



T.C.

GEBZE YÜKSEK TEKNOLOJİ ENSTİTÜSÜ
MÜHENDİSLİK VE FEN BİLİMLERİ ENSTİTÜSÜ

ŞİPŞAK: E-POSTA İLE PAZARLAMA
YAZILIMI

171 321

Utkan Tolga EVREN
YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ
ANABİLİM DALI

TE. YÜKSEKÖĞRETİM KURULU
DOKÜMANTASYON MERKEZİ

TEZ DANIŞMANI
Yrd.Doç.Dr. İbrahim SOĞUKPINAR

GEBZE
2002

171321

Utkan Tolga EVREN 'in tez çalışması G.Y.T.E. Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulunun 01.01.2002 tarih ve 2002/2.....sayılı kararıyla oluşturulan jüri tarafından Bilgisayar Mühendisliği Ana Bilim Dalında YÜKSEK LİSANS Tezi olarak kabul edilmiştir.



ÜYE : Yrd.Doç.Dr. İbrahim SOĞUKPINAR ,GYTE

(Tez Danışmanı)

ÜYE : Yrd.Doç.Dr. Mehmet GÖKTÜRK ,GYTE

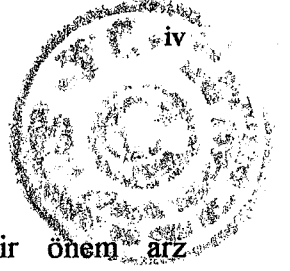
ÜYE : Yrd.Doç.Dr. Adnan CEYLAN ,GYTE



G.Y.T.E. Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 01.03.2002.....tarih ve 2002/9..... sayılı kararı.

İmza/Mühür

ÖZET



Günümüzün modern dünyasında Internet, çok büyük bir önem arz etmektedir. Modern dünya bir çok alanda Internet kullanmakta ve internetin her türlü alandaki etkinliği giderek artmaktadır. Internet insan yaşamına getirdiği ; bilgiye kolayca ve hızla ulaşma olanağının yanı sıra , bu bilgiyi başkalarıyla paylaşabilme imkanını sağlamış, yine bir çok iş alanına çok büyük imkanlar sunmuştur. E-posta , icq veya sohbet gibi Internet teknolojileri ve uygulamaları , insanların birbirleriyle iletişim kurmalarında telefonda bir adım öteye geçilmesini sağlamıştır. İş dünyasının interneti kendi amaçları için kullanması fikrinin hayata geçirilmesiyle , insanlar , firmalar arasındaki mesafeler önemini yitirmiş , insanların ihtiyaçlarını karşılamak bilgisayar tuşlarıyla oynamak kadar sadeleştirilebilmiştir. Günümüzde, modern toplumlarda Internet üzerinden alışveriş yapmak , fatura ödemek , bankacılık işlemi gerçekleştirmek giderek yaygınlaşmakta olup internetin toplum hayatıyla ilgili bir çok değişik alanda kullanılmasıyla ilgili araştırmalar yapılmaktadır.

Internet teknolojileri alanında yapılan araştırmalar ve ilerlemeler sayesinde ,Internetin değişik iş alanlarında kullanılması mümkün olmuştur. Her ne kadar kullanıcılar için internete bağlanmak kolay olsa da , kullanıcılara sağlanan servislerin geliştirilmesi karmaşık yapıların ve yazılım mimarilerinin oluşturulmasını gerektirmektedir.

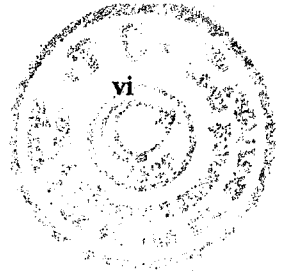
Bu çalışmada Internet üzerinde iş yapma türlerinden birisi olan e-posta ile pazarlama konusu ele alınmış ve bu konuyla ilgili olarak bir yazılım çözümü geliştirilmiştir. Bu çalışma , e-posta , icq , fax veya daha ileri bir iletişim ortamını, bilgi aktarmak , uyarı göndermek , pazarlama yapmak vb. amaçlarıyla nasıl kullanılabileceğini tespit etmeyi ve bunu gerçekleştirmeyi amaçlamıştır. Bu yazılım çözümü , günümüz dünyasında kullanılabilecek bir B2B e-ticaret platformu yaratmayı amaçlayarak ,e-ticaret ve e-pazarlama konularını birleştiren yeni bir çözüm getirmektedir.



