..... 46
Şekil 2.20 Karma şablonu yapısı..... 48
Şekil 2.21 Gözetmen şablonu yapısı..... 49
Şekil 2.22 GDYS ağ uygulaması modeli..... 51
Şekil 2.23 GDYS genel yapısı..... 53
Şekil 2.24 Ağ uygulaması dosya hiyerarşisi..... 56
Şekil 2.25 Tomcat mimarisi..... 57
Şekil 2.26 GDYS ağ uygulaması mantıksal yapısı..... 60
Şekil 2.27 MSD şablonu..... 61
Şekil 2.28 OpenLdap'ın kullandığı veritabanı yapısı..... 76
Şekil 2.29 JVTB mimarisi..... 82
Şekil 2.30 PostgreSQL depolama hiyerarşisi..... 87
Şekil 2.31 Kurulum sonrası dosya yerleşimi..... 88
Şekil 2.32 GDYS sistemi varlık ilişki diyagramı..... 91
Şekil 3.1 Cactus çalışma mekanizması..... 92
Şekil 3.2 İndis sayfası..... 94
Şekil 3.3 Ana menü…..... 95
Şekil 3.4 Listeleme sayfası..... 96
Şekil 3.5 Kayıt ekleme..... 97
Şekil 3.6 Güncelleme..... 98
Şekil 3.7 Tarayıcı..... 99
Şekil 3.8 Değerler listesi..... 100
Şekil 3.9 Takvim..... 101
Şekil 3.10 Role eylem atama..... 102

ÇİZELGELER DİZİNİ

Çizelge 2.1 Nesne modeli ile veri modeli arası farklılıklar..... 40
Çizelge 3.1 Test vaka sonuçları..... 103

1. GİRİŞ

1980'li yılların başında kurumların veya kişilerin bilgisayarda oluşturulmuş kısıtlı tipli dosyalarının depolanması, indislenmesi ve hızlı erişim ihtiyacı doküman yönetimi yapan sistemlerin ortaya çıkmasına sebep olmuştur (Documentum, SharePoint). Yazı veya dosyayı tarayarak elektronik ortama taşıyan sistemler doküman görüntüleme sistemleri olarak doğmuştur. Doküman yönetimi alanında geliştirilmiş sistemler zaman içinde EDYS (Elektronik Doküman Yönetim Sistemi), DR (Doküman Resimleme), DYS (Doküman Yönetim Sistemi), AİY (Ağ İçerik Yönetimi), GİY (Girişim İçerik Yönetimi) olarak evirilmiştir. Gelişen bu sistemler daha farklı dosya tiplerini destekleyen, şebeke ortamında depolayan, güvenlik ve yetkilendirme yapan, işbirliklerini destekleyen sistemlere dönüşüp DYS olarak adlandırılmışlardır.

Genel olarak GDYS bilgisayar programları kümesidir. Amacı elektronik yazı ve resimleri depolamak ve takip etmektir. GDYS aşağıdaki işlevselliklerle ilgilenir:

1 Yer: Yazıyı doldurmak ve erişmek için GDYS içinde yapılması gereken gezinmenin yapılacağı ortam.
2 Dosyalama: Yazının indis ile nasıl organize edileceğini belirler.
3 Geri Alım: Dokümanlar arası gezinme ve arama.
4 Güvenlik: Yetkisiz kişilerden yazıların saklanması.
5 Felaket Kurtarımı: Dokümanların felaket anlarında nasıl korunup kurtarılacağının belirlenmesi.
6 Arşivleme: Eski yazıların gelecek kullanım için korunması.
7 Dağıtım: Yazılara ihtiyacı olan kişilere bu yazının nasıl dağıtılacağının belirlenmesi.
8 İş akışı: Bir dokümanın birden fazla kişi tarafından işlem görmesi gerektiği anlarda bu sürecin nasıl olacağının belirlenmesi.
9 Yaratım: Dokümanların nasıl oluşturulacağının belirlenmesi.
10 Denetim: Kimin bu sistemi kullanacağının belirlenmesi.

Geliştirilen GDYS sisteminde farklı coğrafyalarda çalışmakta olan bilim insanlarının bütünleşik bir sistem altında grup olarak çalışmaları amaçlanmıştır. Akademik

çalışmalarının sonucu olan yayınların aynı anda birden fazla uçta kontrollü olarak görünebilir ve düzenlenebilir, araştırma yapanlar hakkında çalışma ve toplantıları ile ilgili bilgiler sağlanabilir ve yayınların diğer akademik çalışmalara yardımcı olması için daha hızlı anahtar kelime araştırması yapılabilir olmasını sağlayan işlevsellikler kümesini sunmak ana amaçtır. GDYS sisteminin amaçlarını yerine getirebilmesi için sistem gereksinimi analizi sırasında belirlenen isterler şöyle listelenmiştir:

1. Sisteme giriş bir noktadan ve kontrollü olmalıdır. GDYS sistemi kullanıcısı sahip olduğu kullanıcı adı ve şifre ile sisteme girmeli ve sistemden çıkıncaya kadar bir daha giriş yapmasına gerek olmamalıdır. Şifre kontrolü merkezi yetkilendirme hizmeti tarafından yapılmalı ve kontrol hizmetinin ihtiyaç duyduğu bilgiler merkezi veri deposunda tutulmalıdır.
2. Sistem içinde kullanıcılar sahip oldukları haklar doğrultusunda süreçleri gerçekleyebilmelidir. Süreç gerçekleme veya sayfa görüntüleme sahip olunan haklar izin verdiği sürece yapılabilmelidir. Kullanıcı hakları sisteme girdikleri andan itibaren belli olmalıdır ve kullanıcı ile ilgili hak tanımları merkezi bir veri deposunda tutulmalıdır.
3. Kullanıcının sistem içinde yapmış olduğu eylemler merkezi bir veri deposunda günce altına alınmalıdır.
4. Doküman dosya tipleri düz metin, pdf (taşınabilir veri şekli), Windows Word dosya şekli, Windows Excel dosya şekli olabilmelidir. Dosyaların son orijinal örneği sistemin sahip olduğu dosya sistemi içinde tutulmalıdır. Dosya ile ilgili boyut, format, yüklenme tarihi, dizin yolu, yazarı, şablonu bilgileri merkezi veri deposunda tutulmalıdır.
5. Sisteme eklenen dosya boyutu sistem kaynaklarını verimli kullanabilmek için 20 MB'ı aşmamalıdır. Dosyalar ilk yüklenirken içlerinde aranılabilecek kelimeler dosya ile ilişkili olarak merkezi veri deposunda tutulmalıdır.
6. Mevcut dosyalar kullanıcılar tarafından indirilebilmeli ve düzenlemeler yapıldıktan sonra sisteme tekrar yüklenebilmelidir. Sistemden dosya indirme hakkına sahip kullanıcı, dosyayı indirirken dosya düzenleninceye kadar diğer kullanıcılara kilitlenmeli ve düzenleme sonrası sisteme tekrar geri yüklenirken bu kilit kaldırılmalıdır. Bu sırada dosyaya ait arama kelimeleri yeni hale göre

değiştirilmeli ve dosyanın bir önceki ile son hali arasındaki değişiklikleri gösteren farklar merkezi bir veri tabanında tutulmalıdır.

7. Dosya sahip olduğu şablona göre moderator tarafından incelenebilmeli ve gerekli görülen değişikler yapılmak üzere o dosyanın yazarına görev olarak atanabilmelidir.
8. Sistem yöneticisi tarafından kimlik bilgilerini içeren yeni kullanıcı tanımı eklenebilmeli, değiştirilebilmeli, silinebilmeli ve hakları atanabilmelidir. Sisteme yeni kullanıcı ekleme ile merkezi yetkilendirme hizmeti için gereken bilgiler girilebilmelidir.
9. Kullanıcı hakları içinde yapılabilinecek eylemler belirlenebilmelidir. Sistem yöneticisi tarafından yeni hak tanımı, bu hak içinde yapılabilinecek eylemler ve yeni eylem tanımı eklenebilmeli, değiştirilebilmeli ve silinebilmelidir.
10. Sistem kullanıcısının içinde bulunduğu birim belirlenebilmeli, sistem yöneticisi tarafından yeni birim eklenebilmeli, silinebilmeli, değiştirilebilmelidir.
11. Kullanıcılara ait toplantı bilgileri merkezi veri deposunda tutulmalıdır. Kullanıcı tarafından yeni toplantı bilgileri girilebilmeli, değiştirilebilmeli ve silinebilmelidir.
12. Sistem kullanıcısı tarafından yeni dosya şablonu ve dosya tipi tanımı eklenebilmeli, değiştirilebilmeli ve silinebilmelidir.

Yapılan sistem gereksinim analizi sonucu elde edilen ve yukarıda verilen isterlerin GDYS sistemi bünyesinde sunulabilmesi, bir yöntem etrafında uygulama altına alınabilmesine bağlıdır. Bölüm 1.1.1'de sistem tasarımı ve sistem içinde kendilerinden faydalanılacak alt yazılım sistemlerini geliştirmede kullanılacak BMD (Birleşik Modelleme Dili) anlatılmıştır. Bölüm 2.1.1.1.1'de anlatılmış iş kullanım vakaları BMD dili kullanılarak GDYS isterlerinden üretilmiştir. Bölüm 1.1.2'de sunulmuş tasarım şablonları kullanılarak iş kullanım vakalarından sistemin içermesi gereken alt yazılım sistemleri belirlenmiştir. Bölüm 2.1.1'de anlatılan BS (Birleşik Süreçler) yazılım geliştirme tekniği ile alt yazılım sistemleri geliştirilmişlerdir.

GDYS sisteminde ilgi alanlarında özelleşmiş alt yazılım sistemleri kullanılmasındaki ana amaç, sistemi belli mantıksal bloklara ayırarak yönetim, hata giderimi çabalarını

azaltmaktır. Örnek olarak veri tabanı yerine sistem genelinde kullanılacak veri, dosyalar sistemi içinde tutulsa idi verinin yönetimi ve verinin sistemle olan etkileşimini kontrol eden sistemin geliştirmesi gerekirdi. Kullanıcı yetkilendirme hizmeti de aynı sistem içinde sağlanılsa idi o sistemin güvenliğinin kırılgan olmasına sebep olan sorunları da çözmesi gerekirdi. Karmaşıklığın ve keşmekeş'in bir derece daha artacağı sistemde veri ile ilgili bir problem çözülürken yetkilendirmeye dair sağlanmış bir çözümün ortadan kalktığı diğer bir problemin ortaya çıkmasına sebep olunabilirdi. Aynı zamanda her iki hizmeti içeren tek bir sistemin yönetimi de ayrık sistemlere göre daha zor olabilirdi. Sonuç olarak sunulacak sistemin daha profesyonel olması, birbirlerinden ayrı olarak farklı gruplar tarafından geliştirilmiş yetenekli sistemler olan HDEP, uygulama servisi, sürüm kontrol sistemi ve veri tabanı alt yazılım sistemlerinin GDYS bünyesinde kullanılması ile sağlanılmıştır.

GDYS sistemi merkezde bölüm 2.1.3'de anlatılmış ağ hizmeti etrafında örgütlenmiştir. Bölüm 2.2.1'de açıklanmış Tomcat JGS (Java Girişim Sürümü) motoru, ağ hizmeti görevini üstlenmiştir. Uygulaması Tomcat tarafından sağlanan, yayınları geniş kitlelere sunmak ve sistemin büyük ölçekli taleplere güvenli olarak cevap verebilmesi amacıyla java dilinden faydalanan, yüksek başarımlı sonuçlar sağlayan, yapısı ve çalışma mekanizması bölüm 2.1.3.1'de detaylı olarak incelenmiş JSS (Java Sunucu Sayfası) teknolojisi kullanılmıştır. Bölüm 2.1.3.1'de kullanıcı ara yüzleri olan ve JSS teknolojisini uygulayan JSY (Java Sunucu Yüzleri) sayfaları açıklanmıştır. Bölüm 1.1.5'de değinilmiş EJBG, ağ sayfalarının etkileşim gücünü artırmak amacı ile kullanılmıştır. Bölüm 2.2.1.2'de Tomcat ağ hizmeti üstünde koşan ağ uygulamasının mantıksal yapısı sunulmuştur. Ağ uygulamasının mantıksal yapısının belirlenmesinde bölüm 2.2.1.3'de aktarılan MSD (Model Seyir Denetleyici) tasarım şablonundan faydalanılmıştır.

GDYS sisteminin en önemli özelliklerinden biri sistemle etkileşimin kontrollü olmasıdır. Bölüm 2.2.1.4'de bahsedilen JGS platformu içinde bölüm 2.2.1.5'de anlatılan JYKS (Java Yetkilendirme ve Kimlik Denetimi Servisi) vasıtası ile kontrol, kimlik denetimi ve yetkilendirmenin yapılabilmesi ve her eylemin kayıt altına alınabilmesi sağlanılmıştır. Bölüm 1.1.3.1'de anlatılan kimlik denetimi için HDEP (Hafif Dizin

Erişim Protokolü) kullanan GDYS sisteminde bölüm 2.2.2'de anlatılan OpenLdap açık kaynak kodlu hizmetinden faydalanılmıştır. Yetkilendirme ve eylemlerin kayıt edilebilmesi GDYS sistemi içinde geliştirilmiş sınıflarca sağlanılmıştır. Bölüm 2.2.1.6'da anlatılmış JİDA OpenLdap'a Tomcat ağ sunucusu içinden bağlanmayı sağlayan işçatısıdır.

Bölüm 1.1.7'de bahsedilen ve yayınların kalitesini artırmayı amaçlayan sürüm kontrol hizmeti, GDYS bünyesinde gerçeklenilmiştir. Aynı anda bir denetmen tarafından o yayınla ilgili yapılması gereken değişiklikleri içeren görevler ilgililere atanabilir ve uyulması gereken şablonlar belirlenebilir. Yapılan çalışmaların diğer araştırmacılar tarafından belli yetkiler doğrultusunda rahat araştırılabilir olması işlevselliği, her yayının güncel haline ait kelimelerin sürüm kontrolü içinde kayıt altına alınması yoluyla sağlanmıştır.

Veri tabanları, bölüm 2.1.2'de anlatılan tasarım teknikleri kullanılarak birbirleri ile ilişkileri ve yapıları tanımlanan kayıtlar bütünüdür ve yönetimleri yedekleme, arşivleme ve veri bütünlüğü sağlama gibi özellikleri olan veri tabanı sistemleri tarafından yapılır. GDYS sistemi içinde tutulan veri boyutu dosya sistemi içinde idare edilemeyecek boyutta olmasından dolayı sistem genelinde kullanılan bilgileri içeren tüm kayıtlar, bölüm 2.1.2.11'de açıklanan normalizasyon tekniği ile verimliliği artırılmış nesnel ilişkili veri tabanı sistemi içinde tutulmuştur. Bölüm 2.1.2.12'de normalizasyon sonucu meydan gelen performans düşüklüğünü artırmayı amaçlayan denormalizasyon tekniği anlatılmıştır. Bölüm 2.2.3'de yapılandırması ve yapısı anlatılmış açık kaynak kodlu en gelişkin veri tabanı sistemi olan ve PostgreSQL kullanılmıştır.

1.1 Kuramsal Temeller

Giriş bölümünde genel DYS sistemlerinin doğuşu ve işlevsellikleri, GDYS sisteminin amacı, sistem analizi sonucu belirlenen isterleri, bu isterleri yerine getirirken faydalanacağı hizmetler ve genel bölüm açıklamaları sunulmuştur. Kuramsal temeller kısmında GDYS sistemi geliştirimi için kullanılacak teknikler, sistemler özetlenmiş ve bu alandaki sistemlere örnekler sunulmuştur.

İlk olarak sistem analizi için kullanılmış BMD ele alınmış ve devamında tasarım şablonları incelenmiştir.

1.1.1 BMD

GDYS sisteminin tüm analiz, tasarım ve geliştirme süreçlerinde faydalanılmış BMD, bir programlama (ya da yazılım geliştirme) dili olmaktan ziyade iş sistemlerinin nasıl modellenebileceğini belirleyen ve açıklayan yöntemlerin bir araya toplanmış halidir (Booch *et al.* 1998). Daha çok yazılım geliştiriciler tarafından kullanılıyor olsa da iş sistemleri modellenmesinde de kullanılabilir.

Sisteme ait isterler, analiz ve tasarım çabaları içinde İş Kullanım Vakaları Diyagramlarının oluşmasını sağlarlar. Geri kalan tüm modelleme sürecinin temeli olan bu diyagramlar bir sonraki adımda kullanım vakalarına dönüşürler.

BMD'nin en kullanışlı diyagram türü olan Etkinlik Diyagramları ile yazılım haline getirilmek istenen süreçler herkesin anlayabileceği şekilde görüntülenebilir. Bu açıdan etkinlik diyagramları hem yazılımcıya hem de yazılımı kullanacak olan kişilere net bir görüş sağlar.

Bir iş sisteminin yapısını sade ve anlaşılır şekilde ortaya çıkarmak için Paket Diyagramı kullanılabilir. Sınıf Diyagramı ile nesnel yönelimli programlamada temeli teşkil eden sınıflar net şekilde gösterilebilir ve böylece sağlanan ek görsellik ile yazılım tasarlamanın ilerleyen aşamalarında daha yüksek verimlilik sağlanabilir.

BMD kullanılarak oluşturulmuş model tasarım şablonları ve bir programlama dili kullanılarak uygulama olarak vücut bulur.

1.1.2 Tasarım şablonları

BMD ve tasarım şablonları birbirini tamamlar ve ayrı olarak düşünülmemelidirler. Yazılım tasarımında ortak olarak karşılaşılan problemler için geliştirilmiş genel ve tekrar kullanılabilir çözümlerdir. Tasarım şablonu direkt olarak yazılıma dönüşen tamamlanmış tasarım değildir. Problemin nasıl çözüleceğini belirten açıklama veya

modeldir. Nesne yönelimli programlamada ise uygulama içindeki sınıf ve nesneleri tam olarak tanımlamaksızın nesneler ve sınıflar arasında ilişkiyi gösterir. Algoritmalar tasarım şablonu değildir, çünkü algoritmalar tasarımdan çok hesaplama problemi için çözüm sunar.

Tüm yazılım şablonları tasarım şablonu değildir. Tasarım şablonları yazılımın tasarım aşamasındaki problemler için çözüm sunar. Örnek olarak yapısal şablonları gibi diğer yazılım şablonları kendi alanlarındaki problemler için çözüm sunar.

Tasarım aşamasında önceden test edilmiş şablonları kullanarak başlangıçta önemsiz ve belli olmayan problemlerin yazılım yaşam döngüsü boyunca daha büyük bir hal alması engelleneceği gibi yazılımın daha etkin ve esnek olmasını sağlayacaktır.

Tasarım şablonları Yaratımsal, Yapısal ve Davranışsal şablonlar olarak sınıflandırılabilir. Yaratımsal tasarım şablonu için Fabrika, yapısal tasarım şablonu için MSD, davranışsal tasarım şablonu için Gözetmen şablonları verilebilir.

Takip eden bölümde kullanılmış iş çatılarından JYKS ele alınmış devamında JYKS'nin arka planda kullandığı HDEP özetlenmiştir.

1.1.3 Kimlik doğrulama ve yetkilendirme

Buraya kadar tasarlanan sistem, gerekli işçatıları ve hizmetler kullanılarak vücut bulmaya başlar. Kimlik doğrulama, sisteme girmek isteyen kişinin sistem tarafından bilinip bilinmediğini anlamak için ilk giriş anında gerçeklenir ve kişiyi sisteme tanıtan kullanıcı adı ve adla ilişkili özel anahtarın kişi tarafından doğru bilinip bilinmediği sorgulanarak yapılır. Güvenliği en yüksek seviyede sunmayı amaçlayan GDYS, kimlik denetimi ve yetkilendirme işlevi için arka planda çalışan HDEP dizin hizmetini kullanan JGS'nin JYKS (Java Yetkilendirme ve Kimlik Denetimi Servisi) ve JİDA (Java İsimleme ve Dizin Ara yüzü) işçatılarından faydalanır.

1.1.3.1 HDEP

TDP/İP üzerinde çalışan, dizin servislerini sorgulama ve değiştirme amacıyla kullanılan, istemci-sunucu modeline dayanan ve dizin hizmetlerine (directory services) erişebilmek için kullanılan, standart, genişletilebilir uygulama katmanı protokolüdür. Bu protokol, OpenLDAP, Sun Directory Server, Microsoft Active Directory gibi dizin sunucuları tarafından kullanılmaktadır.

Bir dizini kullanmakta rehberlik edecek dört model vardır:

- Bir dizin içine verinin nasıl ekleneceğini tanımlayan bilgi modeli,
- Dizin içinde bulunan belirli bir veriye nasıl ulaşılacağını ve düzenleneceğini belirleyen adlandırma modeli,
- Dizin verisi ile ne yapılacağını belirleyen işlevsel model,
- Dizin verilerini yetkisiz kullanıcılardan koruyacak güvenlik modeli.

HVDŞ (HDEP Veri Değiş tokuş Şekli) dizin verilerini değiş-tokuş etmek için standart metin biçimidir. HDEP istemci uygulamaları geliştirebilmek için HDEP UPA'ları kullanılabilir.

HDEP, mesaj kaynaklı bir protokoldür. Mesaj kaynaklı protokolde istemci istek içeren bir HDEP iletisi oluşturur ve mesajı sunucuya gönderir, sunucu gelen istemi işler ve sonucu bir veya birden fazla HDEP mesajı olarak istemciye gönderir.

HDEP ile istemci bir anda birden fazla istemde bulunabilir. Örneğin bir istemci iki arama işlemini aynı anda yapabilir. Birden fazla işlemi aynı anda yapabilmeyi mümkün kılması HDEP protokolünü AMTP ve benzeri protokollere göre daha esnek ve verimli bir protokol yapmaktadır.

Takip eden bölümde GDYS'nin merkezinde bulunan uygulama servisi ele alınmış devamında ağ uygulamasının ara yüzlerinde kullanılmış EJBG teknolojisi özetlenmiştir.

1.1.4 Uygulama servisi

GDYS sisteminde kullanıcı HDEP sisteminden geçerek uygulama servisi ile etkileşmeye başlar. Uygulama servisi, istemci bilgisayar ve cihazlarına uygulamayı teslim eden yazılım motorudur. Bunlara en iyi örnek ağ hizmetleri verilebilir. İstemcinin sistemle etkileşiminde gerçeklenmesi gereken iş mantığının büyük kısmı uygulama servisi üstünden sağlanır. Örnek olarak veri tabanı erişimi, ağ sayfalarından gelen verinin denetlenmesi verilebilir.

JGS yetkili uygulama hizmetleri arasında Weblogic, WebSphere, Jrun, Apache Geronimo, Oracle OC4J en çok kullanılanlarıdır (Chappell and Jewell 2002). Ağ uygulaması içeren GDYS, JGS yetkili uygulama hizmeti olarak Apache Tomcat kullanır. Servlet ve JSS sayfaları Tomcat içinde modül olarak kullanılır.

İçerik yönetimi hizmetlerinde olduğu gibi birden fazla uygulama hizmeti birden fazla platformda farklı uygulama olarak eş zamanlı olarak çalışabilir. Uygulama hizmeti kullanımının avantajları arasında güvenlik, etkinlik, merkezi yapılandırma, veri ve yazılım bütünlüğü, yapılan değişikliğin tüm istemcilere istemci üstünde değişiklik yapmaksızın sunulabilmesi sıralanabilir.

1.1.5 EJBG

Ağ uygulaması içeren GDYS'nin sahip olduğu sayfalarda etkileşim gücünü artırmayı amaçlayan EJBG, ağ sayfalarında javascript ve XMLHttpRequest çağrısı kullanımı ile etkileşimli uygulamalar yaratan tekniğin adıdır (Geary and Hortsmann 2007).

En yaygın kullanım alanı, sayfayı yeniden yüklemeye gerek kalmaksızın, sayfada görünür değişiklikler yapmaktır. XMLHttpRequest çağrısı kullanılarak birden fazla bağımsız işlem yapılabilir.

EJBG, etkileşimli ağ uygulamaları yaratmak için kullanılan ağ programlama tekniğidir. Ağ sayfalarının etkileşimini, hızını ve kullanılabilirliğini artırabilmek için arka planda

sunucuyla ufak miktarda veri değiş tokuşu yapılarak daha hızlı güncellenebilen ağ sayfaları oluşturulabilir.

EJBG'nin kullandığı tekniğin faydaları arasında sayfanın yüklenmesi ve sayfa ile etkileşimi birbirinden ayrı, farklı zamanlarda gerçeklenen süreçlere bölmesinden dolayı bant genişliğinden tasarruf sağlaması bulunmaktadır. Aynı zamanda daha esnek kullanıcı ara yüzleri oluşturulabildiği için ağ sayfası masa üstü uygulaması ara yüzü kadar yetenekli hale gelebilir.

Dezavantajları arasında ise ağ sayfası arama motorları tarafından indislenememesi, geri tuşuyla bir önceki sayfaya geri dönülememesi ve yetersiz bant genişliğinde istenmeyen davranışların sergilenmesi sıralanabilir.

Takip eden bölümde sistem içindeki verilerin idaresi ile yükümlü veri tabanı sistemi ve devamında sürümleme sistemi özetlenmiştir.

1.1.6 Veri tabanı

Her sistem gibi GDYS sisteminin idare etmekle yükümlü olduğu veriler mevcuttur. GDYS sistemi girişindeki HDEP, onun arkasındaki ağ uygulaması sistemin isterlerini yerine getirmek için kullanıcı, dosya ve benzeri yazılım unsurlarına ait verilere ihtiyaç duyar. Tüm GDYS sistemi verileri veritabanı olarak adlandırılan, sistematik erişim imkânı olan, yönetilebilir, güncellenebilir, taşınabilir, birbirleri arasında tanımlı ilişkiler bulunabilen bilgiler kümesi olarak organize edilir. Veritabanı için bir başka tanım da, bir bilgisayarda sistematik şekilde saklanmış, programlarca işlenebilecek veri yığınıdır.

Veri tabanında asıl önemli kavram, kayıt yığını ya da bilgi parçalarının tanımlanmasıdır. Şema adı verilen bu tanım, veri tabanında kullanılacak bilgi tanımlarının nasıl modelleneceğini gösterir. Şema, Veri Modeli olarak da adlandırılır ve şema oluşturma işlemine Veri Modelleme denir. En yaygın olanı, İlişkisel Model'dir. İlişkisel modelde veriler tablolarda saklanır. Tablolarda bulunan satırlar kayıtların kendisini, sütunlar ise bu kayıtları oluşturan bilgi parçalarının ne türden olduklarını belirtir.

GDYS sisteminde idare edilmesi gereken verinin boyutu dosya sistemi içinde idare edilebilecek miktarların çok üstündedir. Bu sebeple GDYS, bir veri tabanını oluşturmak, saklamak, çoğaltmak, güncellemek ve yönetmek için kullanılan VTYS (Veri Tabanı Yönetim Sistemi) adı verilen programlara ihtiyaç duyar. VTYS özelliklerinin ve yapısının nasıl olmasını gerektiğini inceleyen alan Bilgi Bilimidir.

Veri tabanı yönetim sistemleri, verileri sistematik bir biçimde depolayan yazılımlara verilen isimdir. Birçok yazılım bilgi depolayabilir ama aradaki fark, veri tabanın bu bilgiyi verimli ve hızlı bir şekilde yönetip değiştirebilmesidir. Bilgiye gerekli olduğu zaman ulaşabilmek esastır. İlişkisel Veri Tabanı Yönetim Sistemleri İVTYS (İlişkisel VTYS) büyük miktarlardaki verilerin güvenli bir şekilde tutulabildiği, bilgilere hızlı erişim imkânlarının sağlandığı, bilgilerin bütünlük içerisinde tutulabildiği ve birden fazla kullanıcıya aynı anda bilgiye erişim imkânının sağlanabildiği programlardır.

Veri tabanı yönetim sistemleri arasından MySQL, MsSQL, PostgreSQL, Oracle, Sybase, Berkeley DB, Firebird gibi örnekler verilirken programlanmaları için kullanılan diller arasından YSD (Yapılandırılmış Sorgu Dili), YD/YSD (Yordamsal Dil/YSD) verilebilir. Veri tabanı programcılığında VMD (Veri Muamele Dili) ve VTD (Veri Tanımlama Dili) ifadeleri kullanılır. Veri bütünlüğünün önemli olduğu GDYS sisteminde etkin ve üst düzey veritabanı programlamasına izin veren PostgreSQL kullanılmıştır.

1.1.7 Sürümleme

GDYS sisteminde veri tabanında kendisi ile bilgilerin tutulduğu dokümanların zaman içinde maruz kaldıkları değişimler yine veritabanında tutulur. Sürümleme bu bağlamda devreye girer ve GDYS sistemi için önemli hizmetlerden biridir. Sürüm denetimi bir doküman üzerinde zaman içinde yapılmış değişikliklerin her değişikliğe verilen bir numara ile takip edilebilmesini sağlayan bir hizmettir. Her dokümana her değişikliği için verilen numara ardışık olarak yapılan alfabetik eklerle birlikte oluşturulabilir. Sürümleme takım halinde yapılan geliştirme süreçlerinde takım üyelerinin yaptığı değişiklikler üstünde eşzamanlama sağladığı gibi zaman içinde yapılmış bir değişiklikler kümesine geri dönmeyi de sağlar.

Sürümleme, dosya kilitleme ve sürüm birleştirme bağlamında yapılabilir. Dosya kilitleme bir anda bir kişinin bir dosya üstünde değişiklik yapmasına izin verirken sürüm birleştirme ise aynı anda takım üyelerinin bir dosya üstünde değişiklik yapmasına izin verir ve yapılan değişikliklerin kurallar etrafında sisteme yazılmasını sağlar.

Sürümleme süreci içinde basitçe ilgilenilen dosyanın belirlenip sistemden indirilmesi, değişiklilerin yapılması ve tekrar sisteme konması işlemleri yapılır. Sistemlerin sürümleme için kullandıkları teknikler etrafında bunlara ek olarak çoklu birleştirme, dosya kitleme, ihraç etme ve ithal etme gibi işlemler de sağlanabilir.

Sürümleme desteği için harici bir hizmetten faydalanılmayan GDYS'de görece daha basit olarak gerçeklenebilen dosya kilitleme tekniğini kullanılmıştır.

Kuramsal temeller kısmı içinde GDYS sisteminin yapı taşları açıklanmış ve yapıtaşların oluşturulmasında kullanılmış teknikler özetlenmiştir. Takip eden kısmında ise GDYS sistemine benzer ticari ve açık sistemlere örnekler verilerek içyapıları ve işlevsellikleri anlatılmıştır.

1.1.8 Örnek sistemler

Doküman yönetimi alanında yapılmış ticari ve akademik çalışmalara örnekler arasında Knowledge Tree, Alfresco, Documentum, Main//Pyrus DMS, OpenKM verilebilir.

1.1.8.1 Knowledge Tree

Php dili ile yazılan ağ sunucu olarak Apache'i kullanan ve veri tabanı olarak ise MySQL'i kullanan ticari açık kaynak kodlu DYS'dir. Çoklu platform kurulum programı ile Windows ve Linux işletim sistemleri üstüne kurulabilir. Açık kaynak kodlu olması sayesinde kurumların kendi yapılarına uyarlayabilmesi sağlanır. GGL (Gnu Genel Lisansı) lisansı ile lisanslanır (http://www.knowledgetree.com., 2008).

1.1.8.2 Alfresco

Java teknolojisi ile geliştirilen açık kaynak kodlu, açık standart ve büyük ölçekli DYS'dir. Çoklu platform desteği ile Windows ve Linux işletim sistemi üstünde çalışabilir. Yüksek dereceli modüler olan Alfresco içerik ambarı, standart portal içeriği, Windows ve Linux üstünde sağladığı dosya sistemi bütünlüğü ara yüzü ile kullanımı oldukça esnek ve güçlüdür.

Doküman yönetimi, ağ içerik yönetimi, sürümleme, resim yönetimi, EJBG desteği, bütünleşik yayınlama, ara yüz ile uzak ambar erişimi, işakışı, çoklu dil desteği ve masaüstü programları ile bütünleşme gibi özelliklere sahiptir (http://www.alfresco.com., 2008).

1.1.8.3 Documentum

EMC firmasının mamulü olan Documentum büyük ölçekli ticari içerik yönetim sistemidir. İş yazıları, resim, tıbbi resim, posta, video, GİD (Genişletilebilir İşaretleme Dili) yazıları, ağ sayfaları gibi çok çeşitli yazı tiplerini yönetebilir. Documentum'un çekirdeği güvenli uyumluluk kuralları etrafında depolanabilen bir ambardır. Mantıksal olarak tek olduğu halde fiziksel olarak farklı dağıtık sunucu makineler üstünde yerleşebilir.

Documentum işlevselliği çeşitli kullanıcı ara yüzleri ve uygulama programlama ara yüzleri üstünden kullanılabilir. Documentum'un sunduğu hizmetler takımı ile doküman yönetimi, işbirlik, arama, içerik sınıflandırma, giriş yönetimi, iş süreci yönetimi, müşteri haberleşmesi yönetimi, ağ içeriği yönetimi, form işleme, bilgi hakları yönetimi ve arşivleme yapılabilir (http://www.documentum.com., 2006).

1.1.8.4 Main Pyrus DMS

Main Pyrus DMS açık kaynak kodlu, açık standart DYS'dir. Sunucu kısmı Linux işletim sistemi üstünde çalışır. İstemci kısmı ise Java ile yazılmış, Windows ve Linux üstünde çalışır. Orta ölçekli bir yazılım olup sunucu kısmı C++ ve Php ile yazılmıştır.

Doküman yönetimi, sürümleme, resim tarama desteği, değişiklikleri kayıtlama, çoklu platform desteği, dağıtık sunucu desteği, çoklu dil desteği ve OUBS (Ortak Unix Baskı Sistemi) ile bütünleşme gibi özelliklere sahiptir (http://www.mainpyrus.org., 2008).

1.1.8.5 OpenKM

Orta ölçekli, açık kaynak kodlu DYS'dir. GGL ile lisanslanmaktadır. OpenKM Java JGS teknolojisi ile JBOSS uygulama sunucusu üstünde geliştirilmiştir. Bu sayede Linux ve Windows işletim sistemleri üstünde kurulabilir. Veri tabanı olarak Oracle, Postgesql, MySQL gibi çeşitli ilişkisel veritabanlarını kullanır. Çoklu dil desteğine sahiptir. Büyük ölçekli indirme desteği, esnek arama desteği, kişi yetkilendirme ve eylemleri kayıt etme özelliği gibi özelliklere sahiptir.