SUMMARY

The Internet has a major role in modern world. Various uses of the Internet , makes it the most important communication and business medium. The Internet made finding & sharing information , very fast an simple. E-mail,icq or chat technologies are some examples which replace the telephones. On the other hand , the Internet has made a great change on various business. The distance between customer , firms became unimportant. As a result, having a computer and an internet account is enough to achive lots of things related to human life , such as shopping , committing bank tarnsactions , doing ticket reservations etc.

This work is about , e- mail marketing problem which is a kind of e-business. The problem is tried to solve by developing a software system. It aims to present a solution which uses today's most popular communication mediums such as e-mail,sms,fax,icq in e-mail marketing in a single platform , combining both e-mail marketing an e-commerce subjects in one place.



TEŞEKKÜR

Bu tezin hazırlanmasında ; bana gösterdiği tüm yardımlarından dolayı tez danışmanım G.Y.T.E Bilgisayar Mühendisliği Bölüm Başkanı Yrd.Doç.Dr.İbrahim Soğukpınar'a çok teşekkür ederim.



İÇİNDEKİLER DİZİNİ



	Sayfa
ÖZET	IV
SUMMARY	V
TEŞEKKÜR	VI
İÇİNDEKİLER DİZİNİ	VII
SİMGELER VE KISALTMALAR DİZİNİ	IX
ŞEKİLLER DİZİNİ	X
1. GİRİŞ	1
2. E-TİCARET ve E-POSTA İLE PAZARLAMA	7
2.1. Tanım	7
2.1.1. E-İş	7
2.1.2. E-Ticaret	7
2.1.3. Pazarlama	8
2.2. E-Posta İle Pazarlama	9
2.2.1. E-Posta İle Pazarlamanın Kuralları	10
2.2.2. Geleneksel Pazarlama İle E-Posta İle Pazarlama Arasındaki Farklar	11
2.2.3. Müşteri İlgisini Anlayabilme, E-Posta İçeriği	11
3. E-POSTA İLE PAZARLAMA ÇÖZÜMÜ	12
3.1. E-Posta İle Pazarlama Yazılımı Nedir?	12
3.2. E-Posta İle Pazarlama Yazılımının Mimarisi	12
3.3. E-Posta İle Pazarlama Yazılımının Sistem Analizi	13
3.4. E-Posta İle Pazarlama Yazılımının Teorik Yapısı ve Algoritmalar	15
3.5. E-Posta İle Pazarlama Yazılımının Özellikleri	25
3.5.1. Değişik İş Alanlarına Adaptasyon	26
3.5.2. Genişletilebilirlik	26
3.5.3. Dinamik Kategori İşlemleri	26
3.5.4. Dinamik E-posta İşlemleri	26
3.5.5. SMS Gönderebilme	27
3.5.6 - 3.5.7. FAX , ICQ Gönderebilme	27