GDYS sisteminin isterlerinin icra edilebilmesi için gereken temel kavram ve örnek sistemler anlatılmıştır. Materyal ve yöntem kısmında ise kullanılan alt sistemler, alt sistem yapılandırması ve bütünleştirilmesi çabaları ile birlikte ağ uygulamasının gerçeklenmesinde kullanılan tekniklerin nasıl kullanıldıkları anlatılmıştır (http://www.openkm.com., 2008).

2. MATERYAL VE YÖNTEM

Yayınların yeni çalışmalara kaynak olması, bilgi birikiminin sağlanması, gruplar halinde çalışmalara olanak verilmesi, bilirkişi tecrübelerinden faydalanılması, akademik yayınlarda standardın ve kalitenin artırılması, diğer araştırmaları desteklemesi amacı ile GDYS sistemi geliştirilmiştir. GDYS'de girdi bağlamında ilgilenilen materyal pdf, word, excel, düz metin dosya biçiminde olabilen akademik yayınlardır.

GDYS sisteminin geliştirilmesinde kullanılan yöntem icra etmekle yükümlü olduğu isterlerin gereksinim analizi ile belirlenmesi ile başlar. İsterler, sistem ve yazılım modelleme süreci içinde BMD dili diyagramlarına dönüşür ve sistemin sahip olacağı alt sistemler belirlenmiş olur. Alt sistem kurulumu, diğer sistemlerle olan bütünleşmesi ve gerekli yazılımların geliştirilmesi ile devam eden yöntem aşaması, tamamlanan sisteme ilişkin test vakalarının onanması ile sonuçlanır. Özet olarak anlatılan GDYS sistemi geliştirme yöntemi bölüm 2.1'de anlatılmış sistem analizi ve tasarım çabalarını ve bölüm 2.2'de anlatılmış yapısal gerçeklemeleri içerir.

2.1 GDYS Sistemi Analizi ve Tasarımı

GDYS sisteminin neye benzeyeceğine karar verildiği ve hangi teknolojilerin kullanılacağının ortaya konduğu sistem analizi ve tasarımı yazılım geliştirme süreçlerinin başlangıcıdır ve tüm mühendislik projeleri için ilk aşamasıdır. Sistemin başarısı ilk aşamada yapılan analiz ve tasarıma bağlıdır. Gereksinim analizi sonucu elde edilmiş isterler giriş bölümünde sunulmuştur.

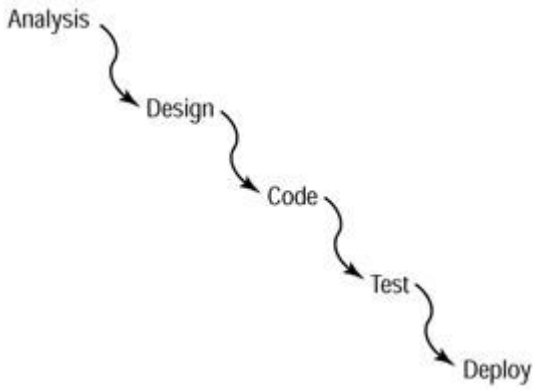
GDYS sisteminde nesne yönelimli yaklaşım, gerçek dünya sorunlarına yordamsal yaklaşımlara göre daha yakın ve gerçekçi çözüm sunmasından dolayı seçilmiştir. Nesne yönelimli yazılım geliştirme yaklaşımında temelde sarmalama (encapsulation), kalıtım (inheritance) ve çok biçimlik (polymorphism) gibi güçlü özellikler oturur. Yazılım geliştirme aşamasında yordamsal yöntemlerde yazılım içinde tutulacak veri yapılar içinde saklanır ve yazılım genel fonksiyonel bloklar halinde geliştirilirdi. Yordamsal

yöntem içinde fonksiyonel bloklar gerektiği yerlere kopyalanır ve değişiklik gerektiği an tüm kopyalar üstünde çalışılması gerekirdi.

Nesne yönelimli yaklaşımda ise yazılan kod parçaları yeniden kullanılabilir özelliğe sahiptir. Geliştirme süreci, birçok proje tamamlandıktan sonra daha önce yazılmış nesnelerin bir araya getirilip bunların üstünde iş kurallarının gerçeklenmesi halini alır. Temeli teşkil edecek işçatıları daha önceden geliştirilebileceği gibi üçüncü şahıslardan da alınabilir. Nesne yönelimli yaklaşım, sistem geliştirme sürecini hızlandırdığı gibi aynı zamanda standardı da sağlar.

Birbirinden farklı çeşitli yazılım geliştirme süreçleri mevcuttur. Örnek olarak şekil 2.1'deki çağlayan mimari, helezonik mimari ve şekil 2.2'deki iteratif mimari verilebilir. Tümünde ortak olarak bulunan yazılım geliştirme yaşam döngüsü ise aşağıdaki 5 adımı içerir:

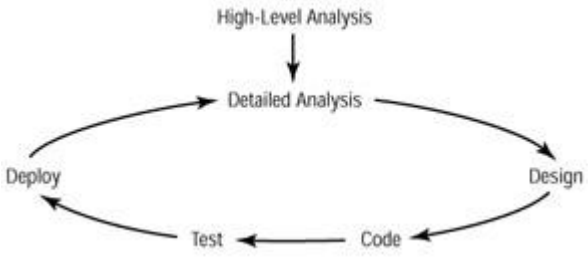
- Analiz,
- Tasarım,
- Geliştirme,
- Test,
- Kurulum.



Şekil 2.1 Çağlayan yazılım geliştirme mimarisi

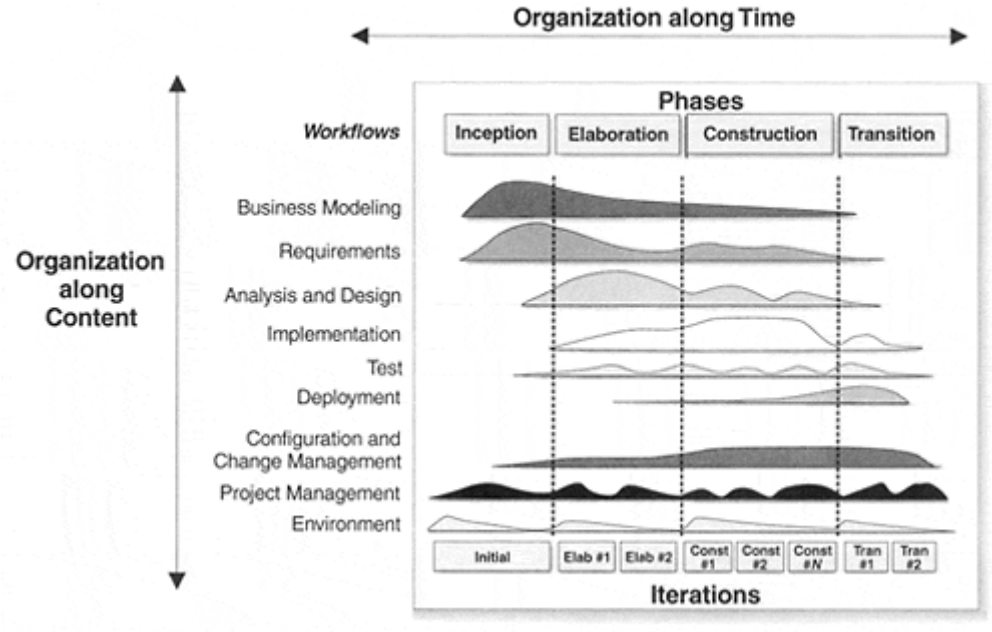
2.1.1 BS (Birleşik Süreçler) yazılım geliştirme süreci

Şekil 2.2'de görülen iteratif yazılım geliştirme süreci gerçek dünya uygulamaları için daha uygundur. Yazılım geliştirme sürecinin herhangi aşamasında tüm gereksinimleri belirleyemeyeceğimiz gibi zaman içinde değişen sistem gereksinimlerine sistemin ayak uydurabilmesi gerekir. Belirtilen eksiklikleri karşılayacak yöntemlerden biri ise Jacobsen (1990) tarafından öne sürülmüş BS'dir.



Şekil 2.2 İteratif yazılım geliştirme mimarisi

BS yazılım geliştirme sürecini şekil 2.3'de görüldüğü gibi 4 ana aşamaya böler ve bunlar içinde yazılım yaşam döngüsü her aşamanın bitişini belirleyen kıstaslar sonuçlanıncaya kadar tekrarlanır (Kruchten 2000).



Şekil 2.3 BS'nin zaman içinde her bir adımında gerçeklenen temel süreçlerin dağılımı

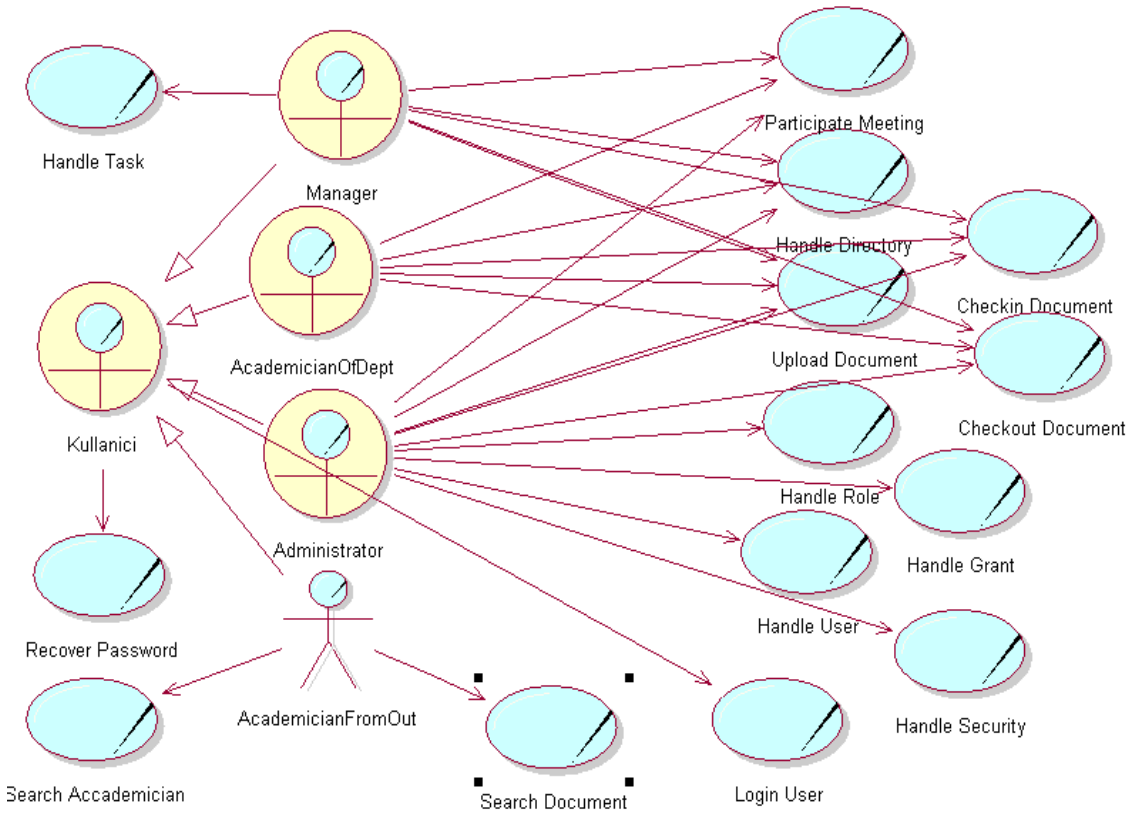
Başlangıç, detaylandırma, yapılandırma ve geçiş aşamalarının her birinde temel yazılım yaşam döngüsü işlevleri gerçeklenir. Şekil 2.3'de görüldüğü gibi başlangıç evresinde tüm gereksinimlerin anlaşılmasına ve geliştirme çabalarının kapsamının belirlenilmesine odaklanılır. Detaylandırma evresinde gereksinimlere odaklanılarak yazılım tasarımı ve uygulaması, mimari ilk örneği ile desteklenilir; çeşitli çözümler denenerek belirli teknik riskler önlenir; belirli araç ve tekniklerin kullanılması öğrenilir. Bir sonraki evrenin temelini oluşturacak mimari ilk örnek üretilir. Yapılandırma evresinde odak tasarım ve gerçeklemedir. Tasarım ilk örneği bu aşamada evrilir ve ilk işlevsel ürün vücut bulur. Geçiş evresinde sistemin amaçları karşılayacak gerçek ve doğru kaliteye sahip olup olmadığından emin olma çabaları devam eder. Hatalar çözülür; kullanıcılar eğitilir; özellikler ayarlanır; kayıp elemanlar eklenir. Kullanıma hazır sonlanmış ürün teslim edilir. Özet olarak ele alınmış iteratif yöntem aşamaları takip eden bölüm 2.1.1.1, 2.1.1.2, 2.1.1.3, 2.1.1.4, 2.1.1.5'de detaylı olarak ele alınarak, GDYS açısından yapılanlar anlatılmıştır.

2.1.1.1 Başlangıç aşaması

Başlangıç aşaması, geliştirilecek sistemin ne yapacağı, sınırlarının ne olduğu, içyapısında hangi rollerin olduğu, dış dünya ile nasıl ve kimlerle etkileşeceği, uygun olup olmadığı gibi sorulara cevapların arandığı aşamadır. İş kullanım vakası (Bussiness use-case) diyagramı, Etkinlik (Activity) diyagramı ile modelleme başlar ve ilerleme plan (iteration plan)'ının hazırlanması ile son bulur. İlerleme planı hangi adımda hangi iş kullanım vakasının gerçekleneceğinin belirlendiği plandır. GDYS sistemine ait iş kullanım vakaları şekil 2.4'deki diyagramda sunulmuştur. Sistemdeki roller, sistemin sağladığı temel işlevler ve sistem dışı etkileşilen kişiler görülür. Takip eden bölümde iş kullanım vakaları ele alınmıştır.

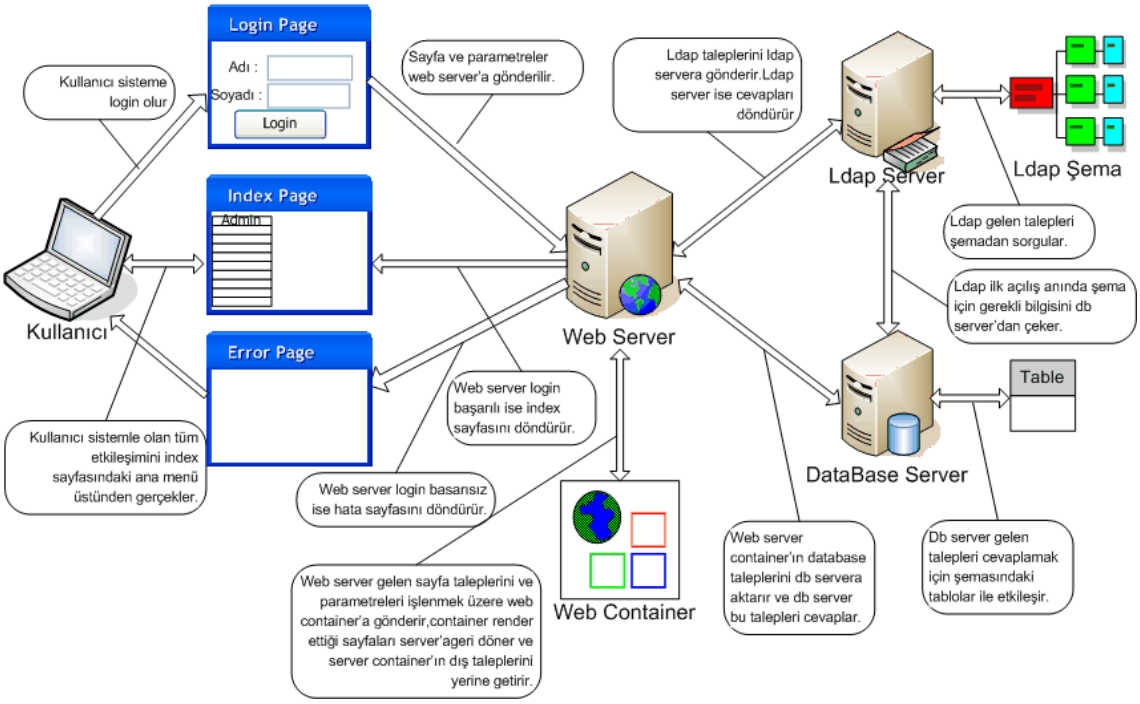
2.1.1.1.1 İş kullanım vakaları

GDYS sistemi isterleri ilk olarak iş kullanım vakalarına dönüştürülür. İş kullanım vakası, sistemin dış kullanıcılar için anlamlı olan sağlayabildiği genel işlevleri gösterir; sistem kullanım vakaları ve etkinlik diyagramları ile detaylandırılırlar (Şekil 2.4).



Şekil 2.4 GDYS sistemi iş kullanım vakaları diyagramı

Sistemin çalışma mantığı merkezde ağ sunucusu üstünde kurulmuştur. Sistemle etkileşim tamamen Tomcat ağ sunucusu üstündeki tanımlı modüllerce yönetilir. Tanımlanmış modüller gerekli yardımcı sunucular ile etkileşerek temel işlevler yerine getirilir. Genel etkileşim süreci şekil 2.5'de görülebilir.



Şekil 2.5 Genel etkileşim süreci

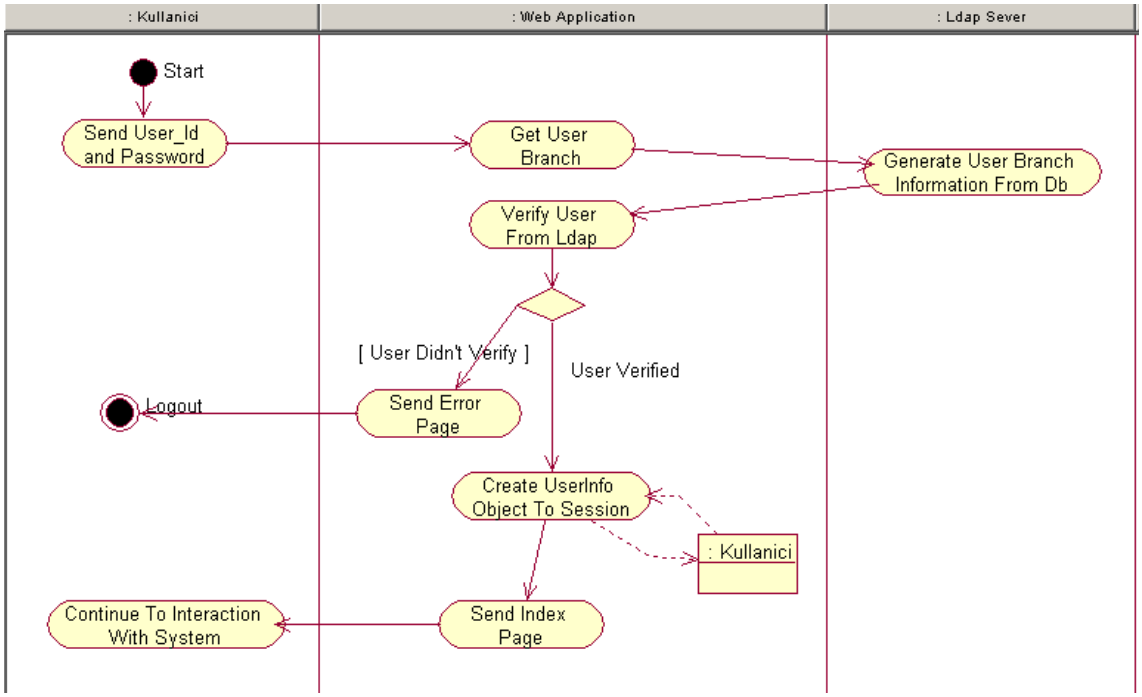
Şekil 2.4'de görüldüğü üzere idareci, bölüm akademisyeni ve yönetici rollerinin doküman ile ilgili iş kullanım vakaları yanında yalnızca kendi rollerinin yapabileceği iş kullanım vakalarına da sahiptirler. Örneğin idareci rolü görev yönetimi yanında doküman, dizin yönetimi gibi iş kullanım vakaları kullanabilir.

GDYS sisteminde önemli olan kullanıcı girişi, doküman yükleme ve kullanıcı ekleme iş kullanım vakaları devam eden bölümlerde detaylı olarak incelenmiştir.

2.1.1.1.1.1 Kullanıcı girişi iş kullanım vakası

Sistemle etkileşim ilk olarak giriş (login) sayfasında kimlik denetimi ile başlar. Sisteme girmek için ağ tarayıcısının bky kısmına http://<TomcatGDYSSunucusu>/dcm girildiğinde karşısına giris.jsf sayfası gelir. Ad ve şifre girilip giriş düğmesi tıklanıldığında AMTP talebi Tomcat ağ sunucusu içindeki ağ kabına (container) aktarılır. Ağ kabı, kullanıcıya ait bilgileri HDEP sunucusunun yönetimindeki ağaç yapısı içindeki kullanıcının tanımlandığı daldan çeker. Talep ile gelen şifre, HDEP'den

alınan şifre ile karşılaştırılır ve karşılaştırma sonucu doğru ise indis.jsf sayfası, yanlış ise hata.jsf sayfası Tomcat sunucu tarafından geri döndürülür. Kimlik denetimi doğru sonuçlanır ise ağ kabı oturum (session) erişim düzeyine kullanıcıya ait HDEP'den gelen grup bilgisini kayıt eder. Ağ kabı grup bilgisini JYKS standardı ile sağlanan yetkilendirme içinde kullanır. Yetkilendirme aşamasında her kullanıcının sahip olduğu rol bilgisi GDYS sistemi tarafından kimlik denetimi sonrası indis.jsf sayfasının sunucu üstünde ilk oluşturulduğu anda elde edilerek oturum erişim düzeyine kullaniciBilgi nesnesi içinde yazılır. Rol bilgisi kullanıcının her talep ettiği eylemde kontrol edilir ve eylemi gerçekleme yetkisi olduğu anlarda eylem gerçeklenir. Şekil 2.6'da kullanıcı girişi için etkinlik diyagramı görülür.

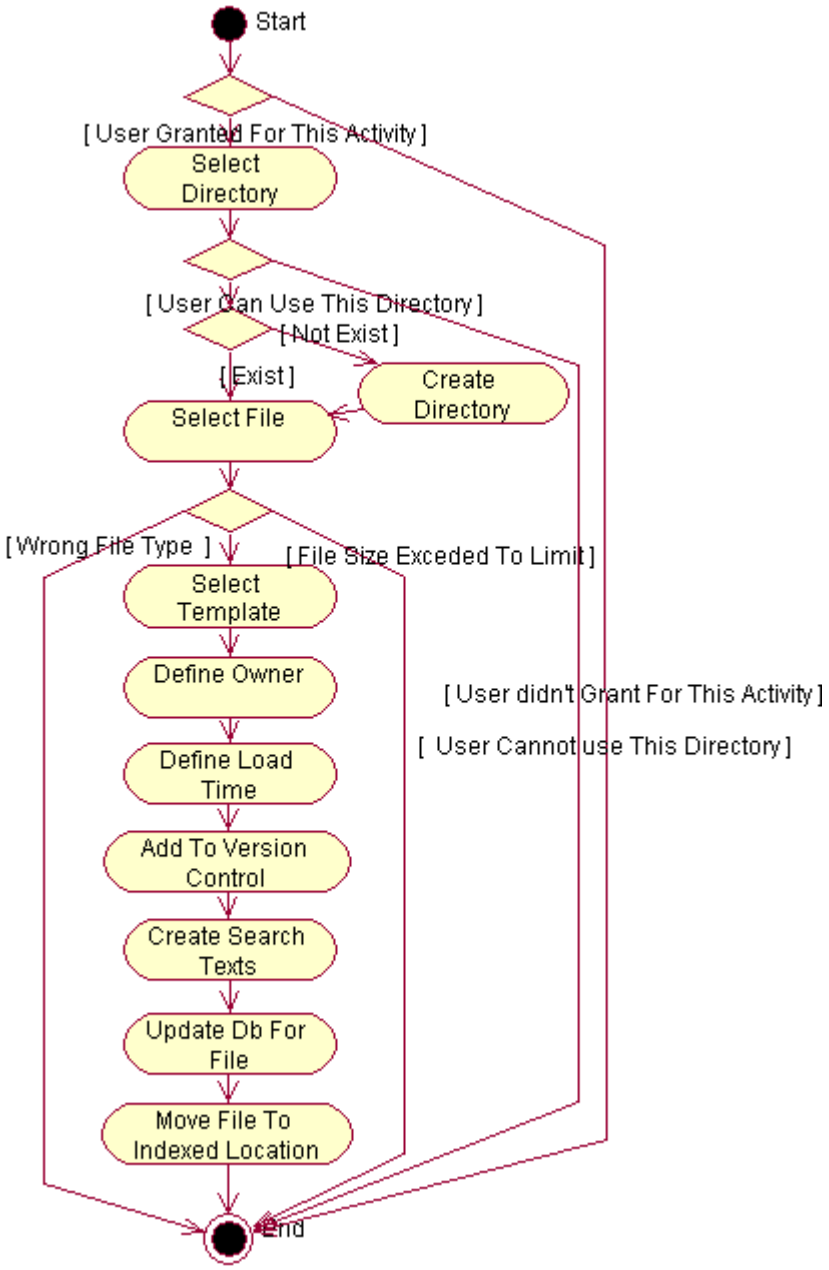


Şekil 2.6 Kullanıcı girişi etkinlik diyagramı

2.1.1.1.1.2 Doküman yükleme iş kullanım vakası

Yükleme etkinlik diyagramında dosya yükleme işlevi gerçeklenirken aktif kullanıcının bu kullanım vakası gerçekleme yetkisi kontrol edilir. Dosyanın yükleneceği dizinin mevcut olup olmadığı ve aktif kullanıcının bu dizine yazma yetkisi kontrol edilir.

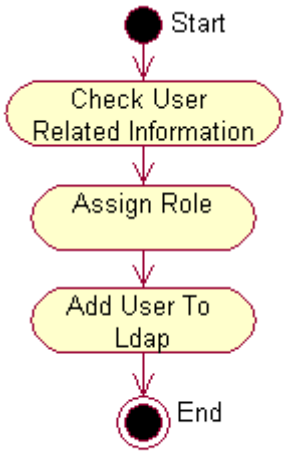
Ardından dosya tipi ve boyutu kontrol edilir. Kontroller sonrası dosya geçerli bir dosya ise arama için içerdiği kelimeler indislenir, sürüm kontrolü altına alınır ve dosya içeriği sunucu üstünde indislenmiş yere yerleştirilir.



Şekil 2.7 Doküman yükleme etkinlik diyagramı

2.1.1.1.1.3 Kullanıcı ekleme iş kullanım vakası

Şekil 2.8'de görüldüğü gibi kullanıcıya ait bilgilerin tutarlılığı kontrol edilip, rol ilişkilendirmesi yapıldıktan sonra kullanıcı girişi kullanım vakası için HDEP şemasında bu kullanıcının bulunacağı dal veri tabanı tablolarında oluşturulur.



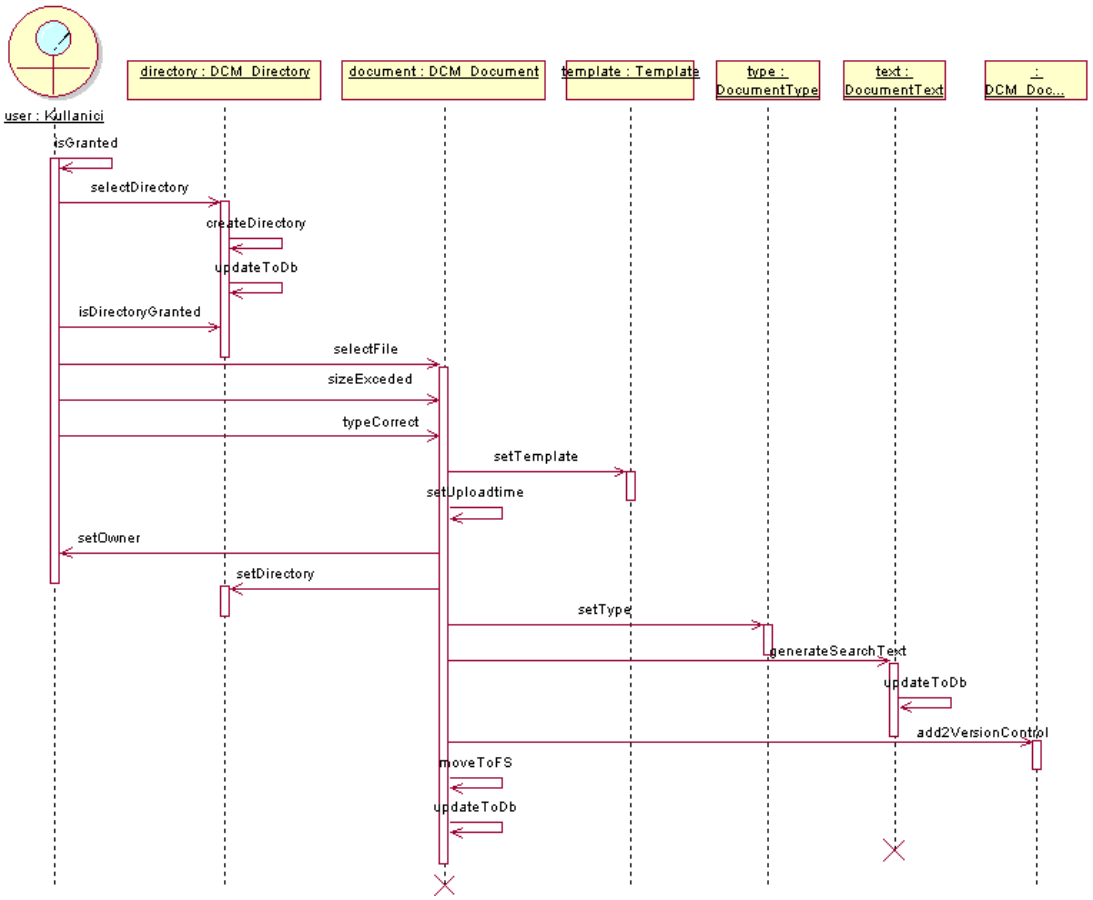
Şekil 2.8 Kullanıcı ekleme etkinlik diyagramı

2.1.1.3 Detaylandırma aşaması

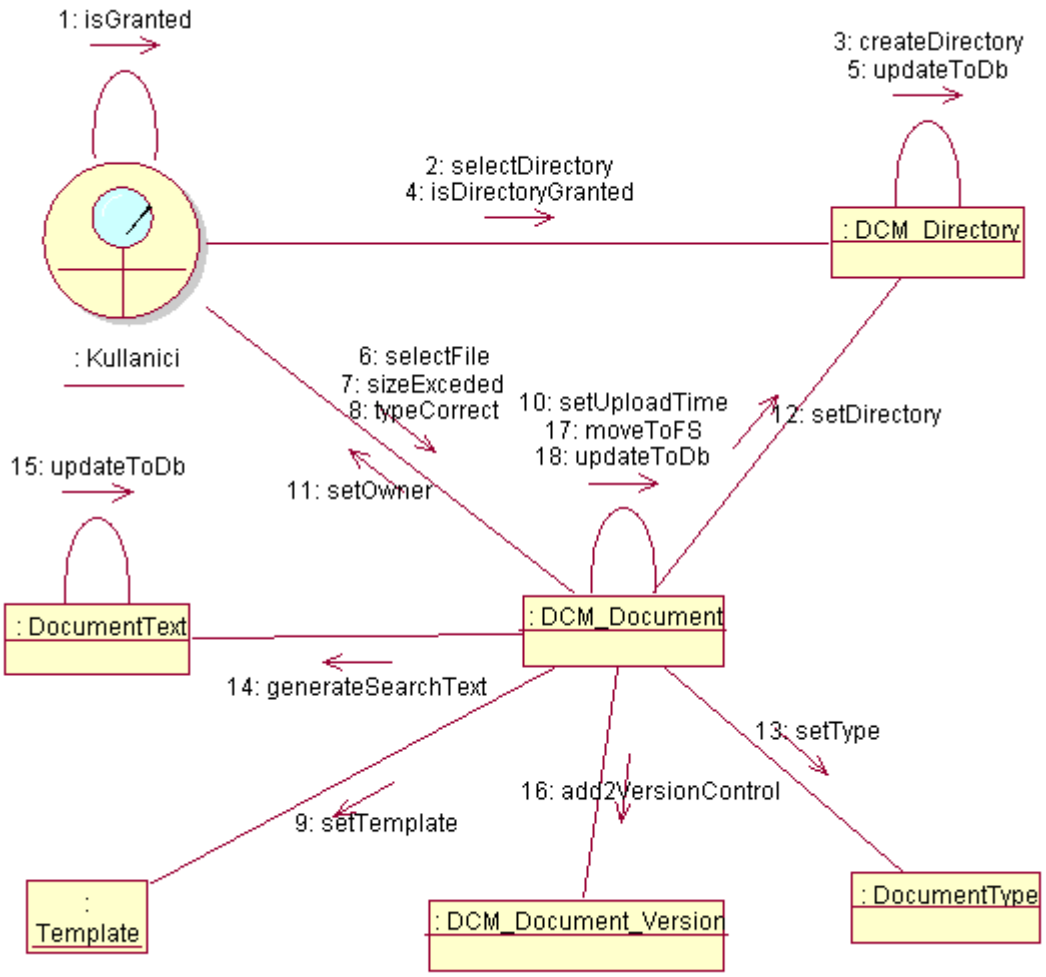
Detaylandırma aşamasında ise mimari yapının ne olacağı, başlangıç aşamasındaki iş kullanım vakalarının kullanım vakaları ile detaylandırıldığı aşamadır. Bu kısımda iş kullanım vakalarını detaylandıran kullanım vakaları diyagramları ve kullanım vakaları içindeki işleme akışını gösteren sekans (sequence) diyagramı, işbirlik (collaboration) diyagramı ve işleme akışındaki nesnelerin durumlarındaki değişiklikleri gösteren durum (state) diyagramları çizilir. Sistemin tüm gereksinimleri SGT (Sistem Gereksinim Tanımlamaları) denen tanımlama içinde içerilir (Booch *et al.* 1998). Test vakalarının hazırlandığı, gereksinimlerin elden geçirildiği bu aşamanın bitişi yüksek risk taşıyan ve mimari olarak önemli kullanım vakalarının tam olarak detaylandığı, başlangıç sınıf diyagramının tamamlandığı zamandır.

Sekans diyagramı kullanım vakası içindeki işleme akışını detaylı olarak gösterir. Kullanım vakasının gerçeklenebilmesi için nesnelerin birbirleri ile olan

mesajlaşmalarını zaman düzleminde gösterir. İşbirlik diyagramı ise daha çok nesneler üstündeki iş dağılımını gösterir. Yıldız dağılımı ile mesajlaşmaların ortasında kalan bir nesne mevcut ise bunun üstünde fazla miktarda iş yükü mevcuttur ve bu iş yükü uygun nesneler arasında dağıtılmalıdır (Khawar and Umrysh 2002). Yükleme kullanım vakasına ait sekans ve işbirlik diyagramları şekil 2.9 - 2.10'da sunulmuştur. Kullanım vakasını başlatan kullanici aktörü ilk önce DYS_Dizin örneği olan nesne ile etkileşir ve DYS_Doküman örneği olan nesne için gerekli tip, arama metni, sürümleme bilgileri toplanır.



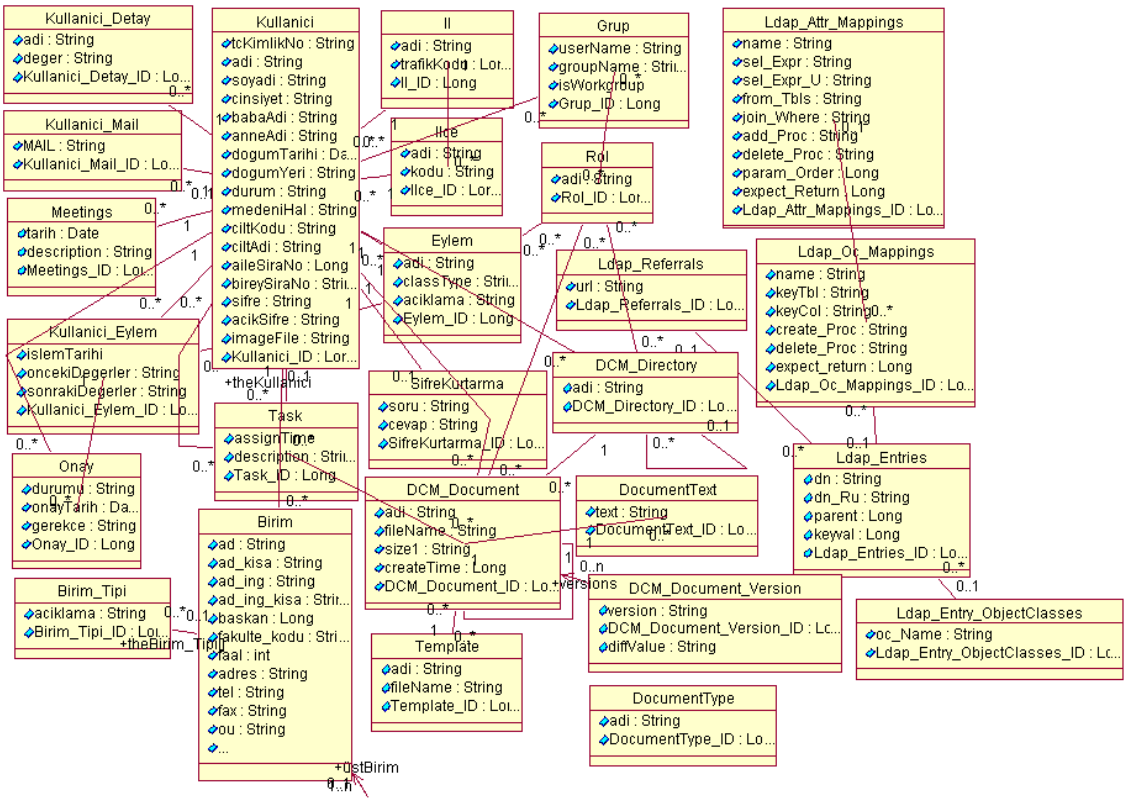
Şekil 2.9 Yükleme kullanım vakasına ait sekans Diyagramı



Şekil 2.10 Yükleme kullanım vakasına ait işbirlik diyagramı

Sınıf diyagramı yazılımın tutacağı bilginin yapısını gösterdiği gibi bu yapıların hiyerarşik durumunu ve birbirleri ile olan ilişkilerini de gösterir. Sınıfların birbirleri ile ilişkilendirme (aggregation), toparlama (association), kompozisyon (composition) ve kalıcı (persistent) olmak üzere 4 tipte ilişkileri mevcuttur (Booch *et al.* 1999). GDYS sisteminin statik yapısını gösteren sınıf diyagramı şekil 2.11'de sunulmuştur. Kullanici sınıfı ve DYS_Doküman sınıfı tüm sistemin merkezindeki önemli iki sınıftır. Yetkilendirme ve günce (log) tutma, Rol ve Kullanici_Eylem sınıfı üstünden yapılır. Bunlar dışında kalan sınıflar tanım sınıfları ve ilişkilendirme (aggreation) sınıflarıdır. Kullanici sınıfı, giriş sürecinden itibaren Tomcat sunucusunun oturum düzeyinde tutulur. Kullanıcının sistem üstünde yapacağı her etkileşim Kullanici_Eylem sınıfı

üstünden denetlenir ve güncesi tutulur. Sistem üstünde yapılacak her kullanım vakası için aktif kullanıcının yetkileri rol sınıfı üstünden belirlenirken DYS_Doküman üstünde işlem yapabilecek kişiler ise DYS_Doküman örneğine atanmış roller üstünden belirlenir.

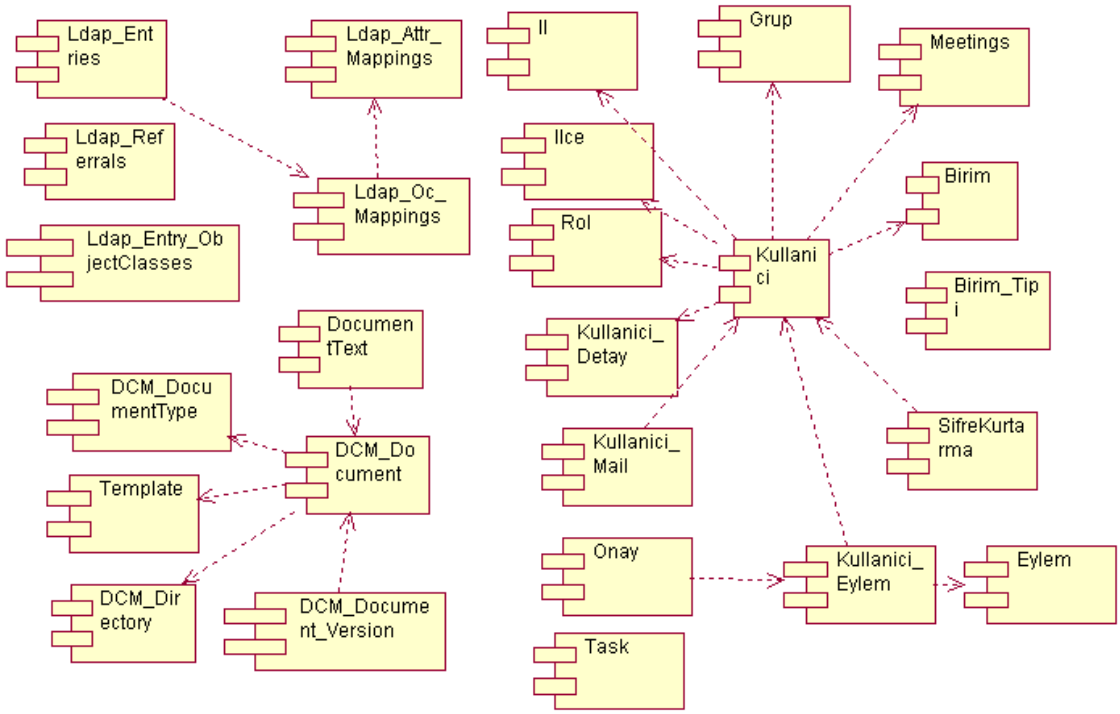


Şekil 2.11 GDYS sistemi sınıf diyagramı

2.1.1.4 Yapılandırma aşaması

GDYS sisteminin geliştiriminde nesne yönelimli dil olan Java dili kullanılmıştır. Yapılandırma aşamasında sistemin detaylandırma aşamasında tamamlanmış tasarımı temel olarak alınır ve sistemin geri kalan gereksinimleri belirlenir, yazılım geliştirilir ve test edilir. Bu aşamada kod üretimi için sınıfların çalışma ve derleme zamanı bağımlılıklarını gösteren Bileşen (Component) diyagramı tamamlanır. Bileşen diyagramı kodlama için seçilen dile bağımlı olarak oluşturulur (Boggs and Boggs 2002). Sistemin sahip olduğu çalıştırılabilir uygulamaları ki her biri aslında sistemin alt sistemidir ve kütüphaneleri içinde hangi sınıfların oturacağı belirlenir. BS yinelemeli bir

yöntem olduğu için hem test vakaları geliştirilirken hem de ortaya çıkan tasarım ve gereksinimle ilgili düzeltmeler kullanım vakası, sekans, işbirlik, sınıf diyagramlarına güncellenir. Yapılandırma aşamasındaki en önemli görevlerden biri geliştirme takımı dışındaki bir grup tarafından yapılan kod inceleme ve kalite garanti incelemesidir. Kod inceleme ile standart ve tasarım kıstasların karşılandığından emin olunur ve fonksiyonellik araştırılır. Yapılandırma aşaması yazılımın tamamlanıp ve test edildiği zaman biter. GDYS sistemine ait şekil 2.12'de görülen bileşen diyagramında sınıfların birbiri ile nasıl bir bağımlılığa sahip olduğu görülür. Örneğin Kullanici sınıfının derlenebilmesi için birim, grup, rol gibi sınıfların derlenmiş ve Kullanici sınıfı tarafından kullanılabilir olması gerekir. Aynı mantıkla Kullanici_Eylem sınıfının derlenebilmesi için Kullanici sınıfının derlenmiş olması gerekir.

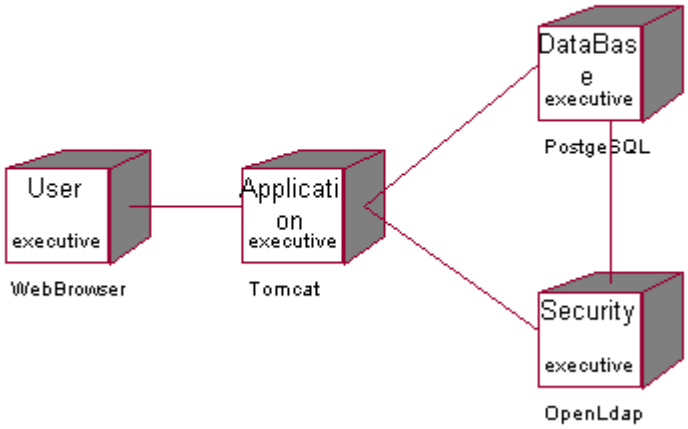


Şekil 2.12 GDYS sistemi bileşen diyagramı

2.1.1.5 Geçiş aşaması

Geçiş aşamasında ise yazılım son kullanıcı ortamına kurulur. Son kullanıcı tarafından dönen isteklere göre bu ana kadar hazırlanmış diyagramlar ve yazılım güncellenir. Kullanıcı eğitimleri düzenlenir ve teslimat ile ilgili olarak kabul testleri yapılır. Sistemin

fiziksel yapısını gösteren Kurulum (Deployment) diyagramı oluşturulur. Kurulum diyagramı, sistemin şebekedeki fiziksel planını göstererek hangi yazılım bileşeninin hangi cihaz üstünde oturacağını gösterir. Bileşen diyagramındaki alt sistemler bu diyagramdaki işlemciler üstünde çalışan derlenmiş yazılımlara dönüşür. Şekil 2.13'de görüleceği üzere en önde kullanıcı makinesi onu karşılayan uygulama sunucusu (Tomcat) ve uygulama sunucusuna hizmet veren veritabanı sunucusu ve HDEP sunucusu arka kısımda bulunur.



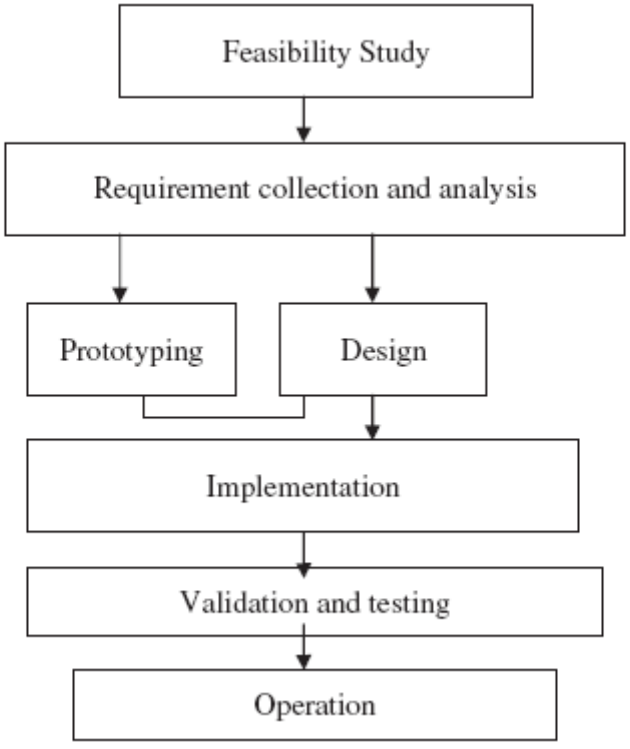
Şekil 2.13 GDYS sistemi kurulum diyagramı

2.1.2 GDYS sistemi veri tabanı tasarımı

Buraya kadar GDYS sisteminin uygulama sunucusu üstünde çalışacak yazılımın BS tekniği ile iş süreçleri yapısal analizi, tasarımı ve gerçeklemesi yapılmıştır. Yapılandırma aşamasında oluşturduğumuz sınıf diyagramı hafızada tutulacak verinin yapısını gösterir. Tüm organizasyonel sistemlerde olduğu gibi zaman içinde yapılacak işlemler için verilerin saklanması gerekir ve bu amaçla veritabanları kullanılır. Nasıl hafızada sakladığımız verinin hafızayı verimli kullanabilmesi için gerekli yapısal teknikler kullanılırsa aynı şekilde veritabanın veriyi verimli saklaması için farklı teknikler kullanmamız gerekir. Bu teknikler içinde normalizasyon başı çeker.

Veritabanı tasarım süreci ilişkili veriyi bütünleştirir. Veritabanı tasarım süreci, ferdi bileşenler arasındaki ilişkinin belirlenmesi ve doğru fonksiyonalitenin bakımından

dolayı oldukça karmaşıktır. Bileşenler arasında çoktan çoğa ilişkiler mevcutsa bu karmaşıklık daha da artar. Veritabanı tasarımı için yapılması gereken aşamalar şekil 2.14'de görülmektedir. Bu tasarım aşamaları genel yazılım yaşam döngüsü adımlarını takip eder (Sumathi and Esakkirajan 2007). Uygunluk çalışması (Feasibility study), tasarlanan veri tabanının amacının net olarak belirtildiği aşamadır. Gereksinim toplama ve analiz (Requirement collection and analysis), hangi verinin depolanacağı, depolanacak verinin nasıl depolanacağını ve veri kısımları arasındaki ilişkinin belirlendiği aşamadır. İlk örnekleme ve tasarım (Prototyping and design) aşaması, iş gereksinimlerini karşılamak için verinin kararlı ve uygun bir formda örgütlenmesidir. Veritabanı tasarım aşamasının kavramsal tasarım (conceptual design), mantıksal tasarım ve fiziksel tasarım (logical design and physical design) olan 3 fazı mevcuttur. Uygulama (Implementation) aşaması, yeni veritabanı içeriğinin kurulumu ve veritabanı süreçleri için kod geliştirimini içerir.



Şekil 2.14 Veritabanı tasarımı genel aşamaları

Veritabanı tasarımında göz önünde bulundurulması gereken verimlilik (efficiency), bütünlük (integrity), şahsiyet (privacy), güvenlik (security), uygulanabilirlik

(implementability), esneklik (flexibility) gibi faktörler vardır. Verimlilik, VTYS'nin yazılımının üstünde koştuğu donanımı verimli kullanmasıdır. Bütünlük, VTYS'nin ve donanımın alt seviyede veriyi bozmaması ve üst seviyede yazılımın ilgilendiği gerçek dünya kısmını doğru biçimde sunabilmesidir. Şahsiyet, yetkisiz girişlere izin vermemesini öne sürer. Güvenlik, veritabanının donanım, yazılım ve yetkisiz giriş tarafından meydana gelecek fiziksel bozulmalara karşı güvenli olmasını getirir. Uygulanabilirlik, uygulama yazılımının veritabanı ile verimli etkileşimi göz önünde bulundurularak kavramsal modelin mantıksal modele rahatlıkla eşlenebilmesi gerekliliğini öne sürer. Esneklik, veritabanı tasarımının gelecek gereksinimleri karşılamak üzere kolay olarak güncellenebilir yapıda olmasını getirir.

Veritabanı tasarımı için tasarım gereçlerinin kullanımı hata payını azalttığı gibi VTD ifadeleri ile yapılacak kodlama süresini minimize ederek tasarım süresini azaltır. Üretkenliği artıran çeşitli tasarım araçlarından aşağıdaki örnekler verilmiştir:

- CASE Studio 2 (http://www.casestudio.com/enu/default.aspx., 2008)
- Design for Databases V3 (http://www.datanamic.com/dezign., 2008)
- DBDesigner4 (http://fabforce.net/dbdesigner4., 2008)
- ER/Studio (http://www.embarcadero.com/products/erstudio/index.html., 2008)
- Happy Fish Database Designer (http://www.embarcadero.com/products/erstudio/index.html., 2008)
- Oracle Designer 2000 (http://www.Oracle.com/technology/products/designer/index.html., 2008)
- QDesigner (http://www.quest.com/QDesigner., 2008)

Veri tabanı tasarımında karşılaşılan kavramsal modelleme sorunları takip eden bölümler içinde açıklanmıştır. Bunlardan ilki fazlalık ve veri aykırılığıdır.

2.1.2.1 Fazlalık ve veri aykırılığı

Veri fazlalığı, aynı verinin birçok kez depolanması anlamına gelir ki veri kaybına uğramadan veri fazlalığı ortadan kaldırılabilir. Veri fazlalığı veri bozukluğuna götürür.

Üç tip veri bozukluğu mevcuttur: Ekleme aykırılıkları (Insertion Anomalies), silme aykırılıkları (Deletion Anomalies), güncelleme aykırılıkları (Updation Anomalies). Fazlalık normal veritabanı işlemleri boyunca problemlere sebep olabilir. Örneğin veri, veritabanına girildiğinde o verinin fazlalık sürümünün olduğu her yerde tekrarlanması gerekir.

Tablo bozukluğu öyle bir yapıdır ki normal veritabanı operasyonu bilgi kaybı olmaksızın gerçeklenemez. Örneğin birim bilgisi ve çalışan bilgisi aynı tablo içinde birlikte tutulursa ekleme aykırılığı ile çalışan kaydı girmeden birim kaydı giremeyiz. Aynı tablo içinde bir kişinin çalıştığı birim adını aynı birimde çalışan diğer kişinin birim adını değiştirmeksizin değiştirebilirsiniz ki bu güncelleme aykırılığıdır. O an için bir birimde çalışan tek kişi varken o kişinin kaydını silersek bölüme ait olan bilgiyi kaybetmiş oluruz ki bu da silme aykırılığıdır. Ek olarak bir birincil anahtar (primary key) için birden fazla değer içeren alanlar var ise tekrarlanan grup (repeating group) örneğidir ki ilişkisel veritabanlarında izin verilmez. Bir başka deyişle, birincil anahtara sahip tabloda bir kayıt için bir alanda birden fazla değer bulunması, gereksiz boşlukların oluşmasına sebep olur.

2.1.2.2 Fonksiyonel bağımlılık

Fonksiyonel bağımlılık, etkileşen özellikler arasındaki ilişkilerdir. Fonksiyonel bağımlılık özellikler arasındaki kısıtları açıklamak için resmi mekanizma sağlar. Eğer A özelliği B özelliğine fonksiyonel bağımlı ise B'nin her örneği için ona karşılık gelen A'nın değeri bilinecektir. Fonksiyonel bağımlılık özelliklerin diğerleri ile nasıl ilişkili olduğunu belirlemeye yardımcı olur.

2.1.2.3 Fonksiyonel bağımlılık notasyonu

A → B ile gösterilen fonksiyonel bağımlılık ve anlamı ise :

- A, B'yi belirler
- B A'ya fonksiyonel olarak bağımlıdır

- A determinant olarak adlandırılır B ise determinant nesnesi olarak adlandırılır.

Eğer B $\subseteq$ A ise fonksiyonel bağımlılık A $\rightarrow$ B önemsizdir.

2.1.2.4 Bileşik determinant

Bir özelliği belirlemek için birden fazla özellik gerekirse determinantlar bileşik determinant olarak adlandırılır.

2.1.2.5 Tam fonksiyonel bağımlılık

Eğer bir özelliğin diğerine fonksiyonel bağımlılığı mevcut fakat determinantın herhangi bir alt kümesine fonksiyonel bağımlılığı mevcut değil ise bu duruma tam fonksiyonel bağımlılık denir.

2.1.2.6 Kısmi fonksiyonel bağımlılık

Nesnesini tanımlamak için birleşik determinantın bir alt kümesinin kullanımı gerekli ise bu durumdaki fonksiyonel bağımlılığa kısmi fonksiyonel bağımlılık denir.

2.1.2.7 Geçişsel bağımlılık

Eğer ara fonksiyonel bağımlılık mevcutsa bir fonksiyonel bağımlılıkta buna geçişsel bağımlılık denir. Notasyonu ise A$\rightarrow$ B, B$\rightarrow$ C, ve eğer A$\rightarrow$ C varsa bu geçişsel bağımlılık olarak A$\rightarrow$ B$\rightarrow$C şeklinde gösterilir.

2.1.2.8 Fonksiyonel bağımlılık sonuç aksiyomları (Armstrong aksiyomları)

2.1.2.8.1 Yansıma aksiyomu

Eğer Y $\subseteq$ X ise X $\rightarrow$ Y ‘dır. Bir özellik kümesinin kendisi herhangi bir alt kümesini fonksiyonel olarak belirler.

2.1.2.8.2 Artırma aksiyomu

Eğer X → Y ve Z ise R tablosunun bir alt kümesi ise XZ → YZ 'dir. Bu aksiyoma göre bir veya daha fazla özellik ile sol veya her iki taraf artırılabilir.

2.1.2.8.3 Geçişlilik aksiyomu

Eğer X → Y , Y → Z varsa X → *Z* vardır. Bir özellik ikinci özelliği belirlerse ve ikincide bir diğer özelliği tek olarak belirlerse ilk özellik son özelliği belirler.

2.1.2.8.4 Yalancı (Pseudo) geçişlilik aksiyomu

Eğer X → Y , YW → Z varsa XW → Z vardır. W'nin boş küme olma durumu yalancı geçişliliğin özel halidir ve geçişlilik özelliğine denk gelir.

2.1.2.8.5 Birleşme aksiyomu

Eğer X→Y ve X→Z ise X→YZ 'dir. Eğer iki fonksiyonel bağımlılık aynı determinant ile mevcut ise yeni bir fonksiyonel bağımlılık determinantın korunup sağ tarafların birleşimi alınarak oluşturulabilir.

2.1.2.8.6 Ayrışma aksiyomu

Eğer X→YZ ise X→Z ve X→Y 'dir. Herhangi bir fonksiyonel bağımlılığın determinantı sağ tarafı oluşturan özelliklerden bir veya daha fazla özelliğin birleşimi fonksiyonel olarak belirler.

2.1.2.9 Fonksiyonel bağımlılığın kapalı kümesi

Bir R ilişkisi ile tanımlı F fonksiyonel bağımlılıkları kümesi için $F_{+,}$ F'in kapalı kümesidir. $F_{+,}$ F tarafından mantıksal olarak ima edilen tüm fonksiyonel bağımlılıklar kümesidir. F_{+}'ın tümünün hesaplanması için Armstrong kuralları yeterlidir. Armstrong

kuralları tekrarlı olarak uygulanırsa F_+'deki tüm fonksiyonel bağımlılıklar bulunur. Kapalı özellik kümesi bir F fonksiyonel bağımlılığı altında A özellikler kümesinin kapalı kümesi A_+, öyle B özellikler kümesi olur ki bu ise A'dan sonuç kurallarını F'in fonksiyonel bağımlılıklarına uygulayarak türetilir. A_+ her zaman boş olmayan kümedir. A→ A yansıma aksiyomuna göre.

Bir A özelliğinin F fonksiyonel bağımlılıkları kümesi altında kapalı kümesi olan A_+'nın hesaplanması için bir algoritma aşağıda sunulmuştur.

••

I=0; A[0]=A;
REPEAT
    I=I+1;
    A[I] = A[I − 1];
    FOR ALL Z->W in F
    IF $Z \subseteq [I]$
    THEN $A[I] = A[I] \cup$ W;
    END FOR
UNTIL A[I] = A[I − 1];
RETURN A+ = A[I];

•

Yukarıdaki algoritmada I bir tam sayıdır. Algoritmada A → A[I] ve $Z \subseteq$ A[I] ile F içinde Z → W'nin bulunmasından sonra A[I] YZ gibi gösterilebilir (Y = A[I] − Z). A → A[I]'ı A → YZ gibi yazabiliriz. F Z → W'ı içerdiği için toplama kuralını uygulayarak A → YZW olarak veya tüme varım hipotezi korunarak $A \rightarrow A[I] \cup W$ olarak sonuçlandırılabilir.

2.1.2.10 Kapsama

F ve G aynı ilişkisel şema üstünde tanımlanmış iki fonksiyonel bağımlılık kümesini göstersin. Eğer F+ = G+ ise F ve G eşdeğerdir. Her ne zaman F+ = G+ ise F G'yi kapsar denir. Aynı ilişkisel şema üstünde tanımlanmış iki fonksiyonel bağımlılık kümesi

F ve G ise ve G F'i kapsar ise uygun G'nin olmayan kümesi H için H+ = G+ varsa G F'in fazlalık olmayan kapsamı denir.

Bu kısımda veri bozuklukları ve veri tabanı içindeki tabloları bir küme olarak ele alırsak kümeler içindeki elemanlar arasında ilişkileri tanımlamak için gerekli bilimsel alt yapıdan bahsedilmiştir. Normalizasyon ile bölüm 2.1.2.2'deki fonksiyonel bağımlılık kısmında anlatılmış ilişki türlerinden faydalanarak veri tabanı daha verimli hale getirilebilir.

2.1.2.11 Normalizasyon

Normalizasyon veritabanı içindeki verinin örgütlenmesi sürecidir (Connolly *et al*. 1999). Tabloların yaratılması ve tablolar arasındaki ilişkilerin fazlalık ve tutarsız bağımlılıkların ortadan kaldırılması ile veri tabanını daha esnek yapmak ve veriyi korumak için tasarlanmış kurallara uygun olarak veri tabanının kurulması görevleri normalizasyonca içerilir. Fazlalık, veri disk miktarını verimsiz kullandığı gibi bakım problemleri de doğurur. Eğer veri birden fazla yerde mevcut ise değişiklik sonucu o verinin tüm bulunduğu yerlerde değiştirilmesi gerekir. Tutarsız bağımlılıklar veriye erişimi güçleştirebilir. Normalizasyon, özellikler arasındaki fonksiyonel bağımlılıkların analizidir. İyi yapılandırılmış ilişkileri üretmek için aykırılıklar ile birlikte ilişkilerin ayrıştırılması sürecidir. İyi yapılandırılmış ilişki minimal fazlalık içerir ve hatasız olarak ekleme, silme, güncellemeye izin verir. Normalizasyon hangi özelliklerin ilişki içinde birlikte gruplanabileceğinin karar verildiği ve gereksiz veri çoğullamasından kaçınmayı sağlayan belirli kısıtları uygulayarak mantıksal tasarımı güçlendiren süreçtir. Normalizasyon teorisi normal form kavramına dayanır. İlişkisel tablo eğer belirli kısıt kümesini tatmin ederse kısmi normal formdur. Şu ana kadar tanımlanmış beş normal form mevcuttur. Normalizasyon veri fazlalığını ortadan kaldırması gerekirken veri bütünlüğü üstündeki gideri ortadan kaldırmaz. Normalizasyon süreci karmaşık veri özelliklerinden basit varlık özelliklerini üretir.

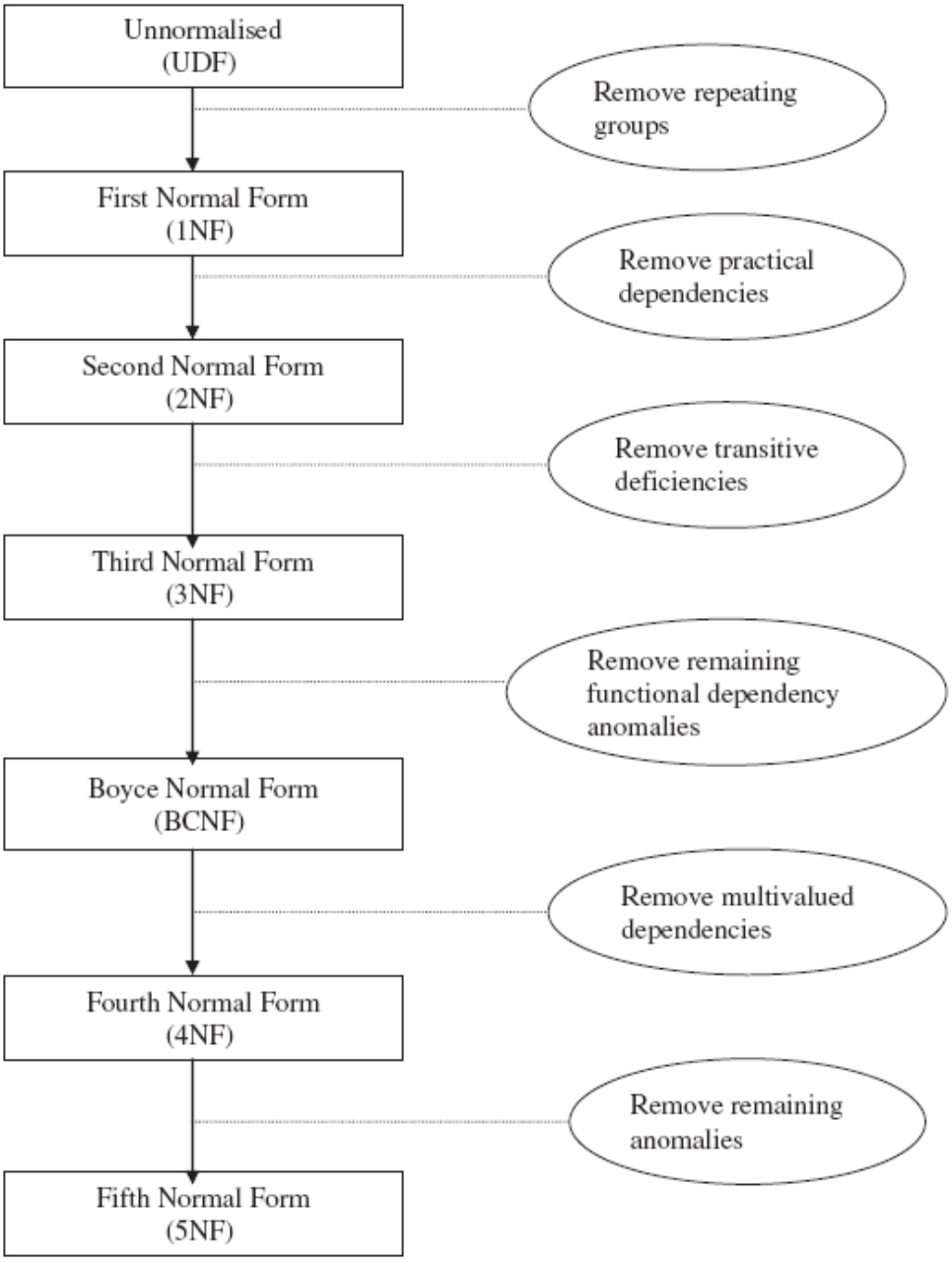
2.1.2.11.1 Normalizasyonun amacı

Ekleme, güncelleme, silme aykırılıklarını azaltırken veri tutarlılığının korunmasına izin verir. Normalizasyon için verilebilecek amaçlar aşağıda sıralanmıştır:

- Veritabanı içinde gerçekleri yalnızca bir kere depolayarak fazlalıktan kaçınmak.
- Veriyi değişikliklerin yerinin daha doğru olarak bulunabilmesini sağlayan şekle sokmak.
- Belirli güncelleme aykırılıklarından kaçınmak.
- Veri tutarlılığı uygulamasından faydalanmak.
- Gereksiz kodlamadan kaçınmak. Tetikleyici (Trigger) ve Depolanmış yordam (Stored procedure) içindeki ekstra programlama normalize olmamış veri için ihtiyaç duyulabilir.

2.1.2.11.2 Normalizasyon adımları

Normalizasyon derecesi normal formlar ile belirlenir. Artan normalizasyon düzeylerindeki normal formlar birinci normal form (1NF), ikinci normal form (2NF), 3NF, Boyce-Codd normal form,4NF ve 5NF'dur (Şekil 2.15). Her bir normal form, güncelleme aykırılıkları ve fazlalıklarla ilişkili belirli özellikleri garanti eden şartlar kümesidir. 3NF yeterince iyi olduğu düşünülür. Belirli örneklerde daha düşük düzeyli normalizasyon düzeyi, sorguların icra edilmesinin büyük vakit aldığı yerdir.



Şekil 2.15 Normalizasyon adımları

İlişkisel teori, veritabanında belirli veri aykırılıklarının oluşmayacağını garanti eden normal formlar olarak adlandırılan bir kısım yapı şartlarını belirler.

2.1.2.11.2.1 Birinci normal form (1NF)

Tablo içindeki tüm sütunlar küçük değerler içeriyorsa ve satır içinde tekrar eden gruplar yoksa bu tablo birinci normal formdadır denir. Bir alan içindeki değerler aynı tipte olmalı ve alan tekil isme sahip olmalıdır. Tablo içindeki her bir kayıt tekil olmalıdır.

2.1.2.11.2.2 İkinci normal form (2NF)

Birinci normal formda ve her anahtar olmayan özellik tam olarak bir birincil anahtara bağımlı ise bu tablo ikinci normaldedir denir.

2.1.2.11.2.3 Üçüncü normal form (3NF)

İlişkinin üçüncü normal formda olması o ilişkinin 2NF olması ve geçişli bağımlılıkların olmaması ile mümkündür. Geçişli bağımlılık bir özellik dolaylı olarak anahtara fonksiyonel olarak bağımlı olduğu zaman oluşur.