3.5.8. Özel Durumlarla İlgili E-Posta Yaratabilme	27
3.5.9. Varolan Bilginin Sisteme Eklenmesi	27
3.5.10. Çoklu Çalışabilme	27
3.5.11. Sihirbazlarla Kolay Kullanım	28
3.5.12. Kullanıcı Hareketlerini İzleyebilme	28
3.5.13. Kullanıcılara Sistemle İlgili Seçenekler Sunabilme	28
4. UYGULAMALAR VE SONUÇLAR	29
4.1. E-Posta İle Pazarlama Yazılımının Kullanıcı Ara Yüzü	29
4.1.1. Kullanıcı Ana Menüsü Arayüzü	29
4.1.2. Posta Sihirbazı Bileşeni	30
4.1.3. Görev Zamanlayıcı Bileşeni	34
4.1.4. Profiller Bileşeni	34
4.1.5. Yayınlar Bileşeni	35
4.1.6. Kampanyalar Bileşeni	36
4.1.7. Raporlar Bileşeni	37
4.1.8. Konfigürasyon Bileşeni	38
4.1.8.1. Alıcı Alanları Bileşeni	39
4.1.8.2. Kategori Bileşeni	40
4.1.8.3. Dinamik Alanlar Bileşeni	40
5. SONUÇLAR VE ÖNERİLER	44
5.1. Sonuçlar	44
5.2. Öneriler	44
KAYNAKLAR	45
ÖZGEÇMİŞ	46
EKLER	

SİMGELER ve KISALTMALAR DİZİNİ

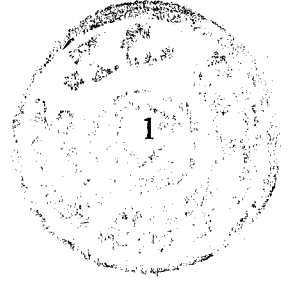
<u>Simgeler</u>	<u>Acıklama</u>
MRQ	: Posta İstek Kuyruğu. Gönderilerin işleme konmadan önce biriktirildikleri lifo mantığıyla çalışan kuyruk yapısı.
MP	; Posta prosesörü. Gönderileri işleme koyarak iletilmelerini sağlayan birim.
JC	: İş toplayıcı. Gönderileri belirli aralıklarla toplayarak MRQ da biriktiren birim.



ŞEKİLLER DİZİNİ



Şekil	Sayfa
3.1. Kullanıcı İşlemi Diyagramı	13
3.2. Sistem Sıra Diyagramı	14
3.3. Kavramsal Model	14
3.4. İleti Gönderme UML Diyagramı	15
3.5. Yazılımın Genel Yapısı	16
3.6. Değişik Dillere Destek Verebilme	17
3.7. İş Kuralları Çalışma Prensipleri	18
3.8. İzleyici Yapısı	20
3.9. Hızlı Posta Gönderme Sistemi	21
3.10. Sistem İzleme/Yönetme	22
3.11. Tekrarlı İş Yaratma Yapısı	24
4.1. Giriş Ekranı	29
4.2. Kullanıcı Ana Menüsü	30
4.3. İleti Yaratma Bileşeni	31
4.4. İletinin Hedeflerini Belirleme	32
4.5. İletinin Gönderme Zamanını Belirleme	33
4.6. SMS İletisi Hazırlama	33
4.7. Görev Zamanlayıcı	34
4.8. Profil Bileşeni Ara Yüzü	35
4.9. Yayınlar Bileşeni	36
4.10. Kampanyalar	37
4.11. Raporlar	38
4.12. Kullanıcı Alanları Arayüzü	39
4.13. Kategori Bileşeni	40
4.14. Dinamik Alanlar	41



1.GİRİŞ

Günümüzde , bilgisayarların yaygın kullanımıyla beraber , iş dünyasının bir çok alanında bir değişim meydana gelmiştir. E-iş , hemen her alanda kendisini göstermekte ve giderek insan hayatında etkin bir araç olmaktadır. Şirketlerin pazarlama yöntemlerinin bilgisayar ortamına kaydırılması da ayrı bir e-iş ve e-ticaret formudur.

E-ticaret , geleneksel ticaret yöntemlerine göre sağladığı avantajlar sayesinde , günümüzde bir çok alanda tercih edilen öncelikli iş yapma metodu olma durumundadır.