2.1.2.11.2.4 Boyce–Codd normal form (BCNF)

İlişkinin BCNF olması için 3NF olması ve her bir determinantın anahtar adayı olması gerekir.

2.1.2.11.2.5 Beşinci normal form (5NF)

Belirsiz olan bağımlılıklarla ilgilenir.

2.1.2.11.2.6 Alan (Domain)/Anahtar (Key) normal form (AA/NF)

İlişki üstündeki her bir kısıt alan ve anahtarların, tanımlarının mantıksal dizisi olmalıdır.

2.1.2.11.3 Normalizasyon dezavantajları

Normalizasyon birçok tablo üretir. Veri çeken sorgular birçok tablo üstünden bağ (join)'lar kullanılarak alınmaya başlar ki bu performansı düşürür.

2.1.2.12 Denormalizasyon

Denormalizasyon sorgu performansını düşüren aşırı normalizasyon durumlarında performansı artırmak için kullanılır.

2.1.2.12.1 Denormalizasyon temel tipleri

Beş tip denormalizasyon tipi vardır:

- Çoktan çoğa ilişkideki iki varlık: Çoktan çoğa ilişkideki iki tablo bir ilişki tablosu doğurur. İlişki tablosunun varlık tablolarından biri ile bağlanarak oluşan tablo ilişki tablosu ve varlık tablolarından biri yerine kullanılarak performans artımı yapılır.
- Bire bir ilişkideki iki varlık: Birebir ilişkili iki tablo tek tablo olarak kullanılarak bağ yükünden kaçınılabilir.
- Birden çoğa ilişkideki referans veri: Yapay birincil anahtar, anahtarı olmayan veya çok karmaşık anahtarı olan bir tabloya eklenerek bu tablo ile ilişkili diğer tablolara bu alan yabancı anahtar (foreign key) olarak eklenerek performans artırılır.
- Çok detaylı verili varlıklar: Çok değerli alanlar farklı bir tablo olarak sunulur. Çoğu zaman bu tabloların ana tabloya bir alan olarak eklenmesi daha verimlidir.
- Türetilmiş özellikler: Bir alandan türetilen diğer alanlar o tablonun diğer alanı olarak kayıt edilmesi verimliliği artırabilir.

Denormalizasyon, performansa etki edecek en baskın süreç bulunarak kayıt maliyeti ile sorgu maliyeti arasındaki en uygun değer çözüme göre yapılır.

Veritabanı tasarımı için gerekli matematik alt yapı ve bunun uygulandığı normalizasyon tekniğinden bahsedilmiştir. Nesnel tasarımla elde edilen sınıf diyagramı ile veri tabanının yapısını oluşturacak veri modeli arasındaki ilişkiden faydalanan yöntemler mevcuttur. Çizelge 2.1'de veri modeli ve nesne modeli arasındaki bakış açıları farklılıkları listelenmiştir.

Çizelge 2.1 Nesne modeli ile veri modeli arası farklılıklar

Nesne Modeli	Veri Modeli
Hafızanın verimli olması için sınıfları nasıl tasarlarım?	Deposunun verimli olması için veri tabanını nasıl tasarlarım?
İlişki içinde olması gereken nesneler nelerdir?	İlişki içinde olması gereken tablolar nelerdir?
Son kullanıcıya daha hissedilir kılmak için veriyi nasıl yapılandırabilirim?	Erişim zamanını hızlandırmak için veriyi nasıl yapılandırabilirim?
Sınıfları oluşturmak için veri ve davranışı nasıl paketlerim?	Veriyi nasıl normalize ederim?
Hangi veri uygulama boyunca kullanılırken hangi veri yalnızca bir alanda kullanılır?	Veri hangi sıklıkla çekilir?
Kod yeniden kullanımı genelleme ve diğer tasarım stratejilerini nasıl kullanabilirim?	Kalıtım mimarisini veri tabanı sağladığı durumda kalıtım kavramını nasıl kullanabilirim?

Çizelge 2.1'deki iki model arası ayrılıklar, odaklandıkları noktalardan dolayı oluşur. Nesne modeli veri ve davranışa odaklanırken veri modeli yalnızca veriye odaklanır.

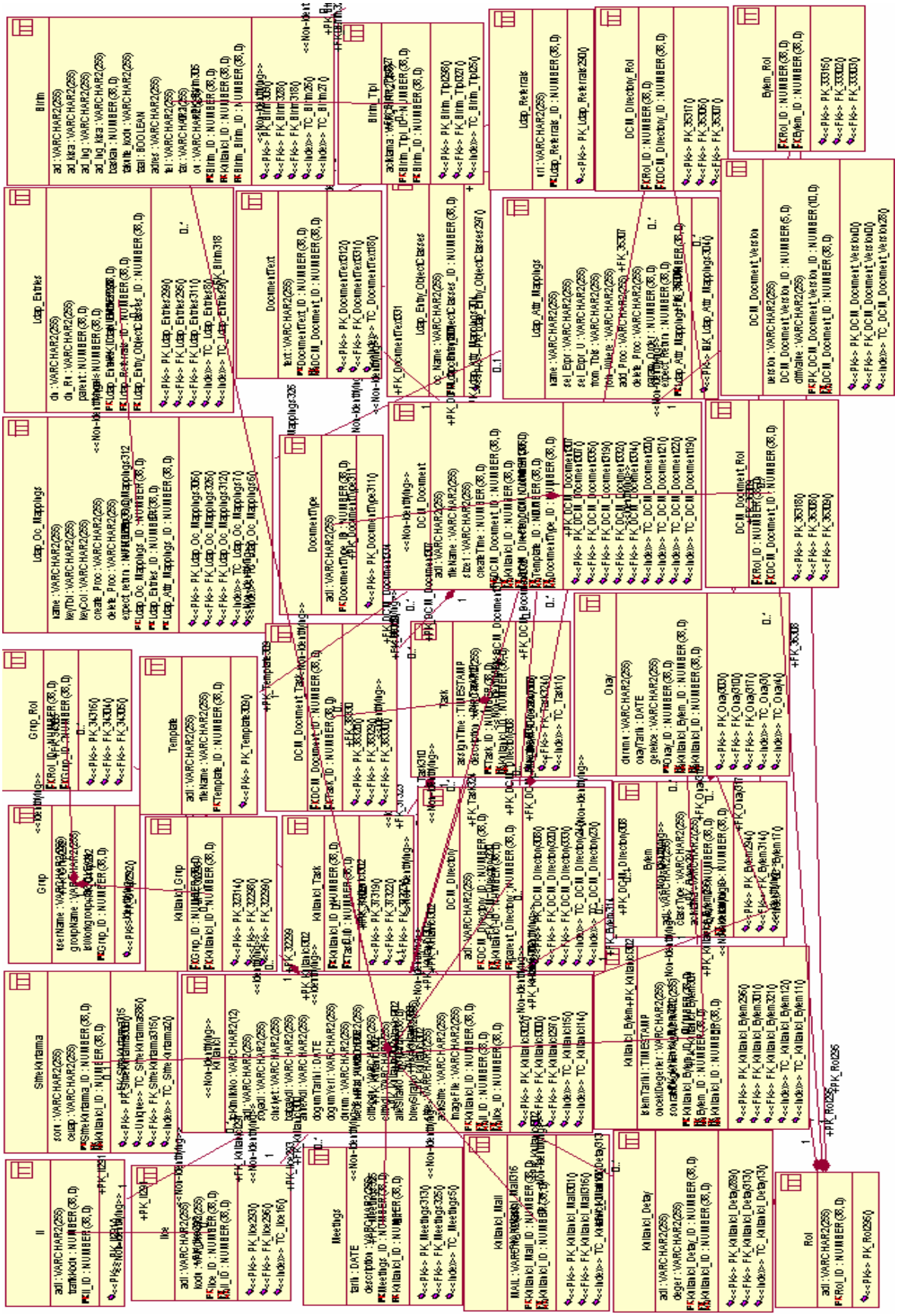
Veritabanı sistemlerinin gelişimi uygulama mantığının bir kısmı yazılacak tetikleyici (trigger), seyir (view) ve depolanmış yordamlar (stored procedure) ile veritabanına eklenebilir. Burada veri ile ilgili mantık veritabanına eklenmesi uygun iken şebeke trafiğini azaltmak için iş mantığının bir kısmının (veri doğrulama gibi) uygulama katmanında icra edilmesi daha uygundur. Bununla birlikte hem veri katmanına hem de

uygulama katmanına sarkan mantık da yapılması gereken değişiklikler her iki tarafa da güncellenmelidir. İş mantığının veri ve uygulama katmanına sarktığı durumlarda uygulama katmanın seçilmesi daha uygundur. Bir veri tabanından diğerine geçilirken veri tabanına gereğinden fazla iş mantığının konulmuş olması tüm geçiş sürecini zorlaştırır ve kimi zaman imkânsız kılar.

Yazılım geliştirme süreci boyunca hazırlanmış sınıf diyagramındaki sınıfların birçoğu veri modelindeki tablolara dönüşür. Hem iki modelin bakış açılarındaki farklılık hamda bir sınıf birden fazla tablo tarafından desteklenebilirken birden fazla sınıf tek tablo tarafından da desteklenebileceği için burada bire bir karşılık yoktur.

GDYS sistemi için sınıf diyagramından oluşturduğumuz veri modeli diyagramı şekil 2.16'da görülebilir. Diyagramdan da görüldüğü gibi 1NF normalizasyonu sağlamak için tanım tabloları kullanılmıştır. 2NF'de tablolara olan referanslar için yabancı anahtarlar kullanılmış ve her tablo için bir birincil anahtar tanımlanmıştır. 3NF'i sağlamak için yani geçişsel bağımlılıkları ortadan kaldırmak amacıyla referans tabloları kullanılmıştır. Örneğin bir kullanıcıya ait Task'leri belirlemek üzere Kullanıcı_Task tablosu Kullanıcı tablosu ile Task tablosu arasında ilişki sağlamak üzere kullanılmıştır.

Analizi yapılarak idare edilecek verinin belirlendiği GDYS sisteminde bir sonraki bölümde sistem isterlerinin kullanıcı tarafından icra edilebilmesi için elde edilmiş kullanım vakalarını eylem olarak kullanan ağ ara yüzlerini içeren ağ uygulamasının tasarımı ve ara yüz teknolojisi anlatılmıştır.



Şekil 2.16 GDYS sistemi veri modeli

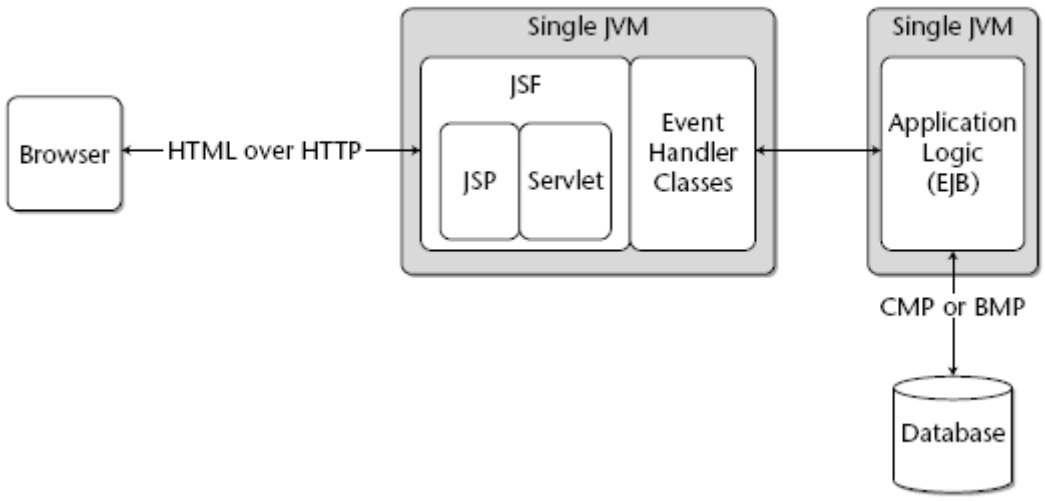
2.1.3 GDYS sistemi ağ tasarımı

GDYS sisteminde kullanıcı ara yüzlerinin normal çalıştırılabilir programdaki ara yüzler kadar interaktif olması ve güvenliğin bir adım daha güçlendirilmesi amacı ile JSY (Java Sunucu Yüzleri) teknolojisi seçilmiştir. JSY teknolojisi alışıldık ağ uygulamalarını oluşturan ağ sayfalarından farklı olarak yeniden kullanılabilir bileşen mimarisinden faydalanır. Aynı zamanda durumsuz (stateless) yapıda olmayıp istemci ara yüzünün son durumu ile ilgili bilgi sunucu tarafında tutulur. Durumlu (Statefull) yapı ara yüzün daha interaktif olması ve daha güvenli olmasını sağlar.

2.1.3.1 JSY

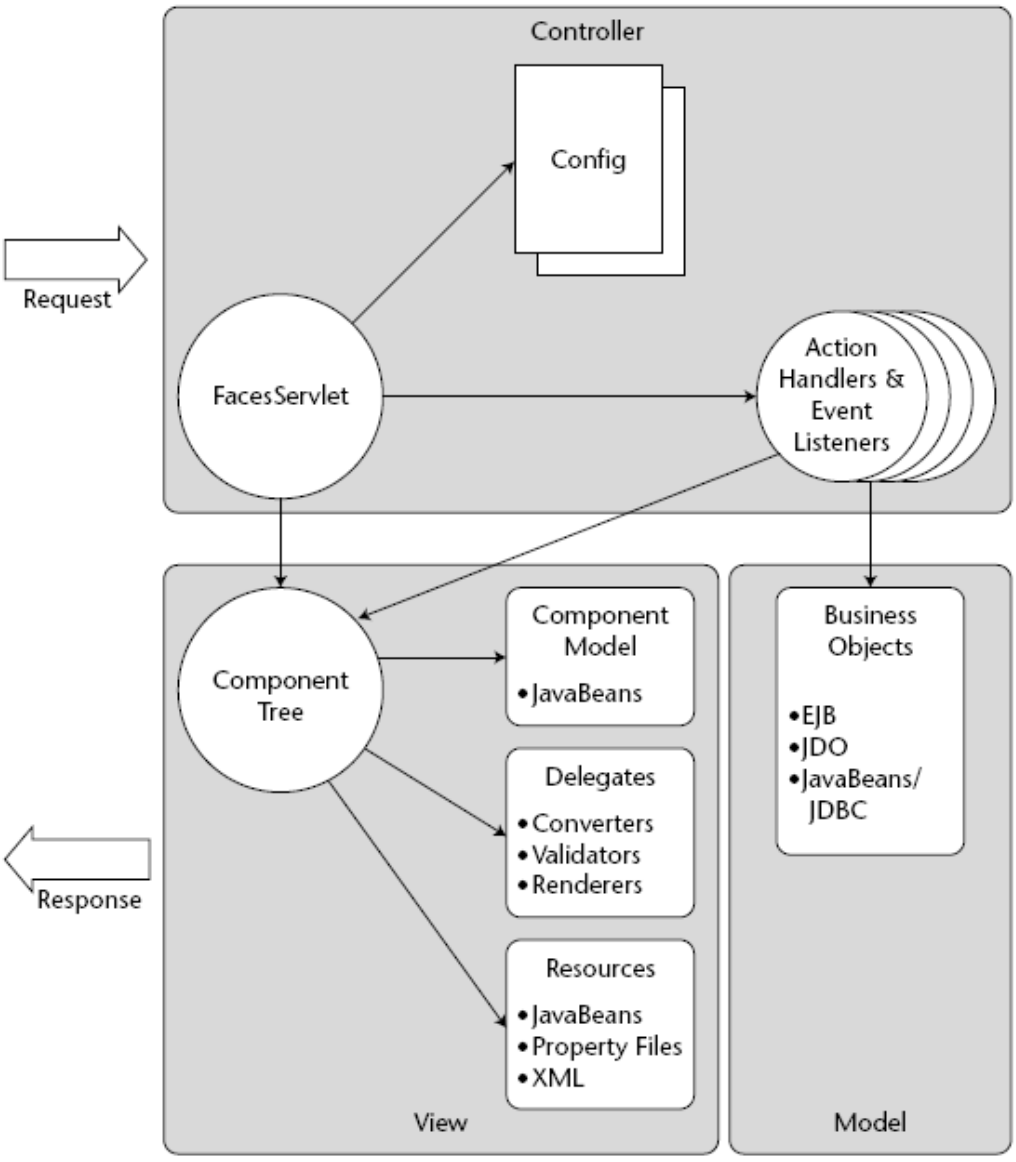
JSY teknolojisi ile üretilmiş ağ sayfalarının ortak özelliği sayfayı oluşturan tüm bileşenler <f:view> ile künyelenmiş tek bir kök bileşen altında ağaç yapısı oluştururlar (Dudney *et al.* 2004).

Şekil 2.17'de görüldüğü gibi JSY, uygulama katmanında JSS/Servlet teknolojisinin oturduğu kısımda bileşen mimarili uygulama oluşturmak için bulunur (Bergsten, H. 2002). Kullanıcı ara yüzü oluşturarak iş mantığını bir sonraki katmanda bulunan sınıflara aktarırlar.



Şekil 2.17 Dört katmanlı JSY uygulama yapısı

JSY, bölüm 2.2.1.3'de anlatılmış MSD şablonu üstüne inşa edilmiştir. Model veri sunumu katmanını, denetleyici (controller) seyir ile model arasındaki iş mantığının oturduğu genel kontrollerin yapıldığı katmanı ve seyir kullanıcıya sunum katmanını temsil eder (Şekil 2.18). JSY seyir katmanı JSS gibi sayfa merkezli bir yapıda olmayıp olay güdümlü, organize edilebilir ağaç yapısına sahiptir.



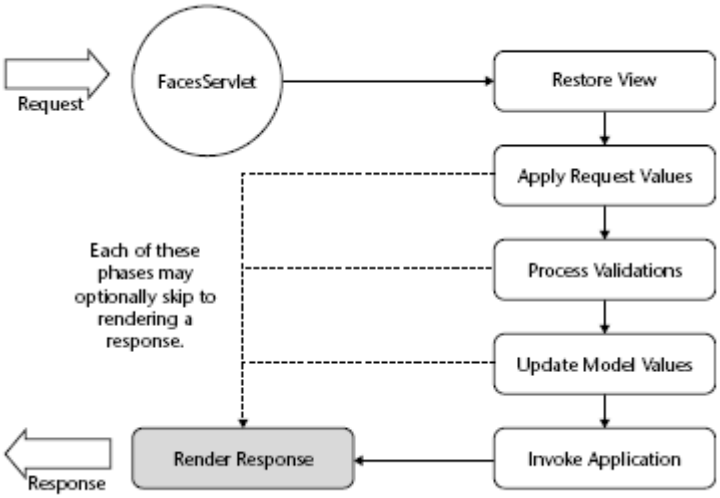
Şekil 2.18 JSY yapı taşları

JSY yapısı FacesServlet denen ön denetçiye sahiptir. FacesServlet ana uygulama içinde Face-config.xml dosyası ile yapılandırılır. GDYS sisteminde kullanılan Faces-config.xml'in bir kısmı aşağıda sunulmuştur.

```
<faces-config>
    <managed-bean>
        <managed-bean-name>skinBean</managed-bean-name>
        <managed-bean-class>
            tr.edu.ankara.utils.common.SkinBean
        </managed-bean-class>
        <managed-bean-scope>session</managed-bean-scope>
        <managed-property>
            <property-name>skin</property-name>
            <property-class>java.lang.String</property-class>
            <value>blueSky</value>
        </managed-property>
    </managed-bean>
    <navigation-rule>
        <from-view-id>/admin/template/searchTemplate.jsp</from-view-id>
        <navigation-case>
            <from-outcome>listTemplate</from-outcome>
            <to-view-id>/admin/template/listTemplates.jsp</to-view-id>
        </navigation-case>
    </navigation-rule>
    <lifecycle>
        <phase-listener>
            tr.edu.ankara.utils.tree.PostbackPhaseListener
        </phase-listener>
    </lifecycle>
</faces-config>
```

2.1.3.2 JSY yaşam döngüsü

İstemciden gelen talepler alınır, belli adımlar takip edilerek cevap tekrar istemciye döndürülür. Şekil 2.19'da gösterilen JSY yaşam döngüsü FacesServlet'in sorumluluğu olan adımları belirtir.



Şekil 2.19 JSY yaşam döngüsü

Şekil 2.19'da görüleceği gibi bu adımlar aşağıda açıklanmıştır:

2.1.3.2.1 Seyir düzeltme (Restore view)

Talep sayfası için bileşen ağacının oluşturulması eylemidir. Eğer bir sayfa aynı kullanıcı oturumunda daha önce sunulmuş ve verisi kayıt edilmiş ise FacesServlet bu veriden bileşen ağacını oluşturur. Talepler arası bileşen durumunun kayıt edilmesi JSY'in önemli bir özelliğidir.

2.1.3.2.2 İstek değerlerini uygulama (Apply request values)

Eğer bileşen ağacı oluşturuldu ise talep ile gelen yeni bilgi bileşen ağacına güncellenir. Bir sonraki adıma geçemeden önce kuyruklanmış veya ani olarak yayınlanması gereken kullanıcı ara yüzü bileşen olayları işlenir.

2.1.3.2.3 Süreç doğrulama (Process validations)

Her kullanıcı ara yüzü bileşeni kendi durumunu kontrol eden kayıt edilmiş doğrulayıcı (validator) veya dâhili doğrulama mantığı içerebilir. Bu adımda bileşenin durumu doğrulanır.

2.1.3.2.4 Model değerlerini güncelleme (Update model values)

Model değerlerinin güncellendiği adım boyunca her ara yüz bileşenin arka plan modeli güncellenir.

2.1.3.2.5 Uygulama çağırma (Invoke application)

Kullanıcı ara yüzü bileşenleri güncellendikten sonra ActionListener, kuyruklanmış ve UICommand ile eşlenmiş ActionEvent'i cevaplar. UICommand bileşeni uygulamanın iş nesneleri ile olan etkileşimi tetikler.

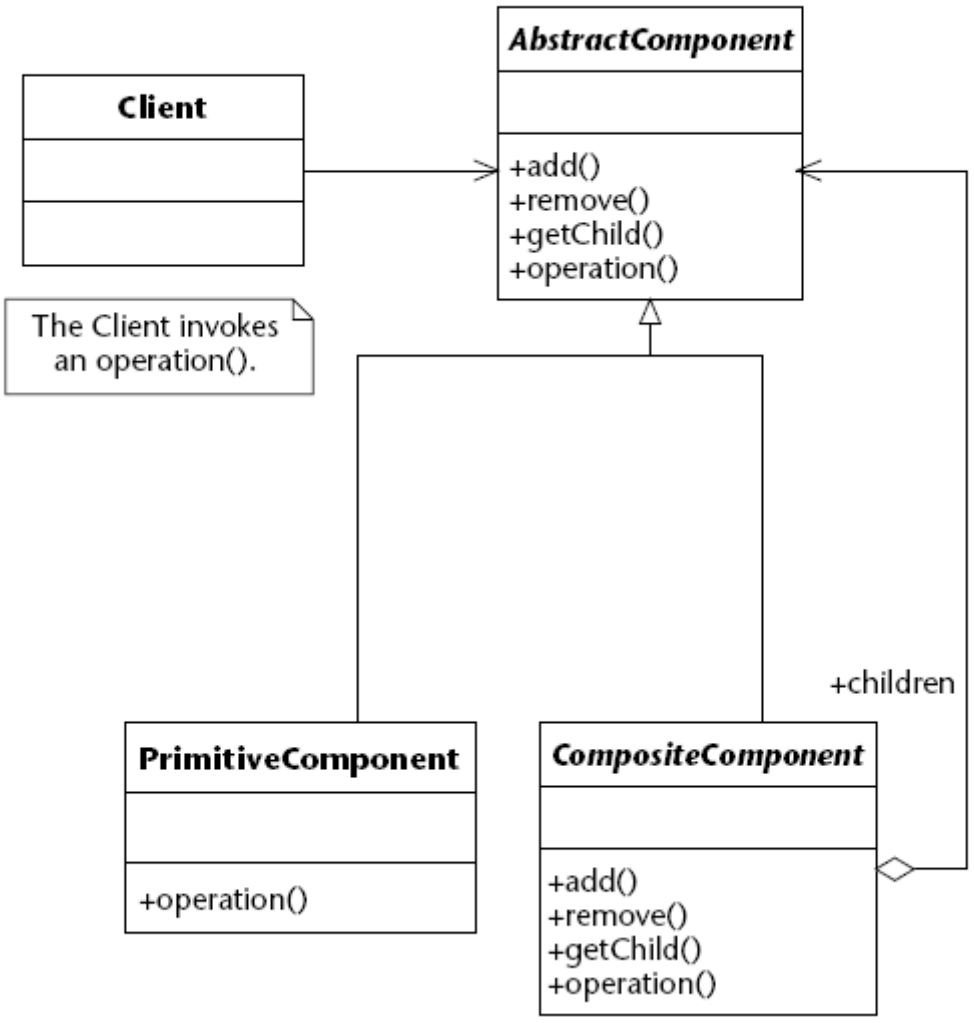
2.1.3.2.6 Cevap verme (Render response)

Talep işleme sonucuna göre cevap ya mevcut bileşen ağacından veya yeni bir bileşen ağacından üretilir. Seyir için JSS sayfası kullanılırsa her bileşen ağacı bir JSS sayfası ile eşlenir.

Yukarıda belirtilen managed-bean JSY uygulamalarında bileşenlerin arka plan verisini ve oturumlar arasında bileşenlerin durumlarını tutan Java sınıflarıdır. Belirtilen özellikler Java yansıtmaya (reflection) uygun olarak sınıf tanımı içinde bulunan özelliklerle aynı olmalıdır. Managed-bean-class ile sınıfın ne olduğu, managed-bean-scope ile sınıfın kabın hangi bağlamında depolanacağı ve managed-property ile sınıf yaratıldığında adı ve tipi ile belirli özelliğe ilk değer olarak ne verileceği belirtilir. Seyir içinde tanımlanan eylem (action) sonucu hangi sayfanın oluşturulacağı faces-config.xml dosyası içinde navigation-rule kısmında belirlenir. From-view-id ile hangi sayfadan navigation-case içinde tanımlanan to-view-id ile belirlenen hangi sayfaya gideceği o

seyirde tanımlanan formdaki eylem içinde kullanılmış iş nesnesinin döndürdüğü kelime from-outcome'da belirlenen kelime ile eşlenerek belirlenir.

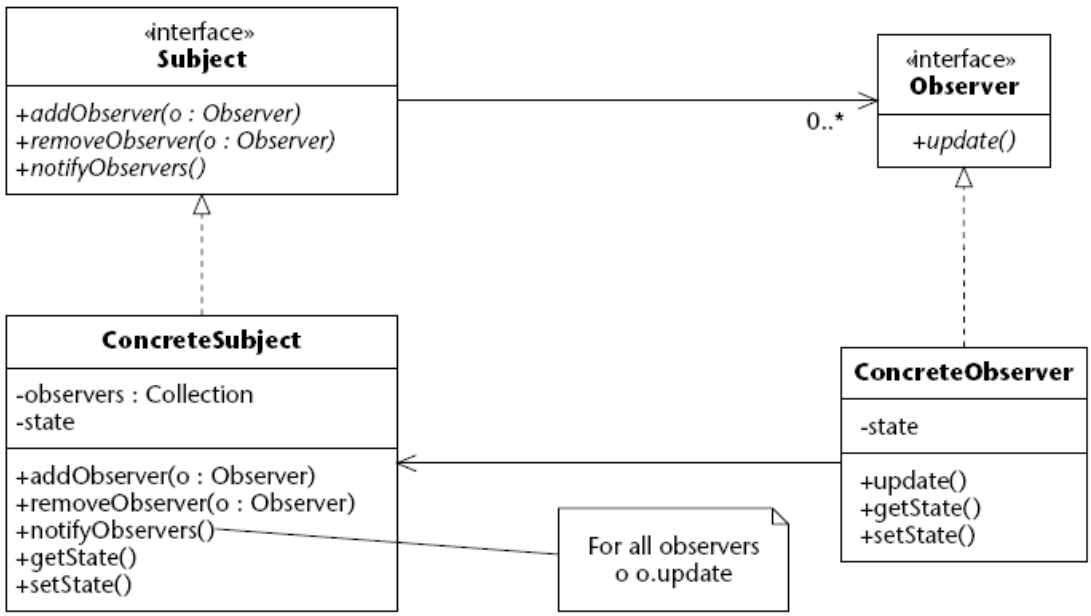
Bileşen ağacı, Karma (Composite) şablonunun uygulanması ile elde edilir (Grand 2002). Karma şablonunda CompositeComponent'in AbstractComponent tipli tüm bileşenleri ilişkilendirerek referans etmesi ağaç yapısının temelini teşkil eder. Bir CompositeComponent, CompositeComponent ve PrimitiveComponent örneklerini kendi bünyesine katabilir. JSY bileşen ağaç yapısının kökünü oluşturan <view> künyesi ile belirlenen bileşen, sayfa içindeki tüm bileşenleri bu yapı ile kendi bünyesinde barındırabilir.



Şekil 2.20 Karma şablonu yapısı

JSY olay (event) mekanizması Gözetmen (Observer) şablonu üstüne kuruludur. Şekil 2.21'de görüldüğü gibi konu bileşene ait arka plan verisini gösterirken gözetmen ise o veride meydana gelen değişiklik sonucu konu tarafından uyarılan bileşeni gösterir. Konu kendine kayıt olmuş gözetmenin tipini önemsemeksizin uyarır ve gözetmen ise bu uyarıma bir cevap verir. JSY iki farklı olay kullanır. ValueChanged, Action olayları ve bunlara cevap oluşturan ara yüz bileşenleri mevcuttur. Olayları cevaplamak isteyen bileşen ilk önce olayı oluşturan konuya (subject) kendini kayıt eder ve bu olayı cevaplamak için olay ile ilgili ara yüzü gerçeklemesi gerekir. Aşağıda görüldüğü gibi seyir içinde bir olaya dinleyici (listener) kayıt edilir.

```
<h:inputText id="userId" value="#{login.userId}">
        <f:valueChangeListener type="logindemo.UserLoginChanged" />
</h:inputText>
```



Şekil 2.21 Gözetmen şablon yapısı

Buraya kadar anlatılan JSY teknolojisinin bir uygulaması olan Red Hat tarafından geliştirilmiş RichFaces işçatısı, GDYS sistemi içinde kullanılır ve açık kaynak kodlu oluşu ve EJBG desteği oluşu tercih sebebidir. JSY yaşam döngüsünü tam olarak gerçeklerken mevcut JSY sayfalarına EJBG yeteneklerini de ekler (Red Hat 2007).

Hızlı olarak karmaşık seyirler oluşturulabilir. Excel sayfası, ses dosyası oluşturma gibi desteği mevcuttur.

JSY desteği bir ağ uygulamasına web.xml dosyası içine GDYS sistemi için kullandığımız aşağıdaki gibi tanımlama parçası ile verilebilir.

```
<servlet>
  <servlet-name>Faces Servlet</servlet-name>
  <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
 </servlet>
 <servlet-mapping>
  <servlet-name>Faces Servlet</servlet-name>
  <url-pattern>*.jsf</url-pattern>
 </servlet-mapping>
```

RichFaces iş çatısı kullanılarak oluşturulan ağ uygulamasının genel sayfa yerleşimi şekil 2.22'de görülebilir. Sayfalar paketler halinde organize edilmiştir. Paketler, menü komutları ile ilişkili olarak gruplanmıştır. Aşağıda güvenlik kontrolü dışında kalan sayfalara ait sınıf diyagramı mevcut iken paketler içindeki sayfalar ise güvenlik kontrolü altındadır.

Her bir kullanım vakası ile ilgili sayfalar bir paket içine alınmıştır. Dip bölüm (footer), başlık, menü ve ilan tahtasından oluşan genel parçalar ile ilgili sayfaları, giriş ve hata (error) sayfaları hariç tüm sayfalarda ortak olduğu için ortak (common) paketi altında toplanmıştır.

Tamamlanan ağ, veritabanı ve HDEP analiz ve tasarım süreçlerini bir sonraki bölümde anlatılmış fiziksel gerçeklemeler takip etmiştir.

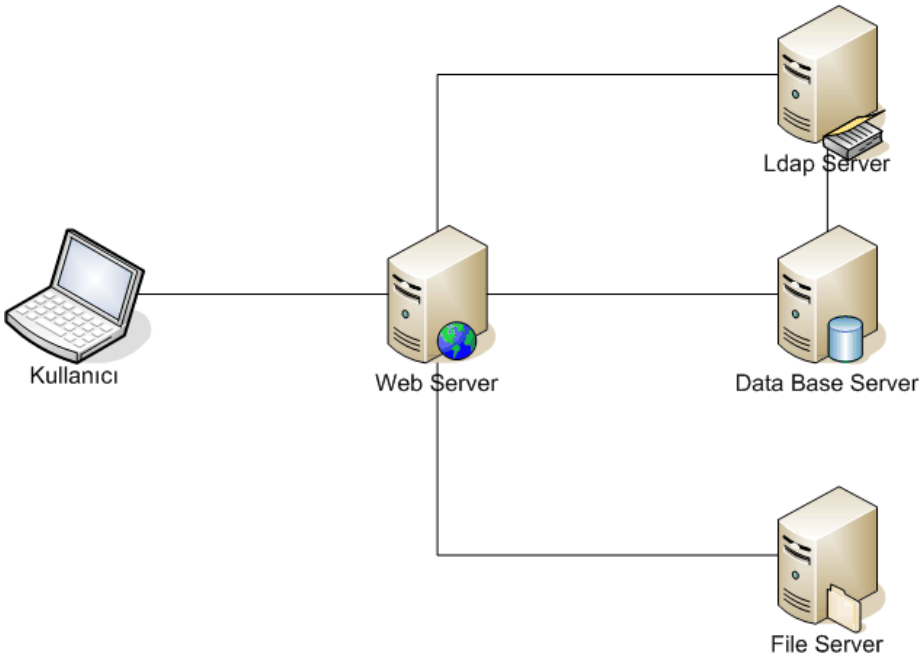
Şekil 2.22 GDYS ağ uygulaması modeli

2.2 GDYS Genel Yapısı

GDYS isterleri ile başlayan analiz çabalarını BMD ile yapılan tasarım çabaları takip etmiş ve gerekli diyagramlar oluşturularak veritabanı, HDEP ve ağ uygulamasını teşkil edecek fiziksel tasarım tamamlanmıştır. Bu kısımda tamamlanmış fiziksel tasarım ve gerçeklenmiş yazılım bileşenleri fiziksel sistemler üstüne alınarak GDYS sistemi hayata geçirilmiştir.