Genel olarak iş yapmak üretilen bir malı satmak demektir. Pazarlama ise malı satmak için müşterilerle iletişime geçmek ve onlara ürünü satmaya çalışmaktır. Geleneksel olarak bir şirket müşterilerine aşağıdaki yöntemlerle ulaşır[1]:

- Yüz-yüze görüşme ile
- Posta ile
- Telefon ile

Satış kanalı olarak adlandırılan bu değişik yöntemler , müşterilere ihtiyaçları olan ürünleri veya servisleri sunmayı amaçlarlar.[10]. Internetin yaygınlaşması , ve iş hayatında kullanılmaya başlanması , bir çok iş alanının geleneksel yöntemlerini değiştirmiş ve e-iş denilen terim ortaya çıkmıştır. Bir araştırmaya göre , 1998 yılında internet ekonomisi 300 milyar dolarlık bir gelir yaratmıştır.[11]. Ayrıca elektronik ortamlarda alışveriş geleneksel yöntemlere göre çok daha popüler hale gelmiştir.[8,9] Internet üzerinde iş yapmanın avantajları şöyle sıralanabilir[1]:

- Çok daha büyük bir kitleye ulaşabilme imkanı
- Müşteriyle daha yakın ilişki kurabilme
- Masraflarda azalma
- Zamandan kazanç

Internet , içerisinde milyonlarca kullanıcı bulunan büyük bir topluluktur.Bu büyük topluluk , Internet üzerinde iş yapmayı tasarlayan bir iş alanı için çok cazip bir müşteri potansiyeli anlamına gelmektedir.

Bu kadar büyük bir hacme , geleneksel satış kanallarıyla ulaşabilmek mümkün değildir.Modern dünyada Internet , iş alanının yeni kanalı haline gelmiştir. Internetle beraber değişen iş yapma yöntemi, beraberinde diğer alanlarda da bazı değişiklikleri yaratmıştır. Internet üzerinde pazarlama da Internet dünyasında iş yapmaktan doğan bir ihtiyaçtır.Denilebilir ki, pazarlamacılar interneti kullanmalı mı yerine pazarlamacılar interneti nasıl kullanmalıdır sorusunun cevabı daha önemlidir[12].

Internet teknolojilerini pazarlama aracı olarak kullanmak konusunda bir çok araştırma yapılmış ve bu konuda da birçok yazılım oluşturulmuştur. Tasarlanan yazılım çözümlerinin bir çoğu yurtdışı kaynaklı olup , yurtiçinde ki benzer çalışmaların sayısı oldukça sınırlıdır. Tüm çalışmaların ortak özellikleri:

- 1.B2B ticarete yönelik olmaları.
- 2.Hedef kitle olarak e-posta hesabı olanları seçmeleri
- 3.HTML tipinde verileri göndermeleri

olarak özetlenebilir.

Yurtiçerisinde bu konu üzerine yapılan çalışmalara örnek olarak messagemarketer gösterilebilir.(www.messagemarketer.com) MessageMarketer , geliştirdiği platformla e-posta ile pazarlama konusunda iddialı bir konumdadır.

Bu konu üzerinde geliştirilen messagemarketer ve benzeri yazılımların özellikleri aşağıda açıklanmıştır:

1. Hedef kitleyi bir veritabanı altında toplayabilme ve buradan yönetebilme.

Hedef kitle , bir çok şirketin bir şekilde elinde bulunan müşteri listesi olarak düşünülebilir.Bu bilgiler , çoğu zaman bir başka yazılımın bünyesinde bir veritabanı aracılığıyla tutulduğu gibi, şirketlerin müşteri hacmine göre bu tür bilgiler , veri tabanı



dışında dosyalarda ,(mesela Microsoft outlook formatında) tutulabilir. Genel olarak , geliştirilen yazılım çözümleri , bu tür dataları kendi içerilerindeki veritabanlarına entegre etmektedirler.

Bunlarla beraber , yazılım çözümleri , bu tür bir veritabanı olmayan firmalara , e-posta ile pazarlamada kullanılabilecek şekilde bir veritabanı yaratabilme imkanında tanır.Böylece , yazılım çözümü , e-posta ile pazarlama çözümü olmasının dışında , firmanın iş yapma düzenini de , pazarlama yöntemlerine hazır hale getirir.