Güvenli Doküman Yönetim Sistemi, akademik yayınları yönetmek ve güvenli olarak yayınlama işlevini yerine getirmek için modüler yapıda çeşitli sistemleri bir arada kullanır. GDYS şekil 2.23'de görüldüğü üzere temel olarak ağ sunucu, HDEP sunucu, veritabanı sunucu ve dosya sunucudan oluşmaktadır. Yayınları ve ilişkili bilgileri güvenli olarak yayınlama görevini JGS standardını uygulayan Tomcat sunucusu yürütür. Güvenlik ve yetkilendirme JYKS standardı ve OpenLdap ile sağlanır. Yayınlar ise dosya sunucusu içinde saklanır. Yayınlar ve araştırmacılar ile ilgili bilgiler ve sistem yapılandırma bilgilerinin bütünlüğü ise veritabanı sunucusunca sağlanır.

Takip eden bölümde ağ uygulamasının koştuğu Tomcat sunucusu hakkında bilgi verilerek kurulumu ve üstünde koşan uygulama yapısı anlatılmıştır.



Şekil 2.23 GDYS genel yapısı

2.2.1 Tomcat ağ sunucusu

Apache Software Foundation'ın Jakarta projesinin bir parçası olarak geliştirilmiş Tomcat servlet motoru açık kaynak kodlu yazılımdır (Bakore *et al.* 2003). Servlet ve JSS tanımlamalarının resmi referans uygulamasıdır. Tomcat tek başına ağ sunucusu ve aynı anda diğer ağ sunucuları için servlet/JSS motoru gibi davranabilir. Tomcat sunucu indirilirken birçok paketle birlikte indirilir. Catalina ve Jasper, servlet/JSS kaplarının adıdır.

Sun (1999), servlet kabı kodunu Apache Software Foundation tarafından geliştirilen JServ ürünü ile birleştirilmek üzere Jakarta Tomcat projesine bağışlamıştır. Oluşturulan Tomcat Sunucu Sun'ın resmi referans uygulamasıdır. Referans uygulaması tamamen spesifikasyon uyumludur. Spesifikasyondaki ileri düzey kısımları kullanmak isteyen kişiler için değerlidir. Sun, servlet mimarisi için yeni spesifikasyonları sunduğu an bunların uygulaması da Tomcat tarafından gerçeklenir.

İlk Tomcat sürümü 3.x, Servlet 2.2 ve JSS 1.1 spesifikasyonları uygulaması olmuştur. 2001 de, Tomcat 4.0 (Catalina) Servlet 2.3 ve JSS 1.2 spesifikasyonları uygulaması olmuş ve Tomcat 5.0 ise Servlet 2.4 ve JSS 2.0 spesifikasyonlarını gerçeklemiştir.

Tomcat Apache lisansı altında dağıtılır ve $C_H Tomcat kurulum dizini olmakla birlikte $C_H/LICENSE dosyasından okunabilir. Bu lisansın anlattığı anahtar noktalar:

- Apache lisansı, Tomcat'in herhangi bir kaynak kodu veya derlemesi dağıtılırken içerilmeli.
- Dağıtıma eklenmiş herhangi bir doküman ASF'e bir kabul içermeli.
- Tomcat kaynak kodundan türetilmiş herhangi bir ürün "The Jakarta Project", "Apache" veya "Apache Software Foundation" terimlerini ASF tarafından izin verilmeden kendi yazılımlarının reklâmını yapmak için kullanamaz.
- Tomcat herhangi bir çeşit garantiye sahip değildir.
- Tomcat herhangi bir ticari veya ticari olmayan kuruluş tarafından limitsiz olarak kullanılabilir.
- Tomcat'i değiştirip ve dağıtan kişiler kendi sürümlerini dağıtırken yapmış oldukları kodlamaları dağıtmak zorunda değiller.
- Tomcat'i değiştirip ve dağıtan kişiler kendi sürümlerini dağıtırken yapmış oldukları kodlamaları ASF'e bağışlamak zorunda değiller.

2.2.1.1 Tomcat kurulum klasörleri ve mimarisi

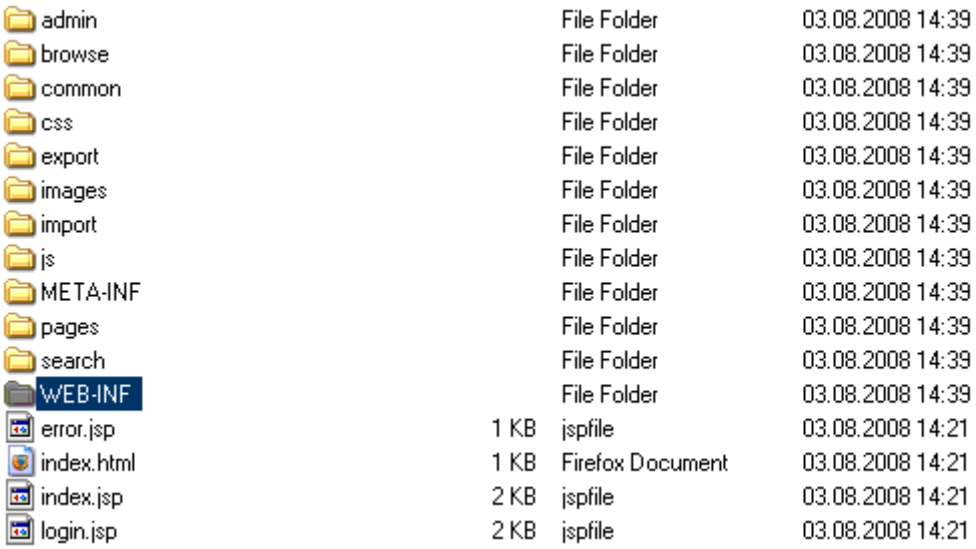
Tomcat ağ sunucusu çalışabilmesi için gerekli dosyaların belli dizinler içinde yerleştirilmesi gerekir. Dizin hiyerarşisi $C_H kurulum dizini altında bulunur:

- **Bin:** Kabuk betikleri, yığın dosyaları ve gerekli çalıştırılabilir dosyaların bulunduğu dizindir. Bunlar kullanılarak ana Tomcat süreci çalıştırılabilir, durdurulabilir ve bakım işleri yapılabilir.
- **Classes:** Kurulu sistemdeki tüm uygulamalar tarafından kullanılabilecek sınıfların bulunduğu dizindir.

- **Lib:** Kurulu sistemdeki tüm uygulamalar tarafından kullanılabilecek sınıfların arşiv halinde bulunduğu dizindir.
- **Server:** Tomcat süreci tarafından dâhili olarak kullanılacak sınıf ve arşiv dosyalarının bulunduğu dizindir.
- **Common:** Bu dizin altına yerleştirilmiş arşiv ve sınıf dosyaları hem sunucu hem de kurulu ağ uygulamaları tarafından kullanılabilir. Genel amaçlı paketler bu dizine yerleştirilebilir. GDYS ağ uygulamasında HDEP ve veri tabanı erişimi için kullanılan paketler bu dizinde bulunmaktadır.
- **Conf:** Sunucu yapılandırma dosyalarının bulunduğu dizindir. Bunlar içinde en önemlisi sunucunun hangi portları dinleyeceğinin, üstünde tanımlı uygulamaların ve Tomcat sunucusunun hangi diğer sunucular ile hangi portdan haberleşeceği, hangi paketleri kullanacağının tanımlandığı server.xml dosyasıdır. Kullanıcı ve kullanıcı ile ilgili şifre ve rol tanımlarının bulunduğu tomcat-users.xml dosyası diğer önemli yapılandırma dosyalarındandır.

  GDYS uygulamasında server.xml içindeki en önemli kısım veritabanı bağlantısının, bağlamın (context) ve posta (mail) sunucusunun tanımlandığı kısımdır. Bu tanımlama bölüm 2.1.3 JVTB (Java Veritabanı Bağlantısı) kısmında ele alınacaktır.
- **Logs:** Uygulamaların günce dosyaları için varsayılan dizindir. Sunucuya dışardan yapılan, localhost ve sunucusunun her yaptığı bağlantı için zamanı belirli günce oluşturulur. Bunlar kullanılarak uygulamardaki hatalar gözetlenip, uygulamaya ait genel eğilimler belirlenir. Aynı zamanda dışardan yapılan güvenlik ile ilgili davranışlar da gözetlenebilir. Her istek, standart hata ve standart çıkış akımları bu kısma yönlendirilebilir. Çoğunlukla ilgilenilen localhost.yyyy-aa-gg.log ve catalina.out günce dosyalarıdır. localhost.yyyy-aa-gg.log dosyası uygulamalar ile ilgili uyarı ve hataları tutarken catalina.out ise sunucusu ile ilgili uyarı ve hataları listeler.
- **Webapps :** Tomcat sunucu üstünde sunulan uygulamaların bulunduğu dizindir. Server.xml içinde bağlam kısmında tanımlanan uygulamalar adı bağlam içinde tanımlanan kök dizin altına belli dosya hiyerarşisi ile yerleşir. GDYS uygulamasında dcm kök dizini burada bulunur.

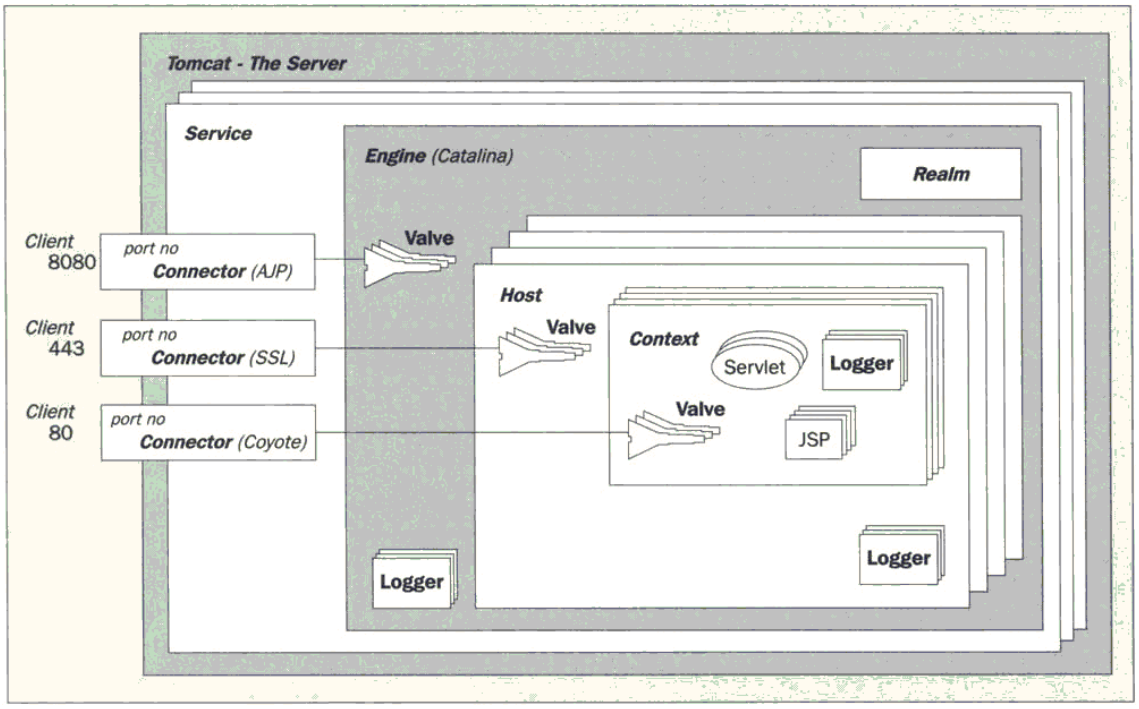
Ağ uygulaması Tomcat içine kurulduğu an kök dizini altında şekil 2.24'dekine benzer dosya yapısı oluşur (Armstrong *et al.* 2003).



Şekil 2.24 Ağ uygulaması dosya hiyerarşisi

WEB-INF klasörü ağ uygulaması için gerekli yapılandırma dosyalarını, paketleri, sınıf dosyalarını içerir ve bu kısma ağ üzerinden erişilemez. WEB-INF altında bulunan web.xml dosyası uygulamanın güvenlik yapısını, Tomcat tarafından sağlanan kaynakları, indis sayfasını, yetkilendirme bilgilerini ve uygulamanın kullanacağı diğer servlet motorları, ilgili süzgeçleri ve tanım bilgilerini içerir. WEB-INF klasörü harici kök klasör altındaki tüm klasör yapısına erişime yetkilendirme ile olanak verilir.

Şekil 2.25'de görülen Tomcat mimarisi iç içe geçmiş bileşenlere dayanır. Mimari içinde en yukarıda bulunan bileşenler diğer bileşenleri içine alır ve kap bileşenleri olarak adlandırılır. Bir başka bileşeni içermeyen bileşenler ise gömülü bileşen olarak adlandırılır.



Şekil 2.25 Tomcat mimarisi

Sunucu, Tomcat'in kendisidir. Kapatmak için ayrı portu vardır ve birbirinden etkilenmemesi gereken uygulamaları aynı makinede çalıştırmak için farklı portları dinleyen örnekler yaratılabilir.

Servis, birkaç bağlantı ve bir kabdan oluşan grubudur ve yönetin amacı ile her servisin bir adı vardır. Kap, bağlantıdan gelen isteği cevaplar. Bağlantı ise uygulamayı istemciye bağlayan bileşendir. Http 1.1 standardını uygulayan bağlantı Coyote'dir.

Sübaplar ise Tomcat'e istek işleme sürecinde araya girip tek nokta giriş (single sign on), başlık indirme (header dump) gibi işlemlerin yapılabilmesini sağlar. İstek işleme sürecine eklenebilir ve çıkartılabilirler. Tüm ağ uygulamasına şeffaftırlar. Sübaplar kap ile sunucu/bağlam, sunucu ile bağlam, bağlam ile kaynak arasına girebilirler.

Sunucu seviyesinde davranışı belirlenebilen günce tutucu, bileşenlerin dâhili durumlarını raporlar.

Sunucu, aynı makine üstünde birden fazla sanal sunucu alan isimleri ile ayrıştırılarak sunulabilir. Kap ise AMTP başlık içinde gelen sunucu adını inceleyerek hangi uygulamaya isteği yönlendireceğini belirler.

Bağlam ise ağ uygulamasının kendisidir. Konfigürasyonu içinde uygulamanın kök dizinin yerini uygulamanın kabına bilgilendiren kısmı içerir. Dinamik yükleme ile uygulamaya ait sınıf değiştiğinde sınıfın yeni hali uygulamaya aktarımı yapılabilir ama kararlı sistemlerde bu işleme izin verilmemelidir. Başlangıç parametreleri ile yapılandırılabilir. Tahsis edilen hata sayfaları ile yönetici çeşitli hata mesajlarını yapılandırabilir.

GDYS sisteminin Tomcat için kullandığı yapılandırma dosyası içindeki önemli kısım aşağıda sunulmuştur:

```
<?xml version="1.0" encoding="UTF-8"?>
<Server port="8005" shutdown="SHUTDOWN">
  <Listener SSLEngine="on"
className="org.apache.catalina.core.AprLifecycleListener"/>
  <Listener className="org.apache.catalina.core.JasperListener"/>
  <Listener className="org.apache.catalina.mbeans.ServerLifecycleListener"/>
  <Listener
className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener"/>
  <GlobalNamingResources>
    <Resource auth="Container" description="User database that can be updated and
saved" factory="org.apache.catalina.users.MemoryUserDatabaseFactory"
name="UserDatabase" pathname="conf/tomcat-users.xml"
type="org.apache.catalina.UserDatabase"/>
  </GlobalNamingResources>
  <Service name="Catalina">
    <Connector connectionTimeout="20000" port="8180" protocol="HTTP/1.1"
redirectPort="8443"/>
    <Connector port="8009" protocol="AJP/1.3" redirectPort="8443"/>
```

```
    <Engine defaultHost="localhost" name="Catalina">
      <Realm className="org.apache.catalina.realm.UserDatabaseRealm"
resourceName="UserDatabase"/>
      <Host appBase="webapps" autoDeploy="true" name="localhost"
unpackWARs="true" xmlNamespaceAware="false" xmlValidation="false">
        <Context docBase="dcm" path="/dcm" reloadable="true"
source="org.eclipse.jst.jee.server:dcm">
          ...
        </Context>
      </Host>
    </Engine>
  </Service>
</Server>
```

Yukarıdaki en önemli kısım sunucu ve bağlam kısımlarıdır. Sunucu ile hangi alanda çalışılacağı ki bununla sanal sunum sağlanır ve bağlam ile hangi uygulamanın o sanal sunucu üstünde sunulacağı belirlenir.

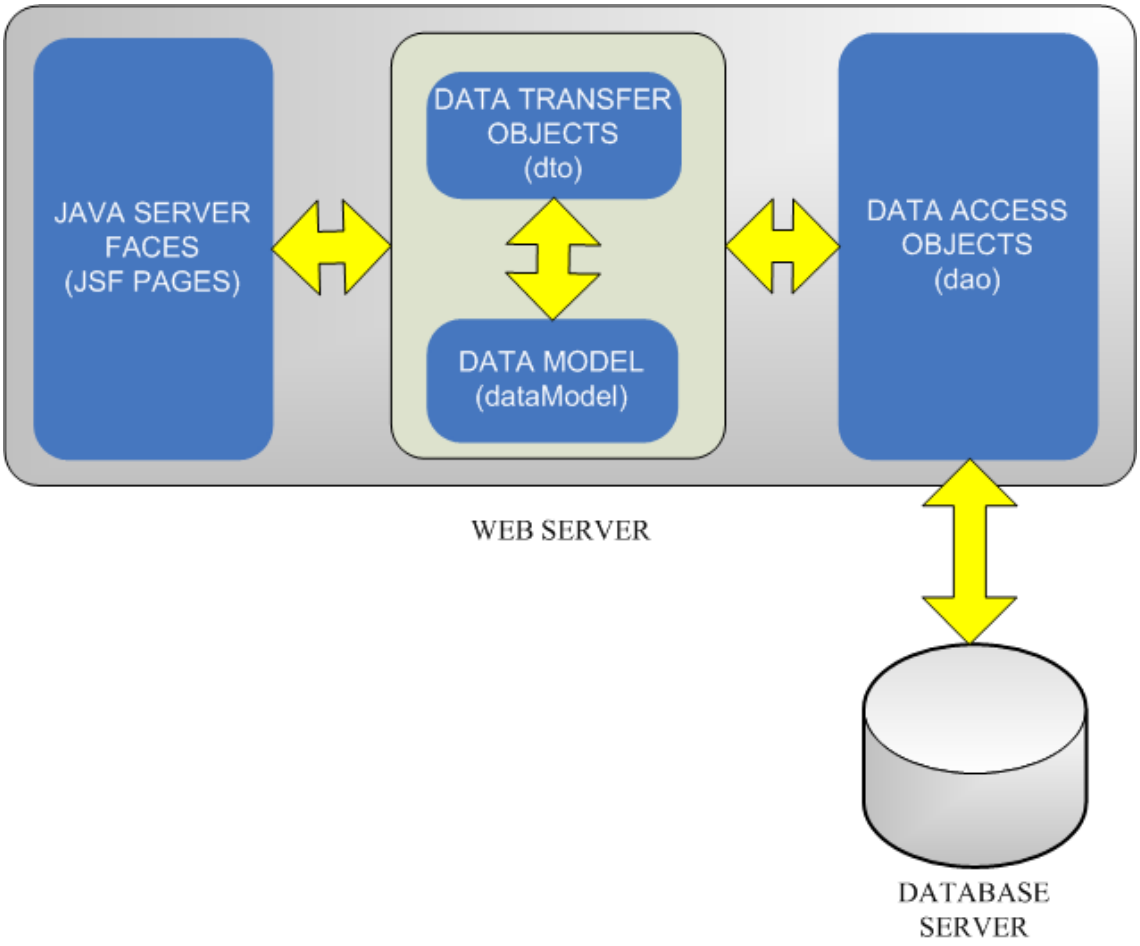
Bu bölümde fiziksel yapısından bahsedilmiş Tomcat sunucusu üstünde koşan ağ uygulamasının mantıksal örgütlenmesi takip eden bölümde sunulmuştur.

2.2.1.2 Ağ uygulaması mantıksal yapısı

GDYS sisteminin Tomcat sunucusu üstünde çalışan ağ uygulaması içinde bulunan ağ sayfaları ve bunlarla ilişkili sınıfların görevleri arasında mantıksal bir organizasyon mevcuttur. Şekil 2.26 içinde görülebilen örgütlenme ile ilgili yapıda ön kısımda kullanıcı ara yüzleri olan JSY sayfaları bulunur. Bunları VTN (Veritabanı Taşıma Nesneleri) sınıfları takip eder. VTN sınıflarının amacı JSY bileşenlerinin geri plan veri modellerini ve JSY sayfaları içinden tetiklenen eylemler için actionListener fonksiyonlarını sağlamaktır. VTN içinde saklanan veri, VEN (Veritabanı Erişim Nesneleri) sınıflarının JVTB kütüphanelerinden faydalanarak veri tabanından çekilir. Kısacası VEN çektiği veriyi VTN içinde depolar. VTN sınıfları direkt olarak JSY

sayfaları tarafından kullanılabildikleri gibi veri listeleme fonksiyonalitesini sağlayan veriModeli sınıfları tarafından da kullanılabilirler. VeriModeli sınıflarının amacı JSY sayfaları içinde listeleme ve sayfalama süreçlerini yönetmektir. Listeleme işlemini gerçeklemesi gereken JSY sayfaları managed-bean olarak veriModeli sınıflarını kullanır. VTN, VEN, veriModeli sınıfları tr.edu.ankara paketi altında VTN, VEN, veriModeli paketleri kullanılarak sağlanır.

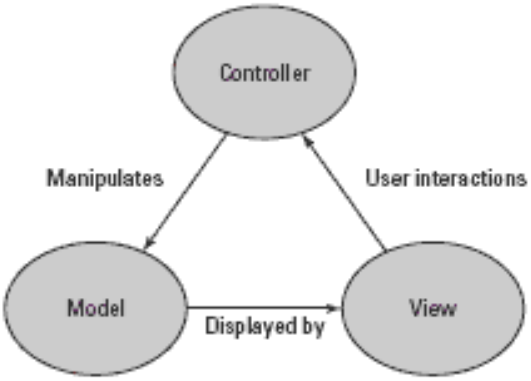
JSY sayfaları, VTN, VEN, veriModeli sınıfları ve FacesServlet sınıfı takip eden bölümde anlatılan MSD şablonunu icra eder.



Şekil 2.26 GDYS ağ uygulaması mantıksal yapısı

2.2.1.3 MSD

MSD, sunum mantığını (view), iş mantığını (control) ve veri nesnelerini (model) birbirinden ayırmak için model sunar. JGS'nin mimarisi rahatlıkla MSD modelle uygulanabilir (Khawar and Umrysh 2002). Varlık sınıfları (Entity beans) ile oturum sınıfları (session beans) kontrol mantığını oluşturmak için kullanılırken tipik olarak varlık sınıfları model mantığını sağlamak için kullanılır. Ağ bileşenleri ise kontrol ve sunum mantığını uygulamak için kullanılırlar. Pratikte MSD ayrımı tam anlamıyla kullanılmaz iken ek şablonlar da kullanılması gerekebilir. Şekil 2.27, faklı mantıksal blokların birlikte nasıl çalıştığını gösterir.



Şekil 2.27 MSD şablonu

2.2.1.3.1 Model

MSD içindeki M veri nesnelerini ima eder. Uygulama içinde işlemi gerçeklemek için gerekli bilginin nasıl belirlenip, kullanılıp, işleneceği ile ilgilenir. Genellikle JVTB bağlantıları ve GJF (Girişim Java Fasulyesi) kullanılarak bu süreçler gerçeklenir. Model görevi GDYS sisteminde VTN, VEN ve verimodeli sınıfları tarafından üstlenilmiştir.

2.2.1.3.2 Seyir (View)

Seyir sunumdan sorumludur. İstemcinin uygulamayı nasıl göreceği ve AMİD (Atlamalı Metin İşaretleme Dili) sorunları ile uğraşır. Çok çeşitli istemci tiplerine destek sağlamak

için KİD (Kablosuz İşaretleme Dili) ve GİD dillerini oldukça sık kullanır. Seyir görevi GDYS sisteminde JSY sayfaları tarafından üstlenilmiştir.

2.2.1.3.3 Denetleyici (Controller)

Kontrol şablonu iş uygulamanın mantığı ile uğraşır. İstemcinin modele erişme hakkı olan seyir ile nasıl ve ne zaman etkileşeceğini yönetir. Kontrol genellikle denetim, yetkilendirme ve uygulamaya ait kuralları uygulamak için gerekli servisler ile etkileşir. Seyirin modeli sunup sunamayacağı kullanıcının ait olduğu haklar uyarınca belirler. Aynı anda iş mantığının da icra edildiği katmandır. MSD fonksiyonel bloklarının ayrımı için gerekli UPA (Uygulama Programlama Ara Yüzü), JGS tarafından sağlanır. Denetleyici görevi GDYS sisteminde FacesServlet tarafından üstlenilmiştir.

GDYS sisteminde kullanılan ağ sunucusu olan Tomcat'in ve onun üstünde koşan ağ uygulamasının mantıksal yapısı incelendikten sonra takip eden bölümde JGS platformunun sunduğu UPA'lar, bunları kullanarak ağ uygulamasının Tomcat, veritabanı ve HDEP sunucusu ile nasıl etkileşebildiği ve GDYS isterlerinin ağ uygulaması tarafından nasıl gerçeklenildiği incelenmiştir.

2.2.1.4 JGS platform, uygulama sunucusu ve ağ kabı

JGS platformu Sun tarafından 1998'de tanıtılan büyük ölçekli bilgi sistemleri için birleşik bir platform oluşturan geniş çaplı UPA koleksiyonudur (McGovern *et al.* 2003). Çok katmanlı mimariye sahip JGS platformu içinde çeşitli mantıksal uygulama bileşenlerini ve servislerini kapsar. Karma şablonunu uygulayan, JGS platformunu gerçekleyen ve istek işleme görevini yerine getiren bileşen olan kap, uygulama sunucuları içinde bulunurlar. JGS platformu kapları aşağıda sıralanmıştır:

- Applet kabı,
- Applet istemci kabı,
- Ağ kabı,
- Girişim Java Fasulye (GJF) kabı.

Uygulama bileşenleri; uygulama istemcileri, appletler, ağ bileşenleri ve sunucu bileşenleri olarak 4 kısma ayrılır. İlk ikisi JSM içinde istemci tarafında çalıştırılan uygulamalardır. Uygulamanın sunum katmanını oluşturan ağ bileşenleri sunucu üstündeki JSM içinde çalıştırılır. Aşağıda verilmiş sunucu bileşenleri uygulama mantığının icra edildiği sınıflardır.

- Applet,
- Servlet,
- JSS,
- GJF,
- Java Posta,
- JMS (Java Mesajlaşma Servisi),
- Konnektör,
- JYE (Java Yönetim Eklentileri),
- JMU (Java Muamele UPA),
- JİDA (Java İsimlendirme ve Dizin Ara yüzü),
- JVTB,
- Yönetim araçları,
- Ağ servisleri,
- Java ATD (Ara yüz Tanımlama Dili),
- INTA-UMÇ (Internet Nesne Talebi Aracıları Arası Protokolü UMÇ) protokol,
- GÇJU (GİD Çözümleme için Java UPA),
- GKJP (GİD Kayıtları için Java UPA),
- KJYK (Java Kaplar için Java Yetkilendirme Servisi Sağlayıcı Kontratı),
- GUJ_UPA (GİD temelli UYÇ için Java UPA),
- JSEU (Java için SYUP (Servis Yönelimli Uygulama Programlama) Eklentileri UPA),
- JFİ (Java Fasulyesi Etkinleştirme İşçatısı),
- Güvenlik servisleri.

İyi bir uygulama sunucusunun sağlaması gerektiği genel özellikler şunlardır:

- Ölçeklenebilirlik:
  Ölçeklenebilir yazılım parçası tamamen yeniden tasarım gerektirmeden bir veya daha fazla istemci ile eşit olarak uğraşabilen yazılım parçasıdır. Sistem tarafında daha fazla istemci için daha fazla kaynağa gereksinimi oluşurken uygulama tarafında böyle bir ihtiyaç söz konusu değildir.

- İstemci bilinemezliği:
  Uygulama sunucusuna istemci talebinin hangi protokol ve istemci uygulaması ile geldiğinden bağımsız olarak uygulamanın talebi cevaplayabilmesidir.

- Servis Yönetimi:
  Çok farklı kaynak bir tek bağdaşık sistem oluşturmak için birlikte çalışır. Yöneticiler tarafından servis ve kaynakların tek bir konsol üzerinden yönetilmesi arzulanır. Gelecek planlama yapabilmek için konsol aynı zamanda çeşitli açılardan sisteme çalışma anı bilgisi de sunmalıdır.

- Geliştirme:
  Mevcut uygulamalara aşırı derecede tesir etmeden yeni uygulama geliştirilebilmeli. Önemli BGO'lar (Bütünleşik Geliştirme Ortamı) ile uyumlu olmalı.

JİDA ile denetimi yapılan kişiler belirlenen yetkileri doğrultusunda JYKS ile belirlenen sistem ve kod çalıştırma izinleri etrafında sistemle etkileşir. JVTB ağ uygulamasının veritabanı ile olan haberleşmesini kontrol eden servistir. Yukarıda verilmiş JGS bileşenleri içinden GDYS sistemi içinde kullanılan JYKS bölüm 2.2.1.5'de, JİDA bölüm 2.2.1.6'da ve JVTB bölüm 2.2.3'de incelenmiştir.

2.2.1.5 JYKS

Java, güvenliği temelden tasarlayan ilk programlama ortamlarından biridir. İnsanoğlunun, sistem bütünlüğü ile uzlaşan ve sistemi sömürmek için gizli yolları bulmak için değişik yolları bulma ilgisinden dolayı güvenlik çözümlerinin gelişmesi hiçbir zaman durmaz. JYKS, saldırı önleme görevini kolaylaştırmak için Java ortamına katılmış en son araçlardan biridir.

Güvenlik planı içindeki gerekliliklerden biri sistem kaynaklarına yetkisiz erişimlere engel olmaktır. JYKS güvenli giriş ve izinler servisi sağlar. JYKS aynı zamanda çoklu alanlara karşı tek giriş ile yetkilendirme desteği verir. Tek giriş ile sistemin değişik kısımlarında yetkiye göre izinler sağlanır. Bu izinler doğrultusunda eylemler gerçeklenir.

Java sistemleri bütün olarak geniş alanda sunulurlar. Birçok yerel ağ bağlantıları ve geniş alan ağ bağlantıları kullanılır. Bu sebeple ağ bağlantılarına ve özel tip bağlantılara ihtiyaç duyulur. Güvenlik standart araçlar ve Java dilinin kendisi ile sağlanır. Java dili ile sağlanan güvenlik çözümlerine örnekler içinde derlenmiş kod (byte-code) doğrulama, çalışma zamanı tip kontrolü, sınıf yükleyicileri verilebilir. Geniş ölçekli sistemlerde müşteri ve personel dosyaları en önemli varlıklardır. Bunların izinsiz misafirlerden saklanımı çok önemlidir. İzinsiz misafirlerin Java dili ve nesne yönelimli programlamanın getirdiği dezavantajlardan faydalanarak aşağıdaki eylemleri gerçekleyebilir:

- Nesne kopyalama,
- Serileştirme ve serileştirmeme,
- İmzalama.

Bunlara karşı writeObject ,readObject, clone yöntemleri yeniden tanımlanıp güvenli sertifika sağlayıcılarının sertifikaları ile imzalanıp sertifika patikaları kullanılabilir.

Kullanıcıları tehlikeli sistemlerin ataklarından korumak için geliştirilmiş Java sandbox, applet'leri çalıştırmak için kullanılırlar. JYKS ise büyük ölçekli sistemleri tehlikeli kullanıcılardan korur ve bunun için aşağıda listelenmiş iki temel mekanizmayı kullanır:

- Kimlik denetimi hizmeti:
  Kullanıcının sistemden faydalanması için ismi ile bilinen kullanıcı olup olmadığını belirler. Denetlenen kullanıcı ise Java giriş bağlamına kayıt edilir. Kullanıcı bilgileri ve izinleri ise sistem içinde tutulur.

- Yetkilendirme hizmeti:
  Yetkilendirme hizmeti, kullanıcı ile kullanıcıdan kullanıcıya ve düzeylerine göre değişebilen izin kümesini birleştirir. İzinler, işletim sistemi üzerinde eylemleri ve dosya erişimlerini kısıtlamadan veritabanı erişimine kadar çeşitli kısıtlamaları belirler. Aynı zamanda kullanıcıları belli sertifika ile eşleyip çalıştırılan kodun son çalışan ile aynı olması sağlanır.

Harici dosyalar ile belirlenen eklenti (plugin) modül yapısına sahip JYKS, sağladığı bu esneklik ve taşınabilirlik sayesinde Java uygulamalarındaki mevcut güvenlik mekanizmasına bütünleştirilebilir.

JYKS özellikli uygulamalar 3 temel fonksiyonu icra eder:

- Uygulama, giriş bilgilerini ya kullanıcıdan açıkça talep eder veya kullanıcı giriş bağlamından alır. Bu bilgiler ya yetki ya yerel uygulamadan ya da genel sertifika yayıncısından alınır.
- Uygulama doAs() veya doAsPrivileged() yöntemini çağırır. Bu yöntem içinde kullanıcı bağlamı ve çağırılacak modül referansları vardır.
- Güvenlik yöneticisi kullanıcının yeterli yetkiye sahip olup olmadığını doğrular ve modülü çağırır veya isteği geri çevirir. Modül kullanıcı bilgilerini kullanarak günce tutar. Günce geri dönüştürme işleminde de kullanılabilir.

JYKS giriş sürecinin 2 katmanlı yapısı yarış koşulları veya modül hatası ile doğru giriş işlemi atlatılamaz. Güvenlik yönetimi görece karmaşıktır ve yapılması gereken eylemler JYKS ile basitleşir.