2. Hedef kitleyi değişik veri tabanlarından toplayabilme

Hedef kitlenin zaten hazır olduğu durumlarda , yazılım çözümleri istenirse yeni bir veri tabanı yaratmadan var olan bilgileri kullanmayı sağlamaktadır.Bu yöntemin en büyük avantajı , hedef kitle ile ilgili bilgi değişikliklerinin tek bir kaynağa yönltilmesi durumundan dolayı , data aktarma gibi güncelleme işlerine gerek olmamasıdır.

3. Hedef kitleye ekleme yapabilme

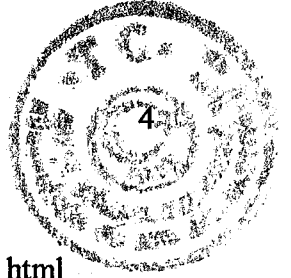
Hemen hemen bütün yazılım çözümleri hedef kitleye daha sonradan eklemeler yapabilmeye olanak tanımaktadır. Bu istenirse yazılım çözümü dışında , yazılım çözümünden habersiz bir şekilde direkt olarak veri tabanına yapılan eklemeler olduğu gibi , yazılım çözümünün sağladığı araçlar vasıtasıyla da yapılabilir.

4. Hedef kitleden geri besleme bilgisi alabilme

Gönderilen bir iletinin devamını takip etmek ve gönderiyi alan alıcının sistemden çıkmak istemesi durumunda , bir daha bu kişiye gönderi yollamamak e-posta ile pazarlama yazılımlarının ortak özelliğidir.

5. Bire bir e-posta ile pazarlama projeleri planlayabilme ve yürütebilme

Var olan yazılım çözümleri , hedef kitleyi ilgi alanlarına göre gruplama imkanı tanımakta ve bu gruplara yönelik ileti hazırlayabilmektedir.İlgi alanları , varolan yazılım çözümlerinde , sistem içerisine gömülü bir şekilde bulunmakta ve yeni bir ilgi alanı yaratılmasını mümkün kılmamaktadır. Bu da , e-posta ile pazarlama yazılımının değişik iş alanlarına entegrasyonunu zorlaştırmaktadır.



6. Metin içeriğine görsel unsurlar ekleyebilme

Gönderilerin içeriğinin oluşturulması , var olan yazılım çözümlerinde html metinlerinin ileti hazırlanırken yapılması , gönderiye eklenecek görsel unsurların kullanıcıdan istenmesi ile yapılmaktadır. Yazılım çözümleri sadece gönderiyi hazırlamaya imkan tanımakta , bunun dışında alternatifler sunmamaktadır.

7. Gönderiler ile ilgili raporlar alabilme

Hedef kitlenin gönderilerin ne kadarını aldığı , ne kadarını alamadığı , gönderiler içerisindeki linklerin ziyaret edilip edilmediği rapor olarak hazırlanabilmektedir.

8. Gönderilerin ücretlendirilmesi

Yazılım çözümlerinin içerisinde , gönderilerin ücretlendirilmesi ile ilgili olarak online ödeme sistemleri mevcut değildir. Bu yüzden , ödeme işlemleri firmalar arasında geleneksel yöntemlerle yapılmaktadır.

Bu çalışmada geliştirilen yazılım , varolan yazılım çözümlerinin eksik görülen yönlerini tamamlamayı amaçlamıştır. Yazılımın getirdiği yenilikler aşağıda açıklanmıştır.

- Yazılım tamamen Internet tabanlı bir e-commerce uygulaması şeklindedir.

Geliştirilen yazılım , bütün işlemleri bir tarayıcı üzerinden yapabilmektedir. Bu yüzden , yazılımı kullanmak isteyen kullanıcıların herhangi bir yükleme işlemi yapmalarına gerek yoktur. Ayrıca , kullanıcılar , yazılımdaki işlemlerin bedellerini yine Internet üzerinden kredi kartı ile ödeyebilirler. Bu istenirse:

1. Her ileti yapıldıktan sonra ,
2. Ay sonunda toplam rakamı bir kerede kredi kartından tahsil ederek , olmak üzere 2 şekilde yapılabilir.

1. Hedef kitleye ekleme yapabilme

Hedef kitleye yeni ekler diğer yazılım çözümlerinin yaptığı gibi yapılabilmeyle beraber , yine tarayıcı üzerinden toplu data yükleme imkanı mevcuttur. Bu durumda ,



text formatındaki veya microsoft cvs formatındaki dosyalardan , data girişi datanın doğruluğuda test edilerek yapılabilir.

Farklı data kaynaklarından toplanan hedef kitle bilgilerinin ayrıca filtrelenebilir de mümkündür.Böylece , değişik data alanlarının bir tek veri kolonunda toplanması mümkün kılınmıştır.

2. İlgi alanlarını dinamik olarak ekleyebilme / değiştirebilme veya silebilme

Kullanıcıların ilgi alanı denilen kategorileri eklemeleri çıkarmaları mümkündür.Böylece diğer yazılımların önemli bir eksiği olan dinamik kategori ekleme sorunu çözülmüştür.İstenirse yeni bir ana kategori yaratılıp , bunun altına alt kategoriler konularak değişik iş alanlarına entegrasyon oldukça kolay hale getirilebilir.

3. Değişik kaynaklardan metin içeriği hazırlayabilme

Var olan çözümlerin tek bir kaynaktan içerik hazırlama limitine karşın , bir çok değişik ortamdan içerik hazırlanabilir:

- İçeriği diğer çözümler gibi gönderi esnasında yaratmak

Kullanıcı , tarayıcı üzerinde göndermek istediği iletiyi azırlar.Metinle beraber görsel unsurlarıda eklemek mümkündür.Ayrıca metni ayrı , görsel unsurları ayrı olarak bir araya getirmek yerine, yine tarayıcıda çalışan bir html editor yardımıyla içeriğin hazırlanıp , gönderilmesi mümkündür.

- Daha önceden hazırlanmış bir e-postayı ileti olarak göndermek

Yazılım platformu dışında , başka bir kaynaktan hazırlanmış bir e-postayı , ileti olarak göndermek mümkündür.Böylece , yazılım kendi dışındaki kaynaklarında kullanılmasına olanak tanır.

- Daha önceden hazırlanmış bir html dökümanını ileti olarak göndermek

Benzer şekilde , başka kaynaklarca hazırlanmış HTML dökümanlarını göndermek mümkündür.Diğer çözümlerden farklı olarak , her döküman analiz edilmekte ve içerisindeki görsel unsurlar anlaşılabilir bunlarda kullanıcıdan istenmekte böylece kullanıcının yapacağı hatalar engellenmektedir.

4. Metin içeriklerini analiz edebilme

Hazırlanan gönderilerin içerisine , daha sonradan değiştirilerek her hedef müşteri için özelleştirilebilen nesneler eklenebilmektedir.Böylece , gönderiler her alıcı için , o alıcıya ait bilgilerle iletilebilirler.

5.e-posta dışında diğer ortamlara da ileti gönderebilme

Geliştirilen yazılım , e-posta dışında sms,fax ve icq mesajları gönderebilmektedir.Ayrıca ileride kullanılması mümkün diğer ortamları da destekleyecek esnekliğe sahiptir.

6. Yazılımı, diğerlerinden ayıuran en önemli özelliği , bir platform sumasıdır.

Buna göre örneğin herhangi bir ileti gönderilirken , (messagemarketer isimli ürünün yaptığı gibi) gönderiyi danışmanlık hizmeti olarak yapmak yerinesunulan platformdan , istenildiği gibi yapmak mümkündür.