JYKS, güvenlik alanlarına (security realm) destek sağlar. JYKS güvenlik alanları kullanıcı ve sistemlerin mantıksal gruplarıdır. Alanlar bir makine üstünde tanımlı kullanıcılar olabildiği gibi bir birimde veya iş grubunda çalışan kişiler olabilir. Windows açısından alan kabaca bir Windows alana eşdeğerken, diğer işletim sistemlerinde dağıtık sunucular üstüne yerleştirilmiş Kerberos ve HDEP sunucular üstündeki kendi güvenlik disiplinleri olan sanal alanlar olabilir. Java güvenlik politikaları, güvenlik alanlarına farklı kod parçaları ile eşleşen farklı yetkileri sağlama desteği verir. Kullanıcılar keyfi yığınlar içine sistemin ihtiyaç duyduğu kısmi izinleri esnek olarak topluca verip geri almak için gruplanırlar. Standart güvenlik modeline grup veya ferdi yetkilendirme yapmak için JYKS güvenlik yöneticisi çağırımı eklenmiştir. JYKS güvenlik alanlarına kullanıcının giriş teşebbüsünden önce kendi ihtiyaçlarını yayınlamasına izin verir. Sunucudan toplanan bilgiye göre istemci giriş sürecini değiştirebilir. Eğer çoklu alan desteklendi ise istemcinin bu alanlardan birini seçebileceği menü sağlanır. Bunun sonucu kullanıcın kayıtlı olduğu tüm alanlarda yetkilendirme yapılması gerekmez. Aynı zamanda JYKS ile bezenmiş üçüncü parti satıcıların şık modülleri ile alternatif çözümler üretilebilir.

Geniş çaplı sistemlerde, kullanıcının her bir servis kullanımı için tek bir giriş bilgisini kullanmasını sağlayan çeşitli sistemler geliştirilmiştir. Bunlardan biri MIT'de kampus projesi için kampus şebekesindeki kaynaklara yama olmayan bir çözümle erişimi sağlayan KERBEROS'dur.

JYKS'ın diğer giriş güvenlik yöneticilerinden (Microsoft Kerberos gibi) farklı olarak herhangi bir özel plan seçimine ihtiyacınızın olmamasıdır. Planların her biri bir veya daha fazla JYKS prensibi ve bunlar ise gerektiği kadar kullanıcı kimliği yaratmak için belirli JYKS konusu ile birleşir. JYKS kullanarak giriş sürecini ilerletmek için dağıtık veya yerel şifre yetkilendirme protokolü kullanılır ve bu kimlik bilgilerinin merkezi yönetimini sağlar.

JYKS sisteminin temel altyapısı Java.security dosyasındaki 4 özelliktir:

- login.configuration.provider: JYKS'ın beklediği temel güvenlik formatını belirler.
- login.config.url.n: n bir pozitif sayı olup her bir giriş yapılandırma dosyasını gösterir.
- policy.provider: Java'nın beklediği güvenlik yapısını belirler.
- policy.url.n: her bir politika dosyasını gösterir. Birden fazla politika dosyası mevcut ise bunlar yekpare politika oluşturmak için birbirine eklenir.

JYKS'ın kendisi javax.security altındaki paketlerden, com.sun.security.auth.module altından bir geri çağırım yöneticisine ihtiyaç duyar. Aşağıda geri çağırım yöneticilerinin yazılıma eklendiği Java kod parçası mevcuttur:

```
import javax.security.auth.*;
import javax.security.auth.login.*;
import javax.security.auth.callback.*;
import com.sun.security.auth.callback.TextCallbackHandler;
import com.sun.security.auth.callback.DialogCallbackHandler;
```

Geri çağırım yönetici, kullanıcı ile olan etkileşimi yönetir ve aynı zamanda kullanıcı/şifre etkileşimini başlatır.

JYKS gücünü TKM'dan (Takılabilir Kimlik Denetimi) alır. Genişleyebilirlik JYKS giriş yapılandırma dosyası içinde inşaa edilmiştir. JYKS giriş yapılandırma dosyası kimlik denetimi yapılırken çağırılacak giriş modüllerini listeler. Modüller takılıp çıkartılabilir yapıda olmaktan hariç birbirinin yerine de kullanılabilir yapıya sahiptirler. Unix için kullanılacak giriş yapılandırma dosyası ile win2k için kullanılacak giriş yapılandırma dosyası yer değişebilir. Giriş yapılandırma dosyası aşağıdaki yapıya sahiptir:

```
<application name> {
        <LoginModuleName> <flag> <LoginModule options>;
        <LoginModuleName> <flag> <LoginModule options>;
        . . .
};
```

Bayraklar her bir modülün özelliklerini ve tüm denetim sürecinin kontrolünü belirler:

- Gerekli (Required): Modül listesi içindeki her bir başarılı üye denetlenir çok katlı giriş süreci için kullanılır.
- Zorunlu (Requisite): Listedeki modül başarısız olursa kontrol o an listenin sonuna gider ve giriş süreci başarısız olur.
- Yeterli (Sufficient): Listedeki modül başarılı olursa kontrol o an listenin sonuna gider ve giriş süreci başarılı olur.
- Seçimlik (Optional): Modül başarılı ya da başarısız olsa da kontrol listedeki bir sonraki modüle geçer.

Karmaşık giriş süreçleri bu şekilde yapılandırılabilir. Karşılaşılan tüm gerekli ve zorunlu modüller giriş sürecinin başarılı olması için doğru sonuçlanmalı. Eğer gerekli ve zorunlu modül yoksa bir tane yeterli veya seçimlik modül doğru sonuçlanmalı. Her modül kendine ait politika (policy) dosyasına sahip olabilir ve bu ise bir kullanıcının değişen izinlere sahip olmasını sağlar.

Esas (Principal), kullanıcı hakkında bilgi taşır ve çeşitli izinleri içerir. Tek olmak zorunda değildir; çünkü aynı esas birden fazla kullanıcı tarafından sahip olunabilir. JYKS konu ise sistem kaynaklarına erişmek için Java güvenlik yöneticisine istek yapan varlıktır ve tekil olmak zorundadır. Esas aşağıdaki varlıkları tanımlayabilir:

- Bir gerçek kullanıcıyı,
- Login ID'yi,
- Sistem düzeyi grup tanımlarını,
- Kullanıcıların rollerini ve durumları tanımlayan grup tanımlarını,

- Sistem servisi ve uygulama modüllerini,
- Tüm sistem veya bir birimi.

Sun tarafından com.sun.security.auth.module içinde aşağıdaki giriş modülleri sağlanmıştır:

- Krb5LoginModule,
- NTLoginModule,
- UnixLoginModule,
- KeyStoreLoginModule,
- JndiLoginModule.

GDYS sisteminde JndiLoginModule kullanılmıştır. Kullanıcı denetimi LoginContext sınıfı etrafında giriş yapılandırması içinde belirlenmiş giriş modülü kullanılarak yapılır. Her bir kullanıcı için kimlik/şifre/izin üçlüsünün kontrolü yönetici için oldukça zaman tüketici güç bir işlemdir. Bu işlem yerine konacak önceden belirlenmiş giriş süreci ise sistemde kırılgan yapı oluşturur. Giriş süreci giriş bağlamı içinde yapılacak geri çağırım yöneticisi çağrısı ile kullanıcı bilgilerinin toplanıp bu çağrı sırasında belirlenmiş modül tarafından denetimin gerçekleştirilmesi ile devam eder. Giriş sürecini modelleyen kısa kod parçası aşağıda sunulmuştur:

```
LoginContext lc = null;
try {
        lc = new LoginContext(“ObtainLogin”,new DialogCallbackHandler());
} catch (LoginException le) {
        System.err.println(“Cannot create LoginContext: “+ le.getMessage());
        System.exit(-1);
} catch (SecurityException se) {
        System.err.println(“Cannot create LoginContext: “+ se.getMessage());
        System.exit(-1);
}
```

```
try {
// attempt authentication
        lc.login();
} catch (LoginException le) {
        System.err.println("Authentication failed: ");
        System.err.println(" " + le.getMessage());
        System.exit(-1);
}
```

Daha sonra kullanıcı yetkilendirmesi için JYKS yetkili uygulama kullanıcı bilgilerini, izinlerini ve sertifika bilgisini çeşitli kaynaklardan çeker. JYKS politika dosyası olmaksızın bu işlem hiçbir anlam ifade etmez. JYKS politika dosyası Java politika dosyası gibi kod çalıştırma iznini belirlemekten başka aynı zamanda o kodu çalıştıracak kişiye ve koda çeşitli izinlerin verilmesini de belirler. Aşağıda sunulan ilk yetki verme (grant) ObtainLogin.jar kodu çalıştırma yetkisini tanımlarken ikinci yetki verme içindeki Esas ile belirlenebilecek kullanıcılara GetOsName.jar kodunun çalıştırma yetkisi verilmiştir.

```
grant CodeBase "file:./ObtainLogin.jar" {
        permission javax.security.auth.AuthPermission
"createLoginContext.ObtainLogin";
        permission javax.security.auth.AuthPermission "doAsPrivileged";
};

grant CodeBase "file:./GetOsName.jar",
        Principal com.sun.security.auth.NTUserPrincipal "Puddintane" {
                permission java.util.PropertyPermission "java.home", "read";
};
```

JİDA giriş yöneticisi yapılandırma için aşağıdaki seçenekleri kullanır:

- user.provider.url: JİDA tedarikçisinin yerini belirleyen BKÇ'ya işaret eder ve protokol formatı ldap://<uri> şeklindedir. Zorunlu seçenektir.

- group.provider.url: JİDA tedarikçisinin yerini belirleyen BKÇ'ya işaret eder ve protokol formatı ldap://<uri> şeklindedir. Zorunlu seçenektir.
- UseFirstPass: İki değerli seçenektir ve Kişi giriş yaptıktan sonra kullanıcı ve şifre bilgisini giriş modülünden alır ve yeniden denetim için kullanılır.
- TryFirstPass: İki değerli seçenektir ve Kişi giriş yaptıktan sonra kullanıcı ve şifre bilgisini giriş modülünden alır ve yeniden denetim için kullanılır.
- StorePass: Denetim ve teslim etme sonrası kullanıcı ve şifre bilgisinin giriş modülün paylaşılan bağlamına depolanmasını sağlar.
- ClearPass: Denetim ve teslim etme sonrası kullanıcı ve şifre bilgisinin giriş modülün paylaşılan bağlamından silinmesini sağlar.

JİDA giriş modülleri aşağıdaki Esasları döndürür:

- UnixPrincipal
- UnixNumericUserPrincipal
- UnixNumericGroupPrincipal

2.2.1.6 JİDA

Dağıtık uygulamalarda en önemli görevlerden biri çalışma anında gerekli sistem parçalarının yerlerinin bulunmasıdır. JİDA, JGS uygulamalarının önemli servislerinden biridir ki dağıtık bileşenlerin bulunması ile ilgili mekanizmayı sağlar (McGovern *et al.* 2003).

İsimleme (naming) hizmeti insan tarafından anlaşılan isimlerin makine tarafından anlaşılan isimlere dönüştürür. Dizin (directory) hizmeti ise isimlere ek özellik eklemeyi getirir. İsimleme hizmetine örnek olarak AİS (Alan İsmi Sistemi) hizmetini verebiliriz. AİS ağ bağlantılarında kullandığımız alan adların ip adreslerine çeviren hizmettir. HDEP protokolünün uygulamasını gerçekleyen dizin hizmetleri olarak NDS (Novell Dizin Servisi), Microsoft'un Active Directory'sini verebiliriz. Bir dizin insan tarafından anlaşılan isimleri uygulamalar tarafından kullanışlı olan bilgilere eşler. Örneğin şebedeki yazıcının adını o yazıcının şebedeki fiziksel adresine eşlemek gibi.

Dizinler hiyerarşik yapıda organize edilirler ve ağaç yapısına sahiptirler. Bu yapı ile fiziksel olarak ayrı yerlerde bulunan yapılaşma tek mantıksal örüntü altına toplanabilir. Tek mantıksal örüntü altındaki yapı farklı fiziksel yerlerdeki sunucular üzerinde sunulabilir. Her lokasyondaki sunucu kendi yeri için ana (master) sunucu görevini üstlenir ve diğer yerleşimlerdeki sunucuların üstündeki bilgiyi kendi üstüne kopyalar. Tek mantıksal örüntü yapısı ile yerel yerleşimlerdeki makineyi o yerleşimlerdeki sunucuya bağlayarak tüm yerleşimleri görmesini sağlanır.

Mevcut veri depoları yerine dizin yapısının kullanılmasının nedeni olarak aşağıdaki sebepler verilebilir:

- Dizinler yalnız okuma erişimi için optimize edilmişlerdir.
- Dizin veri üstünde belli bir yapıyı zorlar.
- Dizinler sıralanmışlardır.
- Dizinler dağıtıklardır.

X.500, ITU-T (Uluslararası Telekomünikasyon Birliği) tarafından ISO (Uluslararası Standartlar Örgütü) ile bağlantılı olarak geliştirilmiş uluslararası standartlar kümesidir. Prensip olarak bu standartlar çoklu dizin sisteminin diğerleri ile bağlanabilmesi ve bilgi değiş tokuşu yapabilmesi için tasarlanmıştır. Dizin hizmeti için DSA (Dizin Servis Ajanı) isim ve özelliklere ait bir veri tabanını kullanır ve idare eder. DKA (Dizin Kullanıcısı Ajanı) ise kullanıcılar için DSA'nin kullanımını idare eder ve DKA, DSA ile haberleşmek için DEP (Dizin Erişim Protokolü) protokolünü kullanır.

University of Michigan, 90'larda HDEP protokolünü geliştirmiştir (Howes *et al.* 1993). HDEP Dizin hizmetlerine erişmek için basit kelime temelli yaklaşımı kullanır. HDEP uygulamaları arasından aşağıdakiler en popüler olanlarıdır:

- SunONE Directory Server,
- Novell eDirectory (eskiden NDS),
- Microsoft Active Directory,
- OpenLDAP.

JİDA servisi bir HDEP sunucusuna ihtiyaç duyar. GDYS sisteminde HDEP sunucusu olarak kullanılan Openldap kurulumu, sistemle olan bütünleştirilmesi ve yapılandırılması takip eden bölümde ele alınmıştır.

2.2.2 OpenLDAP sunucusu

OpenLdap'ı çalıştırmak için slapd.conf yapılandırma dosyası düzenlenerek bir HVDŞ dosyası ithal edilmelidir. Slapd.conf dosyası içinde dizin yapısında kullanılacak veri yapısını tanımlayan şema belirlenmelidir. Dcm.schema GDYS sisteminin kullandığı şemayı tanımlar. Kullanılan özellikler ve bunların veri tipleri bu dosya içinde tanımlanır. GDYS sisteminde Openldap uygulamasının isim ve özelliklerini depolamak ve dizin yapısının daha rahat bakımını sağlamak için standart BerkeleyDB yerine PostgreSQL veritabanı kullanılır. Aşağıda kullandığımız slapd.conf dosyası mevcuttur:

```
# $OpenLDAP: pkg/ldap/servers/slapd/back-sql/rdbms_depend/pgsql/slapd.conf,v 1.5
2005/01/05 15:23:01 ando Exp $
#
# See slapd.conf(5) for details on configuration options.
# This file should NOT be world readable.
#
include         /usr/local/etc/openldap/schema/core.schema
include         /usr/local/etc/openldap/schema/cosine.schema
include         /usr/local/etc/openldap/schema/inetorgperson.schema
include         /usr/local/etc/openldap/schema/dcm.schema
# Define global ACLs to disable default read access.

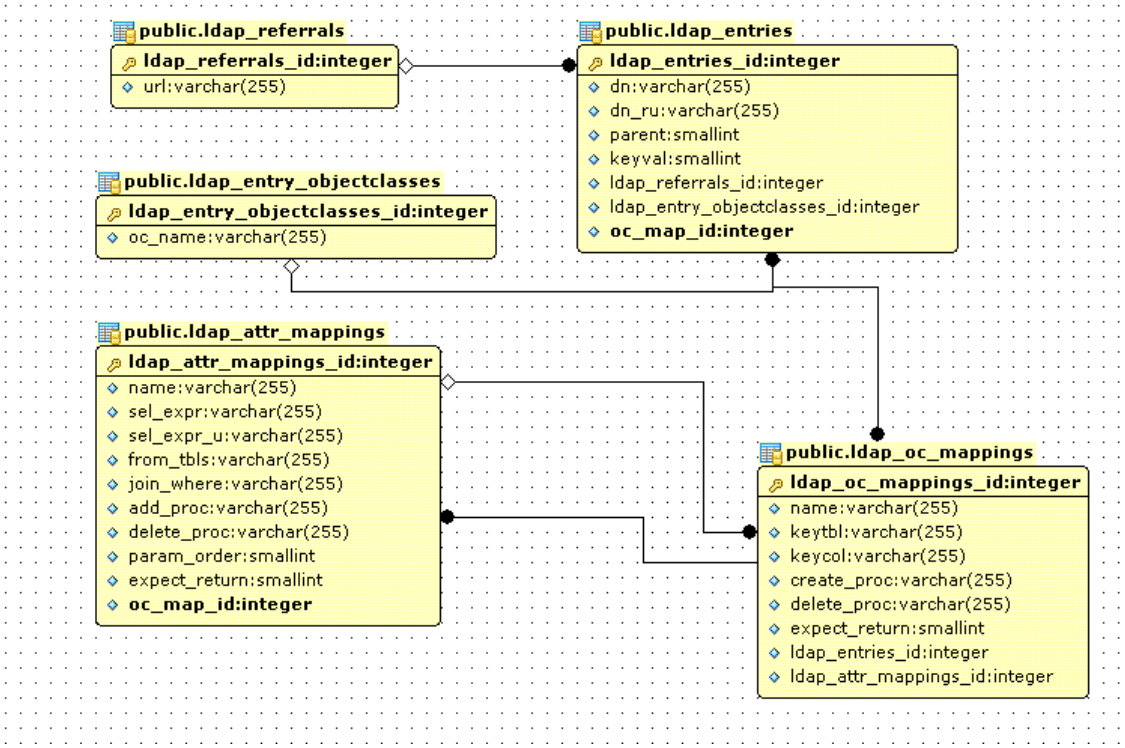
# Do not enable referrals until AFTER you have a working directory
# service AND an understanding of referrals.
#referral       ldap://root.openldap.org

pidfile         /usr/local/var/slapd.pid
argsfile        /usr/local/var/slapd.args
```

```
#######################################################################
#
# sql database definitions
#######################################################################
#

database        sql
suffix          "o=sql,c=RU"
rootdn          "cn=root,o=sql,c=RU"
rootpw          secret
dbname          dcm_db
dbuser          dcm
dbpasswd        dcm
insentry_stmt  "insert into ldap_entries (id,dn,oc_map_id,parent,keyval) values ((select
max(id)+1 from ldap_entries),?,?,?,?)"
upper_func      "upper"
strcast_func    "text"
concat_pattern"?||?"
has_ldapinfo_dn_ru    no
```

Yukarıdaki yapılandırma dosyasındaki veritabanı kısmındaki sql, veritabanı olarak bir İVTYS kullanıldığını anlatır. Dcm_db olarak adlandırdığımız veri tabanında OpenLdap'ın isim ve özelliklerini depolamak için kullandığı veri tabanı yapısı şekil 2.28'de görülebilir.



Şekil 2.28 OpenLdap'ın kullandığı veritabanı yapısı

OpenLdap'ın kullandığı ağaç yapısı ldap_entries tablosunda tutulur. Ağaç yapısının tanım bilgilerinin yapısını belirten şema sınıfı, ldap_oc_mappings tablosunda tutulur. Her ldap_entries kaydı için onun kullandığı şemayı tanımlayan ldap_oc_mappings kaydı mevcuttur. Ldap_entry_objectclasses ise birden fazla objectClass'a sahip ağaç yapısı elemanının diğer objectClass referanslarının tutulduğu tablodur. Ldap_attr_mappings, ldap_oc_mappings tablosu içinde tanımlı şemanın özelliklerinin tanım tablosundaki hangi alana denk geldiği ve nasıl okunup yazılacağı bilgisi tutulur.

HDEP şema, dizin ağaç yapısını belirleyen genel bir ara yüz tanımlar. Belirli bir şema için bazı standartlar mevcuttur. Örneğin inetOrgPerson için RFC 2798 gibi. HDEP dizin üyelerinin yapısı sabit özellik ve değer serisi olabildiği gibi uygulamaya özel şemada farklı anlamı, tipi ve değeri olan özellikleri tanımlanabilir. 2798 standardı ile tanımlı bazı özellikler aşağıda listelenmiştir:

- o Organization
- ou Organizational unit
- c Country

- cn Common name
- sn Surname
- givenname First name
- uid User ID
- dn Distinguished name
- dc Internet domain name
- mail E-mail address

DN ağaç yapısının tepesinden ağaçtaki her dalın yolunu tanımlar. Hiyerarşi çeşitli şekilde tanımlanabilir. En çok kullanılan yol ise internet alan isimlerinin kullanılmasıdır. Örneğin cn=ergin, dc=ankara, dc=edu, dc=tr. Ya da kurumun birimsel yapısı kullanılabilir. Örneğin uid=ergin, ou=bim, o=Ankara University. En soldaki kısım GAA (Göreceli Ayrıştırma Ad) olarak adlandırılır.

JİDA, Java uygulamalarına dizin ve isimleme hizmetleri için UPA temin eder. JİDA, HDEP, NDS, AİS ve hatta yerel dosya sistemi için genel bir ara yüz sağlar. Bununla birlikte HDEP için diğer Java UPA'ları geliştirilmiştir.

JİDA, dizin ve isimleme servisleri ile birlikte genel hiyerarşik ara yüz temin eder. Temel JİDA sınıfları javax.naming paketi ve alt paketleri altına yerleştirilmiştir. Bunlar aşağıda listelenmiştir:

- javax.naming: Basit isimleme hizmetlerine erişir.
- javax.naming.directory: Dizin hizmetlerine erişir.
- javax.naming.event: Dizin ve isimleme servisleri ile uğraşırken olay uyarılarını yönetir.
- javax.naming.ldap: HDEP v3 kontrolü ve genişletilmiş işlemleri ile uğraşır.
- javax.naming.spi: SSA (Servis Sağlayıcısı Ara yüzü)'dan oluşur.

GDYS uygulamasında ağ uygulamasının web.xml yapılandırma dosyasında <security-constraint> ve <login-config> elemanlarından faydalanarak tanımlanan ve Tomcat'in desteklediği kap tarafından yönetilen güvenlik çözümünü kullanılmıştır. Web.xml

elemanları ile güvenlik altına alınan ağ uygulaması kısımları ve ilgili roller, güvenlik yöntemi ve giriş yöntemi belirtilir. Aşağıda GDYS uygulamasındaki kullanılan güvenlik ile ilgili kısımdan alıntı vardır:

```
<security-constraint>
  <display-name>Search</display-name>
  <web-resource-collection>
   <web-resource-name>Search</web-resource-name>
   <url-pattern>/search/*</url-pattern>
   <http-method>PUT</http-method>
   <http-method>DELETE</http-method>
   <http-method>GET</http-method>
   <http-method>POST</http-method>
  </web-resource-collection>
  <auth-constraint>
   <role-name>admin</role-name>
   <role-name>personnel</role-name>
   <role-name>guest</role-name>
  </auth-constraint>
 </security-constraint>
 <login-config>
  <auth-method>FORM</auth-method>
  <form-login-config>
   <form-login-page>/login.jsf</form-login-page>
   <form-error-page>/error.jsf</form-error-page>
  </form-login-config>
 </login-config>
<security-role>
  <description>Personnel</description>
  <role-name>personnel</role-name>
 </security-role>
```

Alan, Tomcat uygulamasında kullanıcı, şifre veritabanı olarak ele alınır ve ek olarak kullanıcıya tahsis edilmiş rollere ait numaralandırma tutulur. Roller grup olarak ele alınabilir ve hangi rolün <security-constraint> ile belirlenen ağ kaynağına erişeceği belirlenir (Geary and Hortsmann 2007).

Kullanıcı bilgilerinin elde edileceği alana bağlantı, Tomcat'in sağladığı kabda uygulaması yapılmış takılabilen bileşen olan org.apache.catalina.Realm ara yüzünün uygulamaları olan bileşenlerle sağlanır. Örnek olarak JDBCRealm, DataSourceRealm, JNDIRealm, MemoryRealm, JAASRealm verilebilir.

Alan konfigürasyonu ise conf/server.xml dosyası içinde Motor, Sunucu, Bağlam kısımlarında <Realm className="... gerçekleme için sınıf ismi" ... gerçekleme için diğer özellikler.../> şeklinde tanımlanabilir.

GDYS uygulamasında JNDIRealm kullanılır. JNDIRealm, HDEP dizin hizmeti içinde kullanıcı aramak için faydalanılacak alan ara yüzü uygulamasıdır. Konfigürasyon dosyasında dizin hizmetinde bağlanılacak dizin connectionURL ile belirlenir. Birden fazla dizin hizmetine mevcut ise connectionURL ile belirlenmiş sunucuya bağlantı kurulamadığında bağlanılabilecek diğer sunucu altenateURL ile belirlenir. Bağlantı sağlandığı zaman alanın kendisinin yetkilendirilmesi için connectionName ve connectionPassword değerleri kullanılır. Yetkilendirilmesi yapılacak kullanıcının araştırılması için userBase, userSubtree, userSearch değerleri belirlenir. UserBase özelliği başlangıç dirContext olan connectionURL'nin altındaki aramanın yapılacağı tepe dizindir. UserSubtree, userBase altındaki tüm alt dizinlerde mi yoksa yalnızca tepe dizinde mi aramanın yapılacağını belirten ikili değerdir. UserSearch aramayı yapacak HDEP süzgecidir.

Kullanıcı yetkilendirme bind mode ve comparison mode kullanılarak yapılır. Bind mode'da alanın sağladığı değerler HDEP hizmeti tarafından ilk önce kullanıcının dn değeri ile HDEP ağaç yapısında kullanıcının bulunduğu dal bulunur ve o daldaki kriptolanmış şifre değeri alandan gelen düz yazı olan şifrenin HDEP tarafından kriptolanan değeri ile karşılaştırılır. Comparison mode'da ise alanın sağladığı dn değeri

ile kullanıcıya ait dal bulunur ve oradan kriptolu şifre değeri alınır. Bu değer alan için kullanıcının sağladığı şifre HDEP'in kullandığı algoritma kullanılarak kriptolanır ve gelen değerle karşılaştırılır. Bu ise güvenlik açığı doğurabilir.

Kullanıcı rolleri, rolün HDEP ağaç yapısında kullanıcının bulunduğu dalın özelliği olması veya rolün ağaç yapısının bir dalı olması yöntemi kullanılarak kullanıcıya tahsis edilir. İlk yöntemde userRoleName özelliği ile kullanıcıya ait roller belirlenirken ikinci yöntemde ise roleBase, roleSubtree, roleSearch, roleName özellikleri kullanılır. RoleBase, rollerin aranacağı dizini belirler. RoleSubtree, roleBase altındaki tüm dizinlerde mi yoksa yalnızca tepe dizinde mi aramanın yapılacağını belirler. RoleSearch, HDEP süzgecini belirler. RoleName, rol dalındaki rolün adını belirleyen özelliktir.

GDYS uygulamasında aşağıdaki gibi bir alan tanımı conf/server.xml dosyasının bağlam kısmında bulunmaktadır:

```
<Realm   className="org.apache.catalina.realm.JNDIRealm" debug="99"
    connectionURL="ldap://localhost:389"
        userBase="ou=people,dc=mycompany,dc=com"
      userSearch="(mail={0})"
    userRoleName="memberOf"
        roleBase="ou=groups,dc=mycompany,dc=com"
        roleName="cn"
      roleSearch="(uniqueMember={0})"
/>
```

JNDIRealm aşağıdaki kurallar etrafında çalışır:

- Kullanıcı korunumlu bölgeden bir kaynağa talepte bulunursa Tomcat kap o alanın authenticate yöntemini çağırır.
- Kullanıcı ve rol bilgileri, yetkilendirme sonrası giriş işlemi sürecinde Tomcat içine kullanıcı oturumu zaman aşımına uğrayana dek veya kullanıcı ağ tarayıcısını kapatana dek yedeklenir.

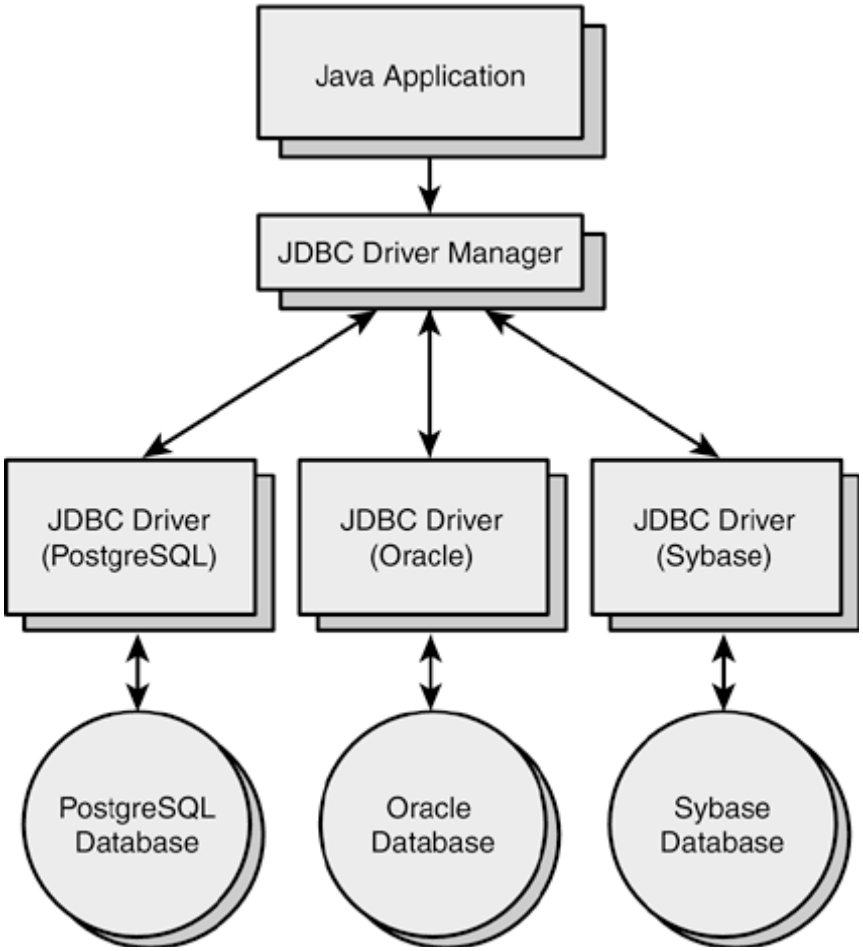
- Dizin içindeki bilgi yönetimi Tomcat sorumluluğu altında değildir. Bunu GDYS uygulaması HDEP'le ilişkili veri tabanı tabloları üstünden yapar.

Takip eden bölümde ağ uygulamasının veritabanı ile etkileşimini sağlayan ve JGS servisi olan JVTB incelenmiş; ağ uygulaması içinde nasıl kodlandığı anlatılmıştır. Devamında JVTB servisinin ihtiyaç duyduğu PostgreSQL sunucusunun kurulumu, yapısı ve yapılandırılması ele alınmıştır.

2.2.3 JVTB ve PostgreSQL sunucu

İş uygulamaları genellikle kullandıkları veriyi İVTYS içinde depolar. Veri birbiri ile ilişkili tablolar içinde tutulur. İki teknoloji kullanılarak Java programları içinden İVTYS içindeki ilişkisel veriye ulaşılabilir. İlki SQLJ denen ANSI (Amerikan Ulusal Standartlar Enstitüsü) ve ISO standardıdır. SQLJ ile Java programları içine YSD ifadeleri gömülür. İkinci yol ise JVTB denen Java Topluluğu (Community) tarafından belirlenmiş teknolojinin kullanımıdır (McGovern *et al.* 2003).

JSV (Java Standard Sürümü) ile JGS; java.sql ve javax.sql paketlerini içerir. Java programları java.sql ve javax.sql paketleri içindeki sınıfları kullanarak veri tabanına erişir. JVTB UPA' sının tüm veritabanları için aynı oluşu JVTB'nin sağladığı en büyük kolaylıklardan biridir ve uygulamayı diğer bir veri tabanına taşımak için o veri tabanına ait JVTB sürücü paketini veritabanı bağlantı dosyasına tanıtmak yeterli olacaktır.



Şekil 2.29 JVTB mimarisi

JVTB kütüphanesi, Java programları ile İVTYS arasındaki katmanı oluşturur. Şekil 2.29'da görüldüğü üzere veri tabanı ile Java uygulaması arasında bağlantı JVTB DriverManager ve JVTB kütüphanesinin uygulama ile etkileşimi sonucu oluşur. Kütüphaneler İVTYS satıcıları ve JGS uygulama sunucusu tarafından sağlanır. Sun her bir JVTB tanımlamalarının kütüphaneleri için referans uygulamasını temin eder. Kullanılabilecek 4 tip JVTB Kütüphanesi mevcuttur:

- **Tip 1:** JVTB-AVTB köprüsü, geniş olarak kullanılabilen AVTB (Açık Veritabanı Bağlantısı) kütüphaneleri üstünden veri tabanı bağlantısı oluşturur. Diğer bağlantı tiplerine göre yavaştır ve her istemci üstünde tek tek yapılandırılmalıdır. Bunlara karşın istemcide AVTB Kütüphanesi kurulu ise diğer bir sınıfa ihtiyaç yoktur.

- **Tip 2:** Java olmayan doğal kütüphane ile birlikte çalışan Java sınıflarından oluşan kütüphanelerdir. Kısmen Java Kütüphanesi olan veritabanı satıcıları tarafından sağlanılan ve her istemci üstünde kurulması gereken kütüphanelerdir. JVTB istekleri doğal VTYS kütüphaneleri çağrılarına dönüştürülür. Tip 2 kütüphanesi eğer JGS uygulaması için kullanılacaksa yalnızca uygulama sunucusu üstünde kurulması yeterlidir.
- **Tip 3:** JDBC-Net kütüphanesi, saf Java ağ kütüphaneleridir. Uygulama sunucuları satıcıları tarafından sağlanırlar ve iki kısımdan oluşurlar. İlk kısım istemci parçasıdır ve VTYS'e bağlı YSD çağrılarını icra eder. İkinci kısım YSD çağrılarının kendisidir ve bunlar ara katman satıcısının kullandığı protokole dönüştürülürler. Çeşitli veritabanları ile çalışma esnekliği sağlar. Bu tipin dezavantajı, pahalı çözüm olan bu esnek kütüphaneleri sağlayan uygulama sunucusu satıcıları ile uğraşılması gereklidir.
- **Tip 4:** Jar ve Zip uzantılı saf Java kütüphaneleri, doğrudan veritabanı sunucusu çağrıları yaparlar. İstemci makinesinde herhangi bir yapılandırılmaya ihtiyaç duymazlar ve dinamik olarak istemci makinesine indirilebilirler.

Bir veritabanı uygulamasında en çok zaman harcayan işlev, bağlantı kurulmasıdır. Java uygulamasında ise bu bağlantı nesnesi yaratımına karşılık gelir. Program bağlantıyı kapattığında ise bu nesne çöp toplayıcı için kaldırılacak bir aday olur ki bu işlemde programın performansına etki eder.

Nesne havuzları ise bağlantı nesnelerinin yeniden kullanımını sağlayarak çöp toplayıcı işlemini alt düzeylere çeker. Oldukça basit olan nesne havuzlarının arkasında fikir, bağlantı nesneleri gibi yeniden kullanılması istenen nesnelerinin bir kümesini yaratarak nesnelere ait referans değişkenlerini uygulamanın yaşam döngüsü boyunca canlı tutmaktır. Bununla birlikte referansları yöneten, getConnection denen ve havuzdaki bir sonraki kullanılmayan nesneyi döndüren, close denen ve nesneyi havuza geri döndüren iki yöntemi içeren tek örnekli (singleton şablonunu uygulayan) bir sınıfın yönetilmesi gereklidir. Bağlantı havuzları uygulama sunucusu ve veritabanı satıcıları tarafından sunulurlar. Uygulama sunucuları bağlantı havuzlarının yapılandırılması için uygulama

sunucusu JİDA ağaç yapısı içinde “connection pool” ve “data source” olarak adlandırılan iki dalı yaratmalıdırlar.

Tomcat uygulama sunucusu veritabanı bağlantı havuzu oluşturmak için aşağıdaki Jakarta-Commons bileşenlerini kullanır:

- Jakarta-Commons DBCP,
- Jakarta-Commons Collection,
- Jakarta-Commons Pool.

$CH/lib/tomcat-dbcp.jar içinde bulunan bileşenlerle birlikte GDYS uygulamasında veritabanı bağlantısı oluşturmak için PostgreSQL veri tabanı tarafından sağlanan $CH/lib/postgresql-8.2-508.jdbc3.jar dosyası kullanılır. DBCP’nin sınıf yükleyicisinin bağlantı sınıfını bulabilmesi için bu dizin içine yerleştirilmesi gerekir.

Tomcat uygulamalarının veritabanı bağlantısı oluşturmak için tüm Tomcat uygulamaları tarafından paylaşılan veya belirli bir uygulama tarafından kullanılan veri kaynağı tanımlanır.

Paylaşımlı veri kaynağı $CH/conf/server.xml dosyasının sunucu kısmında ve belirli bir uygulamaya ait veri kaynağı ise bağlam kısmında tanımlanır. GDYS uygulamasında bağlam kısmında tanımlanılan veri kaynağı aşağıda sunulmuştur:

```
<Context docBase="dcm" path="/dcm" reloadable="true"
source="org.eclipse.jst.jee.server:dcm">
        <Resource name="jdbc/PostgreDS"
                auth="Container"
                type="javax.sql.DataSource"
                username="dcm" password="dcm"
                driverClassName="org.postgresql.Driver"
                url="jdbc:postgresql://localhost:5432/dcm_db"
                maxActive="20" maxIdle="10" maxWait="-1"/>
```

```
</Context>
```

GDYS ağ uygulamasının veri tabanı bağlantısı yaparken hangi kaynağı kullanacağı $CH/webapps/dcm/WEB-INF/web.xml içinde aşağıdaki gibi tanımlanır:

```
<resource-ref>
  <description>postgreSQL Datasource example</description>
  <res-ref-name>jdbc/PostgreDS</res-ref-name>
  <res-type>javax.sql.DataSource</res-type>
  <res-auth>Container</res-auth>
 </resource-ref>
```

Ana veri tabanı bağlantısı uygulamaya ait genel sınıf hiyerarşisinde bulunan tr.edu.ankara.dao.MainDao içindeki aşağıda sunulmuş connect yöntemi ile yapılır:

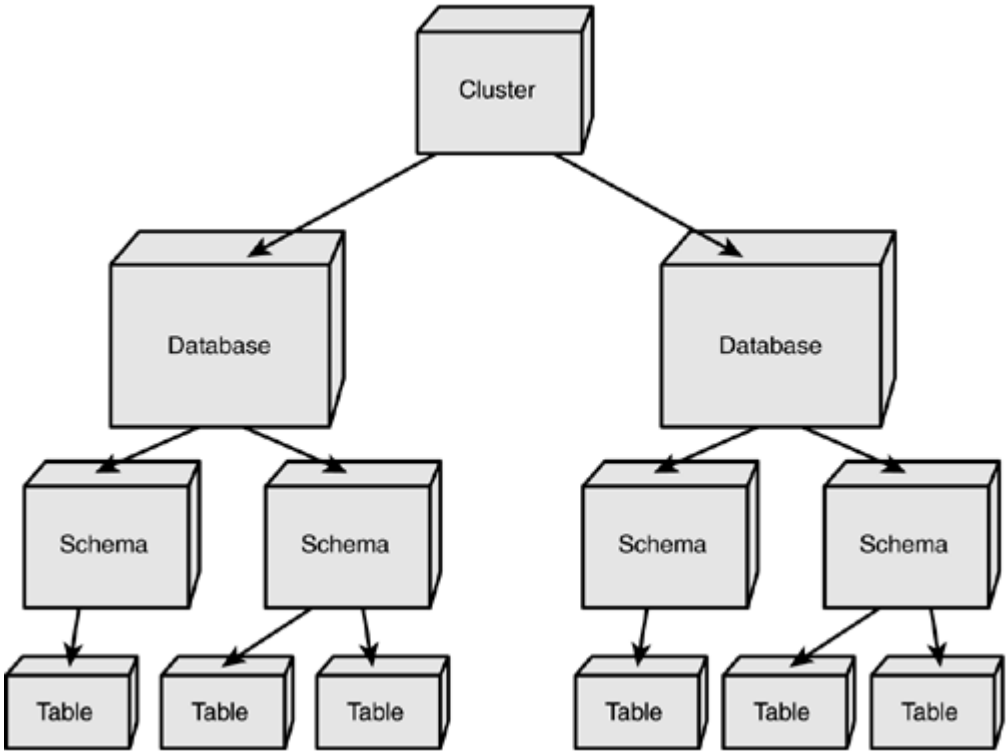
```
public String dsName = "jdbc/PostgreDS";
public void connect()
   {
                  try {
                        Context icxt = new InitialContext();
                        if ( icxt == null ) {
                          throw new Exception("No context!");
                        }
                        Context ecxt  = (Context)icxt.lookup("java:/comp/env");
                        if ( ecxt == null ) {
                          throw new Exception("No context!");
                        }
                        DataSource ds = (DataSource) ecxt.lookup( dsName );
                        if ( ds == null ) {
                          throw new Exception("Data source not found!");
                        }
                        conn = (Connection)ds.getConnection();
```

```
                conn.setAutoCommit(false) ;
            } catch (Exception ex) {
        System.out.println ("sql exception: " + ex.getMessage ());
                ex.printStackTrace();
                conn = null ;
            }
    }
```

İlk olarak java:/comp/env bağlamının bulunması için JİDA araması yapılır. Bulunan bağlamda veri kaynağına referansı elde etmek için ikinci bir JİDA araması daha yapılır. Veri kaynağı ile belirlenen havuzdan bağlantı alınarak gerekli veri tabanı etkileşimi gerçeklenir.

GDYS sistemi veritabanı olarak PostgreSQL veri tabanını kullanır. PostgreSQL 1970'lerin başında Berkeley'de atası olan Ingres ile hayata geçmiş ve ticarileşen Ingres Computer Associates tarafından satın alınmıştır. Berkeley'den Michael Stonebraker 1986'lı yıllarda Ingres veritabanına nesne ilişkisel özellikler eklemiş ve PostgreSQL oluşmuştur. Andrew Yu ve Jolly Chen 1990'ların ortalarında YSD desteğini ekleyerek PostgreSQL şu an bilinen halini almıştır. Halen PostgreSQL açık kaynak grup olan PostgreSQL küresel geliştirme grubu tarafından geliştirilir (Douglas and Douglas 2005).

PostgreSQL, açık kaynak kodlu, istemci/sunucu mimarisine sahip ilişkisel veritabanıdır. Önemli ticari veri tabanlarının sağladığı özelliklerle karşılaştırılabilecek güçlü özellikler kümesi sunar. En önemli faydası satın almak ve bakım için ödeme yapmak zorunda olunmamasıdır. Kalıtım özelliğini sunması, kullanıcıya ait veri tiplerinin PostgreSQL'e eklenebilmesi ve temel veri tiplerinin eklenebilmesi, geometrik veri tiplerinin desteklenmesi ve indislenebilmesi, seçilen dil doğrultusunda yeni operatör ve fonksiyonların eklenebilmesi, örneğin C, C++, Java, Python, Perl, Tcl/Tk ve diğerleri gibi çok farklı programlama dili ile istemci/sunucu mimarili istemci uygulaması inşaa etme ve PL/pgSQL gibi yordamsal dillerle yazılmış yordamların eklenebilmesi gibi birçok güçlü özelliği içerir.



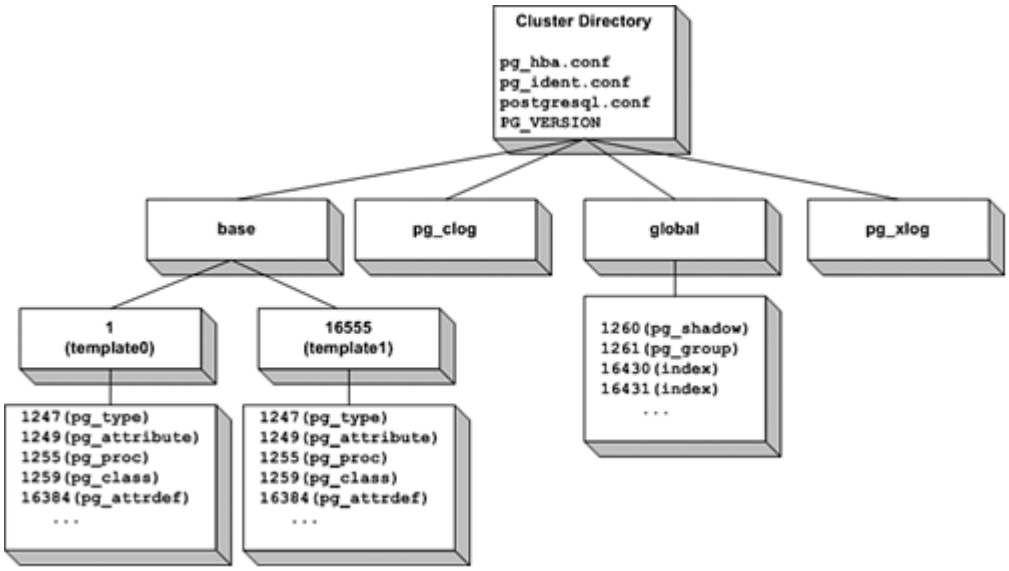
Şekil 2.30 PostgreSQL depolama hiyerarşisi

PostgreSQL'in şekil 2.30'da görüleceği üzere verisini en altta tablolar içinde tutar. Tablolar şemalar içinde gruplanır, şemalar ise veritabanı içinde yönetilir. Veritabanlarının oluşturduğu koleksiyon demet (cluster) olarak adlandırılır. Her demet kendine ait bir dizin ağacına sahiptir ve bir tek postmaster servisi ile yönetilir. Demetin bir ismi olmadığı gibi her demet içinde tüm sistemce paylaşılan sistem tabloları (pg_database, pg_shadow, pg_group) mevcuttur. &PG_DATA ortam değişkeni, demet için kök dizinin yerini belirler. Demet içindeki her veri tabanı için &PG_DATA/base altında pg_database sistem tablosundaki oid'si ile belirlenen bir dizin mevcuttur onların altında ise her tablonun ismi pg_class sistem tablosundaki oid'si ile belirlenen kendine ait dosyaları mevcuttur.

PostgreSQL veritabanın kurulumu seçilen platforma göre değişir. Unix benzeri sistemlerde kaynak koddan derleme ve paket yöneticileri ile kurulum kullanılırken Windows'ta ise derlenmiş binary'leri içeren kurulum yazılımları tercih edilir. GDYS sistemi Unix benzeri işletim sistemi olan Ubuntu Linux distrosu üstünde geliştirilmiştir.

PostgreSQL veritabanı Ubuntu üstündeki APT paket yöneticisi kullanılarak aşağıdaki komut ile kurulmuştur:

#> apt-get install posgtresql



Şekil 2.31 Kurulum sonrası dosya yerleşimi

&PG_DATA ortam değişkeni ile işaret edilen dizin en üst kurulum dizinidir. İşaret edilen dizindeki pg_hba.conf dosyası sunucu temelli denetim dosyasıdır. Pg_hba.conf dosyası ile PostgreSQL istemcileri sunucu temelinde kimlik denetiminin nasıl yapacağını anlar. Pg_ident.conf dosyası işletim sistemi kullanıcıları ile PostgreSQL kullanıcıları arasındaki eşleme şemasını anlatır. Postgresql.conf, PostgreSQL sunucusunu çeşitli açılardan kontrol eden çalışma anı parametrelerinin listesini içerir. PG_VERSION, initdb'den gelen sürüm numarasını içeren basit tekst dosyasıdır. Pg_xlog dizini ileri-yazım güncelerini içerir. PostgreSQL bir veri tabanı tablosundaki kaydı değiştirdiğinde ilk önce günce içine değişikliği kayıt eder belli bir zaman sonra bu değişiklik gerçek disk üstündeki o tablo ile ilişkili dosyaya ait sayfalara kayıt edilir. Pg_clog teslim etme güncelerini tutar. Global dizini tüm sistemce paylaşılan pg_shadow, pg_group, pg_database tablolarının dosyalarını içerir.

Initdb betiği ile veritabanı demeti yaratılırken kalıcı yapılandırma seçimlerinin bulunduğu postgresql.conf dosyası yaratılmış olur. Bu dosya isim=değer çiftlerinden

oluşur ve # karakterini izleyen tekst sunucu tarafından ihmal edilir. GDYS sisteminin kullandığı postgresql.conf dosyasının kullandığı önemli değerler aşağıda sunulmuştur.

```
port = 5432
max_connections = 100
shared_buffers = 1000                   # min 16 or max_connections*2, 8KB
log_destination = 'stderr'              # Valid values are combinations of
redirect_stderr = on                    # Enable capturing of stderr into
log_line_prefix = '%t '                 # Special values:
stats_start_collector = on
stats_row_level = on
autovacuum = on                         # enable autovacuum subprocess?
lc_messages = 'C'                       # locale for system error message  strings
lc_monetary = 'C'                       # locale for monetary formatting
lc_numeric = 'C'                        # locale for number formatting
lc_time = 'C'                           # locale for time formatting
```

GDYS sunucu temelli kimlik denetimi sırasında aşağıdaki pg_hba.conf dosyası içindeki önemli kısımları kullanır:

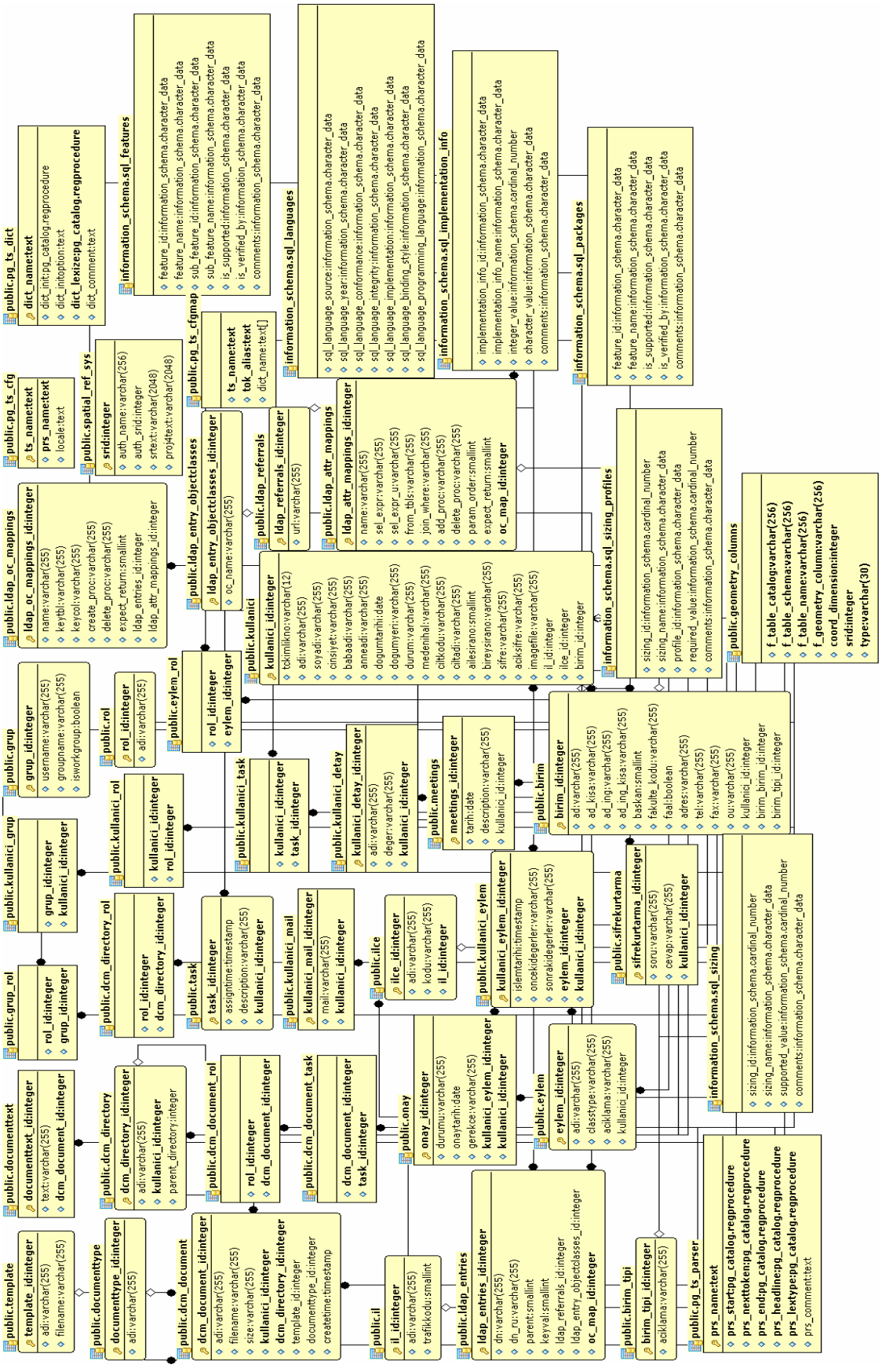
```
# TYPE  DATABASE    USER        CIDR-ADDRESS          METHOD
# IPv4 local connections:
host    all         all         127.0.0.1/32          md5
# IPv6 local connections:
host    all         all         ::1/128               md5
host    all         dcm         192.168.0.1/32   ident
```

GDYS, pg_ident.conf içinde bir isim=değer çiftine sahip değildir. Fakat pg_hba.conf içindeki kullanıcı kısmındaki isme karşılık gelen MAPNAME, işletim sistemi kullanıcı ismine karşılık gelen IDENT-USERNAME ve PostgreSQL kullanıcısı ismine karşılık gelen PG-USERNAME üçlüsünden oluşan satır eklenerek işletim sistemi temelli kimlik denetimi kullanılabilir. Aşağıda örnek bir satır mevcuttur:

```
# MAPNAME     IDENT-USERNAME    PG-USERNAME
dcm                 dcm_os                  dcm_pg
```

Yapılan tasarım çalışması sonucu GDYS sistemi için gerekli veri tabanı yapısı şekil 2.32'de sunulduğu gibidir.

Test vakaları ile sistemin isterlerinin karşılanıp karşılanmadığı incelenir. Tamamlanan GDYS sisteminin isterlerine ait iş kullanım vakaları için test vakaları bulgular bölümünde ele alınmıştır.



Şekil 2.32 GDYS sistemi varlık ilişki diyagramı

3. BULGULAR

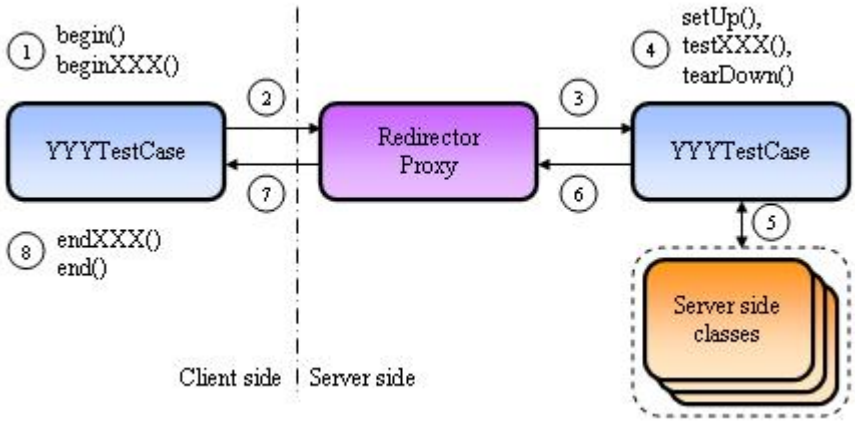
GDYS sistemi için geliştirilen kullanıma yönelik testler ve sistemin kullanılması sonucu elde edilecek dokümanların kalitesi sistemin başarısını gösterir. Geliştirilen GDYS sistemi ile ilgili olarak elde edeceğimiz sonuçlar sisteme uygulayacağımız test vakalarının sonucunda elde edilir.

3.1 Test Vakaları

Test vakaları ile sistemin isterlerinin karşılanıp karşılanmadığı incelenir. Test vakaları yazımında otomatize edilmiş test ve kullanım testleri gibi iki farklı yöntem belirlenebilir.

3.1.1 Cactus

Otomatize edilmiş test vakalarında kullanıcı test süreci içinde bulunmaz. Test için yazılmış işçatıları kullanılarak test programcıkları geliştirilir. Örnek olarak Apache grup tarafından geliştirilen Cactus işçatısı verilebilir. Cactus işçatısı Junit olarak bilinen işçatısı üstüne JGS ağ uygulamalarının test edilmesi için geliştirilmiştir. ServletTestCase, JspTestCase, FilterTestCase sınıflarından türetilen sınıflar yardımı ile testler gerçeklenir. Cactus'un çalışma mekanizmasını anlatan diyagram şekil 3.1'de verilmiştir.



Şekil 3.1 Cactus çalışma mekanizması

İstemci tarafından test vakası başlatıldığında TestCase sınıfı türevi için runTest yordamı çağrılır. RunTest yordamı istemci tarafında sırası ile begin, http istemi için parametrelerin belirlenmesini sağlayan beginXXX yordamlarını çağırır. XXX ise yazılan vakanın adıdır. Yönlendirici (Redirector) vekile açılan bağlantı ile aktarılan istem sunucu tarafında TestCase türevi sınıfının yeniden oluşturulup setUp ve vakanın uygulamasını yapan testXXX yordamlarının çağırılmasına sebep olur. TestXXX içinde assert, fail ve assertEquals gibi fonksiyonlarla vakanın durumu kontrol edilir. Cevap istemci tarafına döndüğü an son kontrollerin yapılması için endXXX ve end yordamları çağırılır.

3.1.2 Elle yapılan test vakaları

Elle yapılan test vakalarında ise sistem analizi kısmında verilen kullanım vakalarını sistemin tatmin edip edemediğini anlamak üzere çeşitli test vakaları oluşturulur. Test vakası kararın verilebilmesi için çeşitli şartlar ve değişkenler içerir. Tipik test vakası yapısı aşağıdaki bileşenleri içerir:

- Test Vakası No,
- Test Vakası Adı,
- Test Vakası Açıklaması,
- Test Vakası Konfigürasyonu,
- Test Vakası Başlangıç Şartları,
- Test Vakası Çalışma Adımları,
- Beklenilen Davranış ve Çıktılar,

Yukarıdaki değişkenler ile uygulanacak test vakaları tanımlanır. Sisteme uygulanmaları sonucu elde edilecek sonuçlar Test Vaka No, Test Vaka Adı , Beklenilen Davranış Gerçeklendi mi? ve Gözlemlenen Davranış sütunlarını içeren Test Vaka Sonuçları Tablosu ile verilir.

GDYS sisteminin isterlerine ait test vaka sonuçları tablosu ile karşılaştırılarak sistemin tamamlanıp tamamlanmadığına karar verilir. Bir sonraki bölümde GDYS sistemine ait

test vakaları listelenmiştir.

3.2 GDYS Sistemi Test Vakaları

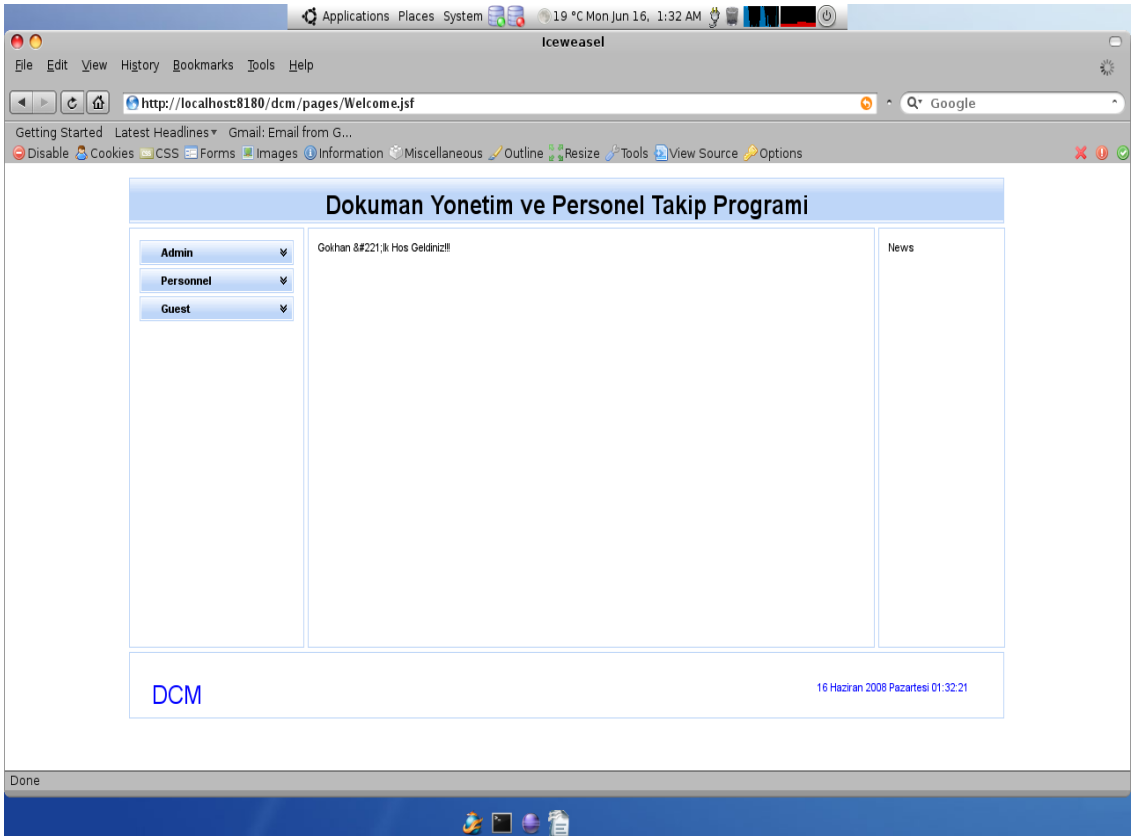
Test Vaka No: T1

Test Vakası Adı: Sisteme giriş

Test Vakası Açıklaması: Sisteme giriş işlemi doğrulanır.

Test Vakası Çalışma Adımları: http://<TomcatGDYSSunucusu>/dcm ile sisteme girilir. Kullanıcı adı olarak TC. kimlik no ve şifre girilir (Şu an için deneme kullanıcısı ile şifresi kullanılabilir. gokhan/tomcat).

Beklenilen Davranış ve Çıktılar: İndis sayfasında menü ile kullanıcın adı ve soyadını içeren karşılama mesajı görülmelidir. Menü kısmı admin, personnel, guest olmak üzere 3 kısımdan oluşur. Şekil 3.2 görülmelidir.



Şekil 3.2 İndis sayfası

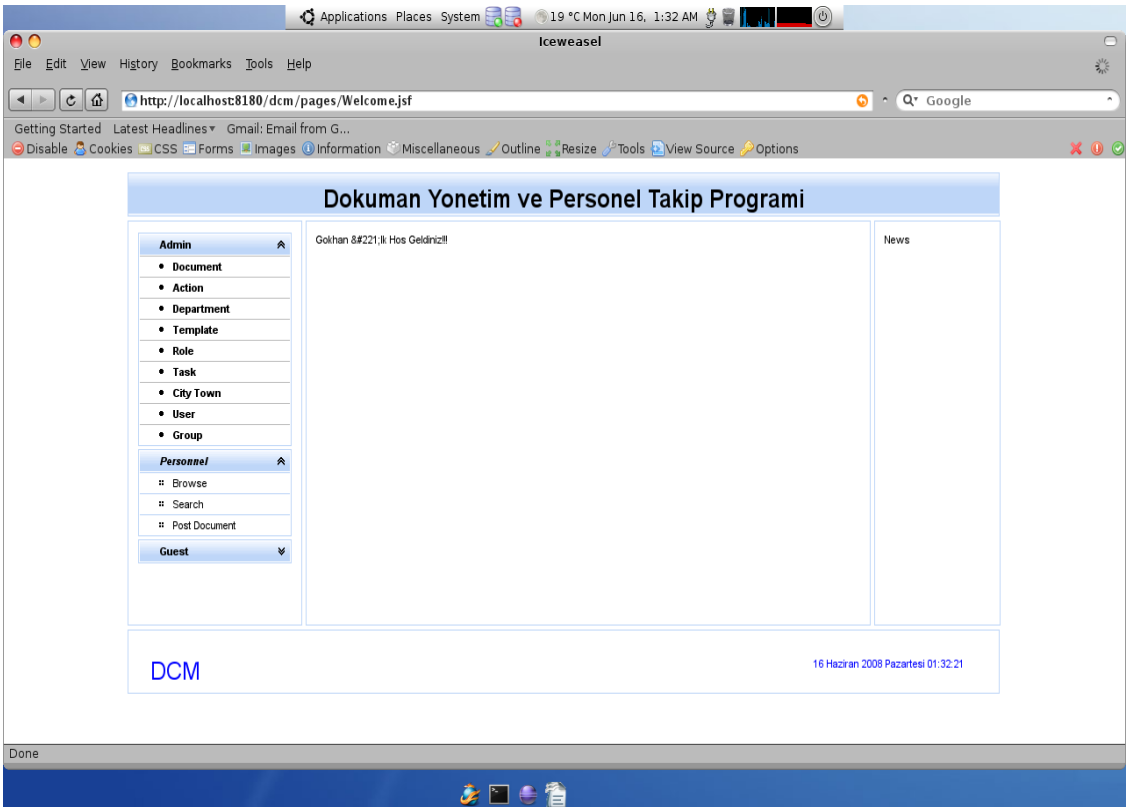
Test Vaka No: T2

Test Vakası Adı: Kayıt listeleme

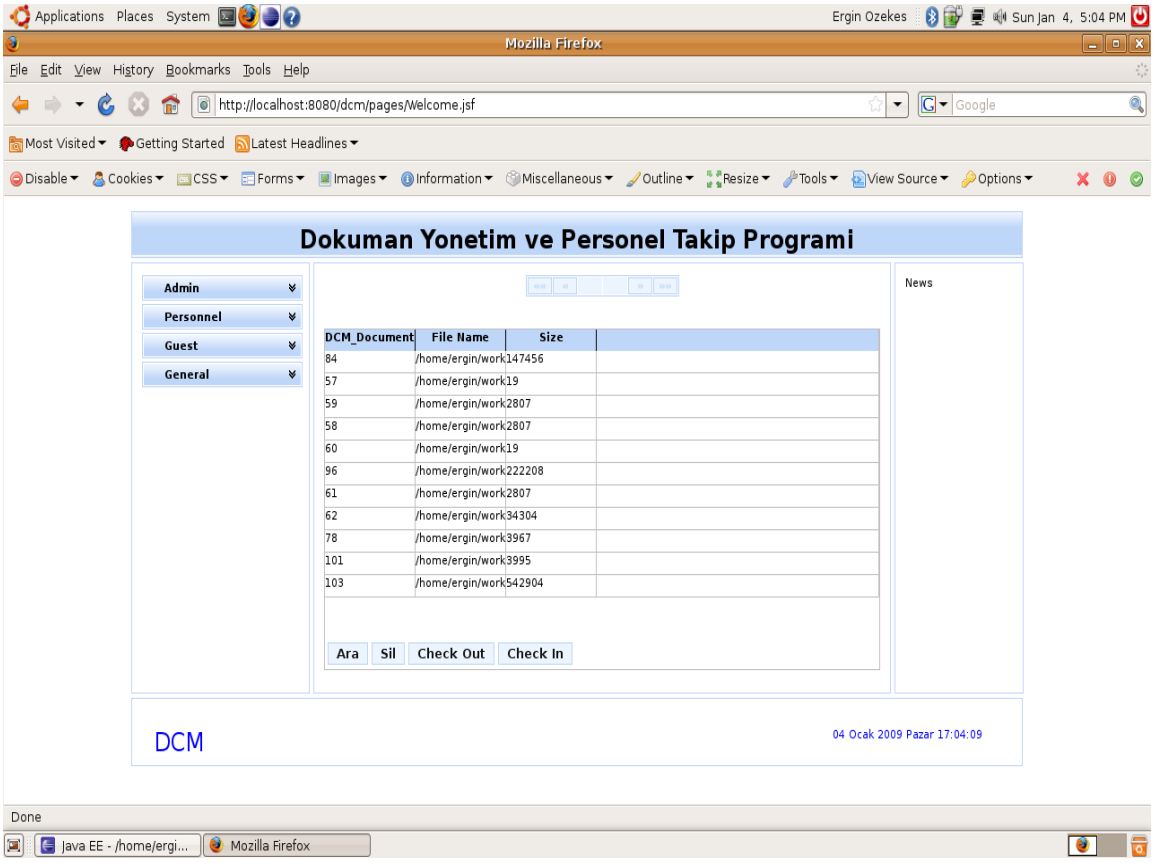
Test Vakası Açıklaması: Sistemde tutulan kayıtların listelenmesi doğrulanır.

Test Vakası Çalışma Adımları: Admin kısmında ise şekil 3.3'de görüldüğü gibi Document, Action, Department, Template, Role, Task, City, Town, User, Meetings, Grup verilerinin düzenleneceği alt menü üyeleri bulunur. Her alt menü içinde yeni kayıt ekleme ve listeleme menüleri mevcuttur. Bu listeleme menüsü tıklanıp liste ekranı getirilir. Her liste de başlık kısmındaki sütun adına tıklayarak görünen listelemenin o sütuna göre sıralanması sağlanır.

Beklenilen Davranış ve Çıktılar: Şekil 3.4'de görüldüğü gibi listeleme ekranı görülmeli ve sıralama sonucu sütun içindeki değerlerin sıralandığı görülmelidir.