2. E-TİCARET ve E-POSTA İLE PAZARLAMA

2.1. Tanım

2.1.1. E-İş

Günümüzde Internet ; artık sadece bir bilimsel bilgisayar ağı olma formunu kaybetmiş , bunun yerine yeni jenerasyon iş alanlarının yapılmasına olanak sağlayan bir platform haline gelmiştir.Internet ile , bilgi almak ve satmak, kitap , bilgisayar vb. ürünlerin alım satımını yapmak veya konser/tatil gibi aktiviteler için bilet almak, rezervasyon yaptırmak ,veya bankacılık işlemlerini gerçekleştirmek gibi insanların gündelik hayatta ihtiyaç duyduğu bir çok şeyi yapmak mümkündür. Bütün bunlarla beraber , Internet insan hayatıyla hemen her alanda entegre olmaktadır. Yakın gelecekte, evlerde kullanılan elektronik eşyaların internete bağlanarak , mesela bir buzdolabının yakındaki markete sütün bittiğini anlayarak süt siparişi vermesi ve süt alımını tamamlaması şaşırtıcı veya enteresan bir olay olarak değil de normal bir işlem zinciri olarak görülecektir.

E-iş , Internet ortamında Internet kurallarıyla iş yapma sistemidir.[1]

2.1.2. E-Ticaret

E-ticaret ; sanıldığı gibi aksine e-iş kavramının kendisi değil , onun tıpkı diğerleri gibi bir alt koludur.E-ticaret , geleneksel ticaret araçlarından yararlanarak , Internet üzerinde ticaret yapma yöntemidir.[1] E-ticaretin kapsamında , doğal olarak bir değiş tokuş söz konusudur. Günümüzde , günlük hayatta kullanılan para teriminin yerini alan kredi kartı,çek , sayısal para gibi enstrümanlar e-ticaretin temel taşlarıdır. Teknolojinin gelişimiyle beraber kullanılmaya başlanan bir çok ödeme sistemi içerisinde , kredi kartları ile ödeme e-ticarete en çok kullanılan ödeme metodudur. Buna göre , Internet üzerinden yapılan bir ticari işlemde , alıcı satıcıya sahip olduğu kredi kartı üzerinden satın aldığı servisin bedelini öder.

2.1.3. Pazarlama

AMABD(the American Marketing Association Board of Directors) un tanımına göre ,pazarlama:

Fikirlerin,ürünlerin ve servislerin , bireysel ve organize amaçların gerçekleştirilmesi amacıyla fiyatlandırma,tanıtma ve dağıtma işlemlerinin planlanması ve yapılmasıdır.

Pazarlama , müşterilere veya potansiyel müşterilere ulaşarak onlara şirket,marka ,ürün ve servisler hakkında bilgi vermek için yapılan aktivitedir. Bu ayrıca müşterilerin veya potansiyel müşterilerin ihtiyaçlarını daha iyi anlamak ve ona göre geleceğe yönelik kararlar alabilmeyi de kapsar[7].

Bilgi çağı öncesi toplumlarda pazarlama , çoğu zaman müşterilerle yüz-yüze teması gerektirir. Bu tip aktiviteler belirli bir yatırımı (insan gücü, reklam harcamaları vb) zorunlu kılar. Bilgi toplumundaysa her şey bilgi çağı öncesi toplumlara göre çok hızlıdır[1]. İnternette pazarlama hem daha hızlı sonuç veren, hem daha ucuz ve hem de daha doğru sonuçlar verebilme özelliklerine sahiptir. Genel olarak İnternet üzerinde bir iş alanı kendisini duyurabilmek için aşağıdaki yöntemleri kullanır:

- Banner gibi İnternet reklamları
- Arama motorları
- Tartışma grupları

Bu yaklaşımdaki problem , iş alanının ulaşabileceği müşteri potansiyelinin sadece müşterilere bağlı olmasıdır. Hatta , bir müşterinin İnternet sayfalarında dolaşırken bir iş alanıyla ilgili reklamı görmesi şans faktörüne çok bağlıdır.