Şekil 3.3 Ana menü



Şekil 3.4 Listeleme sayfası

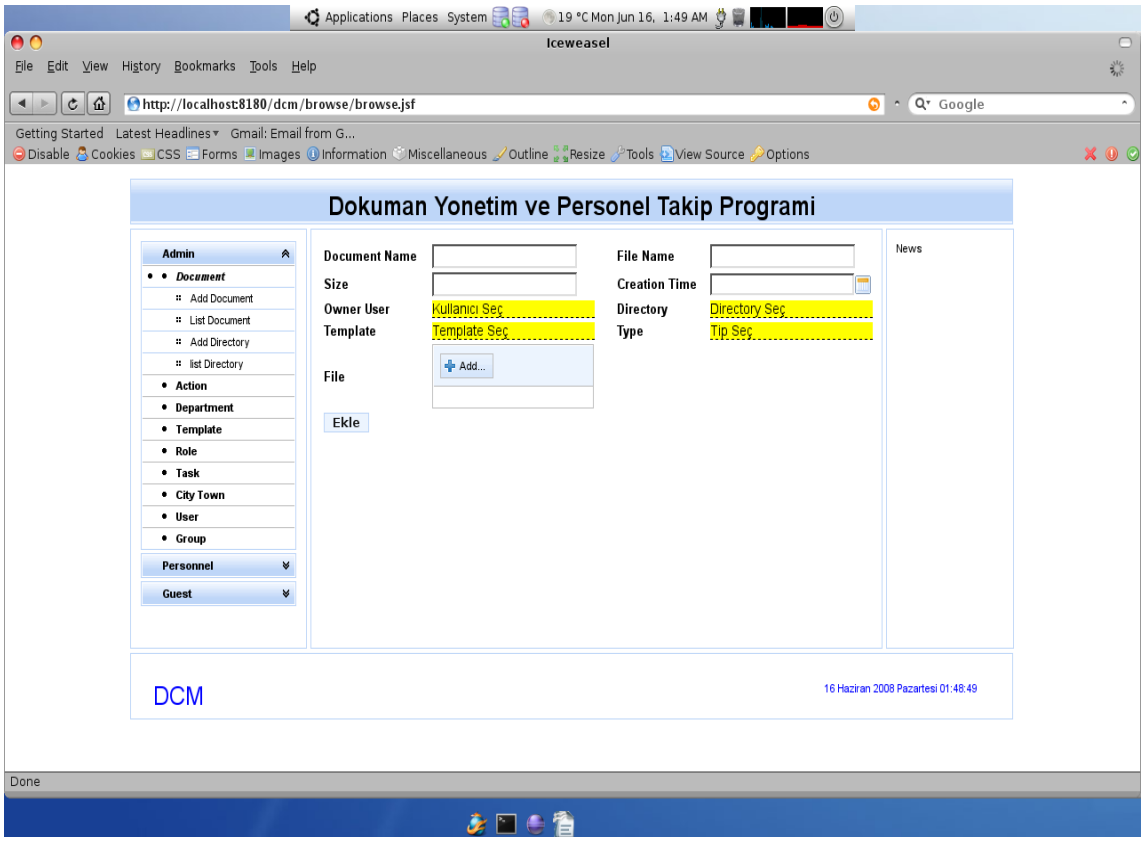
Test Vaka No: T3

Test Vakası Adı: Kayıt ekleme

Test Vakası Açıklaması: Sisteme yeni kayıt eklenişinin doğrulanması.

Test Vakası Çalışma Adımları: Her alt menü içinde yeni kayıt ekleme ve listeleme menüleri mevcuttur. Kayıt ekleme menüsü tıklandığı zaman şekil 3.5'dekine benzer form ile karşılaşılır. Gerekli değerler doldurulur. Ekle düğmesi tıklandığı zaman o kayıt ile ilgili listeleme ekranına geri dönülür.

Beklenilen Davranış ve Çıktılar: Listeleme ekranında yeni girilen kayıta ait satır görülmelidir. Şekil 3.4'dekine benzer listeleme ekranı görülmelidir.



Şekil 3.5 Kayıt Ekleme

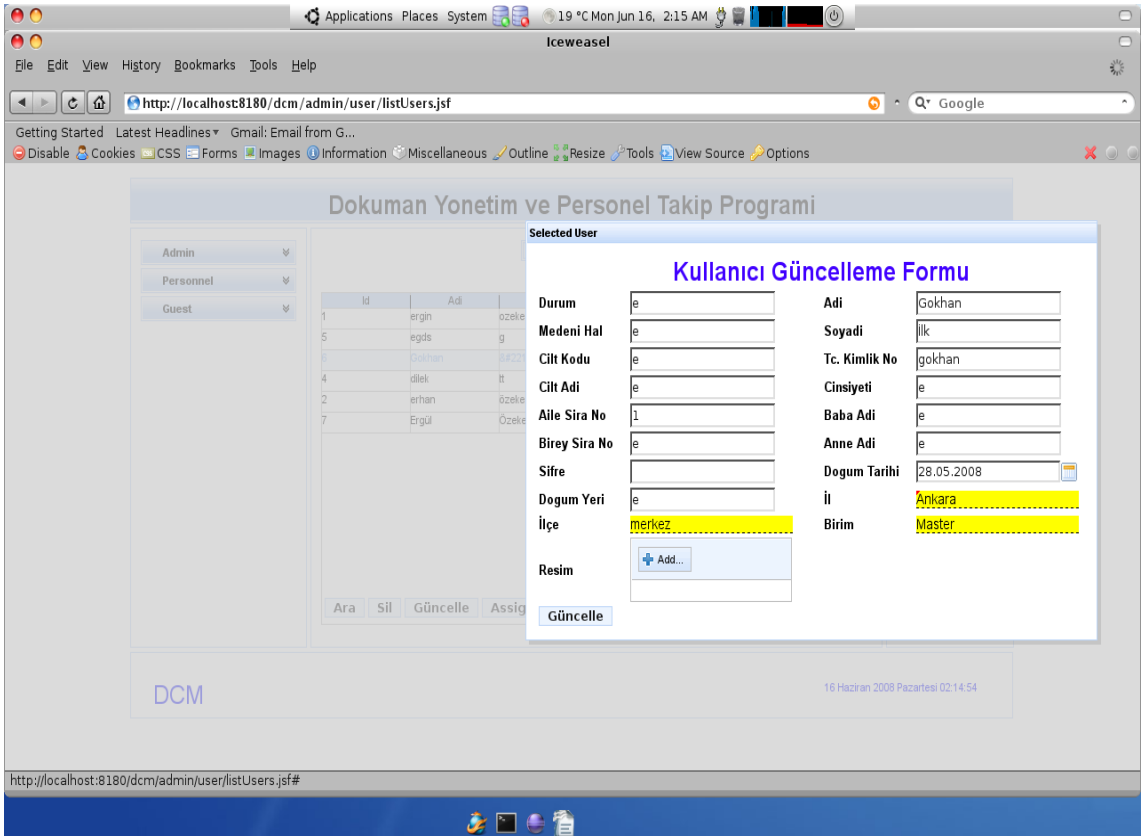
Test Vaka No: T4

Test Vakası Adı: Kayıt güncelleme

Test Vakası Açıklaması: Sistemdeki mevcut kayıtın güncellenmesinin doğrulanması.

Test Vakası Çalışma Adımları: Her listeleme formunun altında güncelle düğmesi bulunur. Kayıt güncelleme ise listeleme ekranında Shift tuşu basılı iken kayıtın bulunduğu satıra tıklanıp kayıt seçilir ve Güncelle düğmesine tıklanarak yapılır. Bu düğme tıklandığında ise modal formda güncelleme formu gözükür. Örneğin kullanıcı güncelleme şekil 3.6'da görüldüğü gibi kullanıcı güncelleme formundaki verileri güncelleyerek Güncelle düğmesi tıklanır.

Beklenilen Davranış ve Çıktılar: Modal formda iken eski değerler kendi alanlarında mevcut olmalıdır. Listeleme ekranında güncellenen kayıt görülmelidir. Şekil 3.4'dekine benzer listeleme ekranı görülmelidir.



Şekil 3.6 Güncelleme

Test Vaka No: T5

Test Vakası Adı: Kayıt silme

Test Vakası Açıklaması: Sistemdeki mevcut kayıtın silinmesinin doğrulanması.

Test Vakası Çalışma Adımları: Her listeleme formunun altında Sil düğmesi bulunur. Kayıt silme listeleme ekranında Shift tuşu basılı iken kayıtın bulunduğu satıra tıklanılarak kayıt seçilip Sil düğmesine tıklanarak yapılır.

Beklenilen Davranış ve Çıktılar: Listeleme ekranında silinen kayıt görülmemelidir.

Test Vaka No: T6

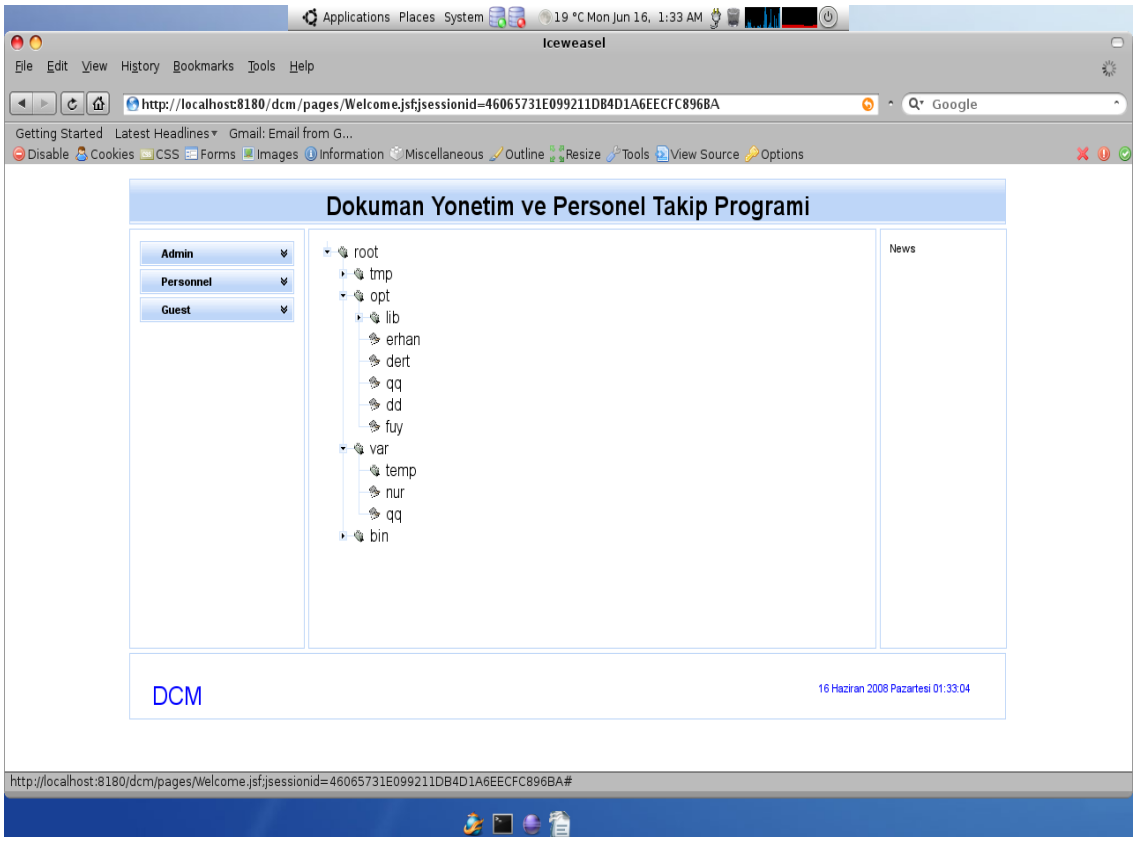
Test Vakası Adı: Tarayıcı

Test Vakası Açıklaması: Sistemdeki mevcut dokümanları içinde bulundukları dizinler içinde ağaç yapısı şeklinde listelenmesinin doğrulanması.

Test Vakası Çalışma Adımları: Sistem içindeki dosyaları ağaç yapısı şeklinde görüntülenmesi için ana menüdeki Personnel veya Guest menüleri altındaki Browse alt

menüsü tıklanır.

Beklenilen Davranış ve Çıktılar: Şekil 3.7'deki gibi bir liste görülmelidir.



Şekil 3.7 Tarayıcı

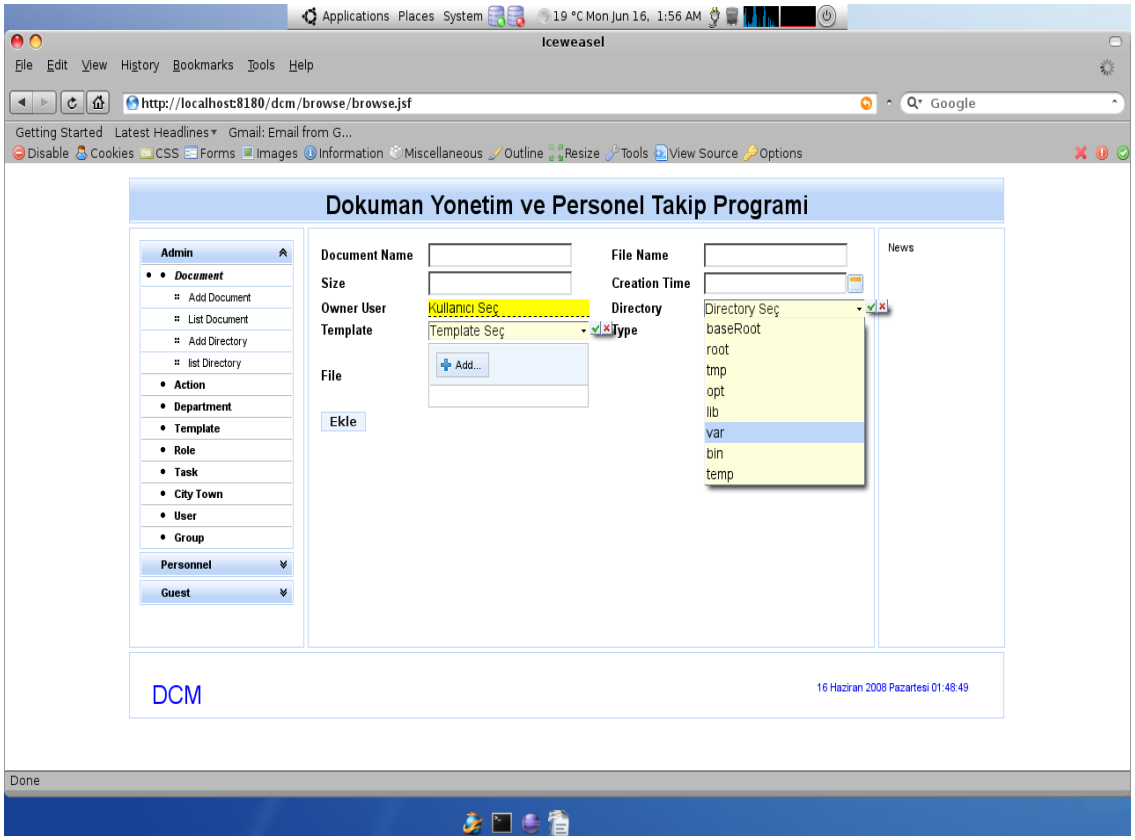
Test Vaka No: T7

Test Vakası Adı: Değerler listesi (List of values (LOV))

Test Vakası Açıklaması: Sistemdeki mevcut formlarda başka formlarda bulunan kayıtlara referans tanımlamanın doğrulanması.

Test Vakası Çalışma Adımları: Burada GDYS sisteminin genelinde kullanılmış değerler listesi için sarı renkli kombo kutuların tıklanması ile çıkan listeden istenilen seçim yapılır sonra ya Enter tuşlanarak veya kombo kutusunun yanında çıkan şekil 3.8'de görüleceği gibi yeşil onay düğmesi tıklanılır. Yeşil onay düğmesi tıklanmazsa seçim yapılmamış anlamı içerir.

Beklenilen Davranış ve Çıktılar: Yapılan seçim sarı renkli kombo kutusunun olduğu yerde görülmelidir. Yeşil onay düğmesi tıklanmadı ise seçim görülmemelidir.



Şekil 3.8 Değerler listesi

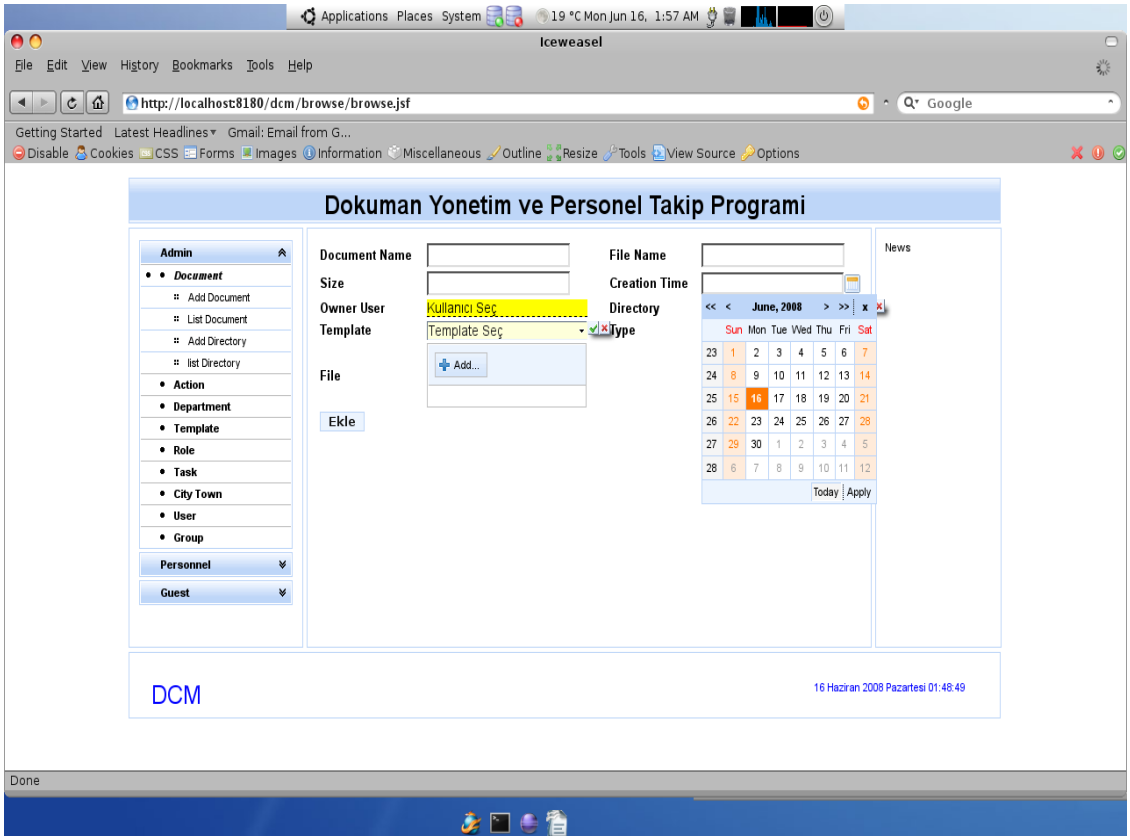
Test Vaka No: T8

Test Vakası Adı: Takvim

Test Vakası Açıklaması: Sistemdeki mevcut formlarda tarih değerinin girilmesinin doğrulanması.

Test Vakası Çalışma Adımları: Tarih seçimi için GDYS sisteminde genel olarak şekil 3.9'da görüldüğü gibi takvim bileşeni kullanılmıştır. Burada tarih belirlendikten sonra alttaki Apply düğmesi tıklanmalıdır.

Beklenilen Davranış ve Çıktılar: Apply düğmesi tıklandıktan sonra tarih değeri takvim bileşeninin bulunduğu yerde görülmelidir.



Şekil 3.9 Takvim

Test Vaka No: T9

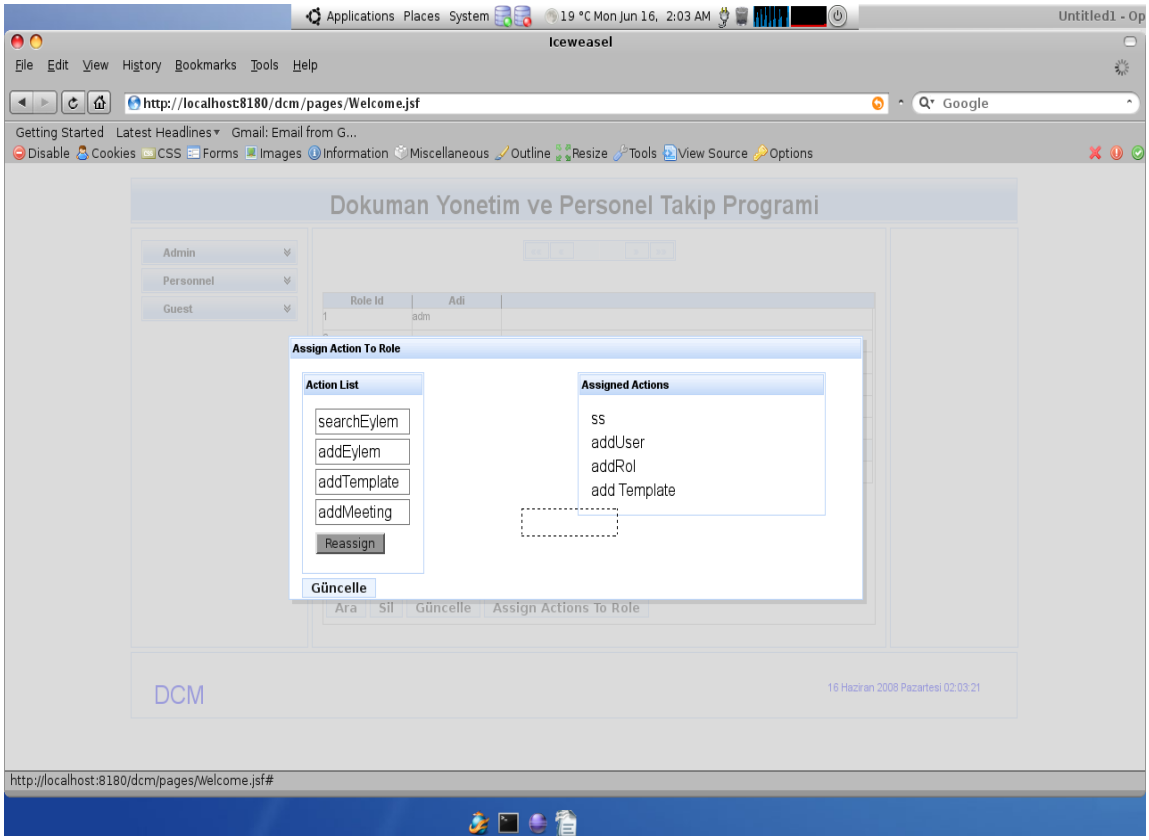
Test Vakası Adı: Role eylemler atama

Test Vakası Açıklaması: Sistemde bulunan rollere o rolün yapabileceği eylemlerin atanması.

Test Vakası Çalışma Adımları: AssignRole2User, AssignTask2User, AssignAction2Role gibi menü işlevselliği ise şekil 3.10'de görüleceği üzere listeleme ekranında bir kayıt seçilir (örneğin rol), AssignAction2Role düğmesine tıklanılır ve modal formda sürükle bırakla o role atanacak eylemlerin seçileceği form gelir. Soldaki eylem listesinden role atanacak eylemler seçilerek fare ile sağdaki listeye sürüklenir ve bırakılır. Seçilen eylem role eklenen eylemler içine girer. Güncelle düğmesi tıklanarak seçilen tüm eylemler o role atanır. Bu role ise herhangi bir kullanıcıya aynı yöntemle atanarak o kullanıcının yetkileri belirlenir.

Beklenilen Davranış ve Çıktılar: Role eklenen eylemler o role sahip kullanıcı ile sisteme giriş yapılıp eylemlerin gerçeklenebilmesi beklenir. Kişi yetkisi olmadığı halde

bir menü işlevini yapmak isterse sistem tarafından bu işlem sonuçlandırılmaz. O eylem gerçeklenmemelidir.



Şekil 3.10 Role Eylem Atama

Test Vaka No: T10

Test Vakası Adı: Günce tutma

Test Vakası Açıklaması: Sistemdeki mevcut kullanıcıların veriyi etkileyen her işleminin güncesinin tutulduğunun doğrulanması.

Test Vakası Çalışma Adımları: Bu sistemde ilk giriş yapıldığında zaman sisteme giren kişinin tüm yaptıklarının güncesinin tutulması her yaptığı işlemden sonra yapılır. Bu güncelerden istenirse geriye doğru veriler eski haline alınabilir ve incelenebilir.

Beklenilen Davranış ve Çıktılar: Kullanıcı işlemi sonrası kullanıcının işlemine ait değerler Kullanici_Eylem veritabanı tablosunda görülmelidir.

3.3 Test Vaka Sonuçları

Yukarıda listelenmiş test vakalarının sisteme uygulanması sonucu çizelge 3.1 elde edilmiştir.

Çizelge 3.1 Test vaka sonuçları

Test Vaka No	Test Vaka Adı	Beklenilen Davranış Gerçeklendi mi?	Gözlemlenen Davranış
T1	Sisteme Giriş	Evet	
T2	Kayıt Listeleme	Evet	
T3	Kayıt Ekleme	Evet	
T4	Kayıt Güncelleme	Evet	
T5	Kayıt Silme	Evet	
T6	Tarayıcı	Evet	
T7	Değerler Listesi	Evet	
T8	Takvim	Evet	
T9	Role eylemler atama	Evet	
T10	Günce Tutma	Evet	

Çizelge 3.1'de Test Vaka No, Test Vaka Adı , Beklenilen Davranış Gerçeklendi mi? ve Gözlemlenen Davranış sütunlarını içeren bilgiler Test Vaka Sonuçları olarak verilir. Gözlemlenen davranış sütunu beklenen davranış gerçekleşmedi ise test sonucu, görülen davranış ve hatalar ile doldurulur.

4. SONUÇ ve TARTIŞMA

Bulgular bölümünde verilmiş çizelge 3.1'deki test vaka sonuçları, GDYS sistemi isterlerinin sağlandığını göstererek yazılım geliştirme sürecinin en son adımı olan kabul aşamasının tamamlanıldığına işaret etmektedir.

GDYS için BMD temelli BS tasarım yönteminin kullanılması ile gereksinimler doğrultusunda istenilen boyutlara ölçeklenebilen esnek bir sistem geliştirilmiştir. GDYS sistemine ait şekil 2.12'deki bileşen diyagramında görülebileceği üzere sistem farklı işlevlere sahip yazılımlardan oluşturulmuştur. GDYS sisteminde kullanılan tüm yazılımların yazılım demeti destekleri mevcuttur. Yazılımların hepsi tek makine üstünde Linux veya Windows işletim sistemi üstünde koşturulabileceği gibi aynı yazılım ihtiyaca göre belli yapılandırılmaya sahip birden fazla makine üstünde ister yazılım ister donanım olarak birbirlerine demetlenerek de koşturulabilir. GDYS sistemi, demo veya geliştirme amaçlı tek makine üstünde tüm sistemin koşturulabileceği gibi aynı anda binlere hitap edebilecek profesyonel sistemlere de ölçeklenebilir.

BS temelli geliştirme, yazılım geliştirme aşamasında gelen isteklerin sisteme oldukça rahat bir şekilde bütünleştirilmesine izin verir. Örneğin sisteme sürüm desteği eklenilmesi istediğinde gerçeklenen ekleme süreci, tasarım aşamasındaki kullanım vakasına müdahale edip bu kısma yapılan eke ilişkin sınıfların tanımlanması, modelden kod oluşturma ile yazılıma ek modüllerin eklenmesi, bu sınıflardan veri tabanı için gerekli değişiklere ait yeni VTD ifadelerinin oluşturulması ve veri tabanına uygulanması ile yerine getirilebilir (Boggs and Boggs 2002).

Sisteme yeni giren bir kullanıcının sistem hakkında fazla bir bilgiye sahip olmadan sistemi kullanabilmesi amaçlanmış ve kullanılan işçatıları ile kullanıcı dostu ara yüzler geliştirilmiştir. Kullanıcı için hem basit hem de yetenekli olan ağ sayfaları oluşturulmuştur. Örnek sistemler kısmında tanıtılmış ticari ve açık sistemlerde JSF teknolojisi kullanılmamıştır. GDYS, DYS alanında JSF teknolojisi kullanan ilk yazılım sistemi örneklerindendir.

JGV platformunun tüm nimetlerinden faydalanan GDYS sistemi örnek sistemler kısmında bahsedilmiş sistemlerden daha güçlü kimlik denetimi ve yetkilendirme yapar. Alanında OpenLdap kullanan ilk örneklerden olan GDYS, Tomcat'nin sağladığı alan modülleri sayesinde arka planda gerçeklenen kimlik denetimi ve yetkilendirme sürecini diğer güvenlik çözümlerine kolaylıkla taşıyabilir.

Profesyonel teknikler kullanılarak geliştirilmiş GDYS sistemi ile yapılan kelime arama, yeni dosya ekleme, yeni kullanıcı ekleme, dosya düzenleme, kişi arama, sürümleme gibi özellikler akademik yayınların incelenmesi açısından rahatlık sağladığı gibi kullanılan şablonlar ve görevlerle birlikte kalitelerinin de artması sağlanmıştır.

Sonuç olarak bireylerin tümleşik bir sistem altında grup olarak çalışmaları sağlanmıştır. Grup çalışması sayesinde birbirleri ile etkileşerek bilgi paylaşımı sağlanabilmiştir. Güvenli ve aynı zamanda bilirkişi denetiminin sağladığı tecrübelerden de faydalanarak bilim insanlarının daha hızlı ve etkin akademik yayın üretmelerine imkân sağlayacak bir yazılım sistemi sunulmuştur.

KAYNAKLAR

Anonymous. 2007. Red Hat. RichFaces Developer Guide.

Anonymous. 2008. Alfresco. Web Sitesi: http://www.alfresco.com, Erişim Tarihi: 30.8.2008

Anonymous. 2008. CASE Studio 2. Web Sitesi: http://www.casestudio.com/enu/default.aspx, Erişim Tarihi: 12.10.2008

Anonymous. 2008. DBDesigner4. Web Sitesi: http://fabforce.net/dbdesigner4, Erişim Tarihi: 22.08.2008

Anonymous. 2008. Design for Databases V3. Web Sitesi: http://www.datanamic.com/dezign, Erişim Tarihi: 17.08.2008

Anonymous. 2008. Documentum. Web Sitesi: http://www.documentum.com, Erişim Tarihi: 19.10.2008

Anonymous. 2008. ER/Studio. Web Sitesi: http://www.embarcadero.com/products/erstudio/index.html, Erişim Tarihi: 10.09.2008

Anonymous. 2008. Happy Fish Database Designer. Web Sitesi: http://www.embarcadero.com/products/erstudio/index.html, Erişim Tarihi: 29.10.2008

Anonymous. 2008. KnowledgeTree. Web Sitesi: http://www.knowledgetree.com, Erişim Tarihi: 18.09.2008

Anonymous. 2008. Main Pyrus. Web Sitesi: http://www.mainpyrus.org, Erişim Tarihi: 01.09.2008

Anonymous. 2008. OpenKM. Web Sitesi: http://www.openkm.com, Erişim Tarihi: 22.10.2008

Anonymous. 2008. Oracle Designer 2000. Web Sitesi: http://www.Oracle.com/technology/products/designer/index.html, Erişim Tarihi: 02.10.2008

Anonymous. 2008. QDesigner. Web Sitesi: http://www.quest.com/QDesigner, Erişim Tarihi: 19.09.2008

Armstrong, E., Ball, J. and Bodoff, S. 2003.The Java Web Services Tutorial. California, Sun Microsystems

Bakore, A., Bhattacharjee, D. and Bhattacharya, S. 2003. Professional Apache Tomcat. Indianapolis, Wiley.

Bergsten, H. 2002. JavaServer Pages, 2nd Edition. O'Reilly.

Boggs, W. and Boggs, M. 2002. Mastering UML with Rational Rose 2002. Alameda, Sybex

Booch, G., Rumbaugh, J. and Jacobson, I. 1998. The Unified Modeling Language User Guide. New York, Addison-Wesley.

Booch, G., Rumbaugh, J. and Jacobson, I. 1999. The Unified Modeling Language Referans Manual. New York, Addison-Wesley.

Chappell, D. and Jewell, T. 2002. Java Web Services , First Edition . O'Reilly.

Connolly, T., Begg, C. and Strachan, A. 1999. Data Systems , Second Edition. New York, Addison Wesley.

Douglas, K. and Douglas, S. 2005. PosgreSQL: The comprehensive guide to building, programming, and administering PostgreSQL databases, Second Edition. Sams Publishing

Dudney, B., Lehr, J. and Willis, B. 2004. Mastering JavaServer Faces. Indianapolis, Wiley.

Geary, D. and Hortsmann, C. 2007. Core Java Server Faces, Second Edition. New York, Prentice Hall.

Grand, M. 2002. Java Enterprise Design Patterns , vol.3. New York: Wiley.

Howes, T., Kille, S. and Yeong, W. 1993. X.500 Lightweight Directory Access Protocol. RFC 1487

Khawar, Z. A. and Umrysh, C. E. 2002. Developing Enterprise Java applications with J2EE and UML. New York, Addison-Wesley.

Kruchten, P. 2000. The Rational Unified Process An Introduction, Second Edition. New York, Addison Wesley.

McGovern, J., Adatia, R. and Fain, Y. 2003. Java 2 Enterprise Edition 1.4 Bible. Indianapolis, Wiley.

Sumathi, S. and Esakkirajan, S. 2007. Fundamentals of Relational Database Management Systems. New York, Springer

ÖZGEÇMİŞ

Adı Soyadı : Ergin ÖZEKES

Doğum Yeri : Sinop/Merkez

Doğum Tarihi : 18/06/1974

Medeni Hali : Evli

Yabancı Dili : İngilizce

Eğitim Durumu (Kurum ve Yıl)

Lise : Sinop Atatürk Lisesi (1991)

Lisans : Yıldız Teknik Üniversitesi Mühendislik Fakültesi Elektronik ve Haberleşme Mühendisliği Bölümü (1995)

Yüksek Lisans : Ankara Üniversitesi Fen Bilimleri Enstitüsü Elektronik Mühendisliği Anabilim Dalı (Şubat 2005 – Ocak 2009)

Çalıştığı Kurum/Kurumlar ve Yıl :

Ankara Üniversitesi, Bilgi İşlem Dairesi Başkanlığı (2001 – 2007)

Manisa Vestel Electronic (2007 – --)