2.2. E-Posta İle Pazarlama

E-posta insanlara ulaşabilmek için kullanılan en iyi yöntemdir . [2]

Her gün, milyonlarca Internet kullanıcısı milyonlarca e-posta okumakta ,bunları yanıtlamakta veya silmektedir. E-posta, bir zamanlar telefonun olduğu gibi ,bu günün dünyasında en önemli iletişim aracıdır. E-posta ile pazarlamada bu aracı kullanarak pazarlama yapma yöntemidir.E-posta ile pazarlamada amaç müşteriye beklemek yerine ona direk olarak ulaşabilmektir.Böyle bir pazarlama ile elde edilen bir çok avantaj bulunmaktadır. Pazarlamanın amaçlarını şöyle sıralayabiliriz:

- Müşteri isteklerini anlayabilmek
- Mal veya servis hakkında müşteri memnuniyetini ve kaliteyi anlayabilmek.
- Müşteri davranışlarını anlayabilmek ve sınıflandırabilmek.
- Yeni stratejiler belirleyebilmek.

Bu tip verilerle , bir iş alanı kendisiyle ilgili aşağıdaki analizleri yapabilir:

- Müşterileri istek ve ilgilerine göre sınıflandırabilme.
- Ürünlerin değerini anlayabilme ve rakiplerle karşılaştırabilme.
- Ürünler hakkında karar verebilme.
- Ürünlerin özelliklerini değiştirebilme.
- Geleceğe dönük planlar yapabilme

Bunlarla beraber e-posta ile pazarlama tüm bu sonuçlara hızlı bir şekilde ulaşmayı sağlar.En basit olarak,e-posta ile pazarlamada müşteri kendisinin ilgilendiği bir konuyla ilgili olarak bir e-posta alır.Bu kişi e-postayı gönderen iş alanı tarafından bilinen bir müşteri olabileceği gibi kendisi hakkında hiç bir şey bilinmeyen biride olabilir.Bu e-postaya müşterinin göstereceği tepki, pazarlamanın başarısını belirler.Bu noktada bazı unsurların göz önüne alınması gerekir.Bir kişiye e-posta göndermek o kişinin bu postayı okuyacağı anlamına gelmez. Her gün, binlerce e-posta hiç okunmadan silinmektedir.

Daha önemlisi bazı e-postalar, e-posta sunucuları tarafından analiz edilmekte ve uygun görülmeyenler tamamen ret edilmektedir. Dolayısıyla her ne kadar E-posta ile pazarlama bir çok avantaja sahip olsa da , sadece kurallarına uyularak yapılan bir pazarlama başarılı olabilir.

2.2.1. E-Posta İle Pazarlamanın Kuralları

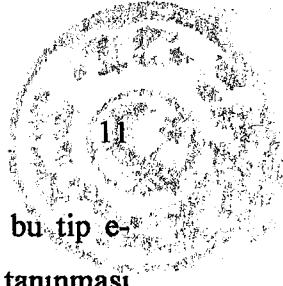
E-posta ile pazarlama eğer doğru kullanılırsa önemli bir pazarlama gücüdür. Eğer kurallarına uyulmazsa kolaylıkla spam olarak değerlendirilebilir. Spam alıcıların bilgi ve onayı olmadan gönderilen ticari amaçlı e-posta şeklinde tanımlanabilir.[3]

Bir kişinin e-posta adresini bilmek ona pazarlama amaçlı e-posta atabilmek için yeterli değildir. Aşağıdaki durumlar da kişiye pazarlama postası göndermek için yeterli şartlar yoktur:

- Bir e-posta adresini web sitesinde görmek.
- Bir forum içerisinde kişinin ilgi alanlarını açıkladığı bir e-postasını görmek.
- Tartışma gruplarından e-posta adresi toplamak.
- Hedef kitlenin e-posta adreslerini satın almak.

Bütün bu durumlar, e-postanın sahibinin izni olmadığı için e-posta ile pazarlama için yeterli değildir. E-posta sahibi gönderilecek e-postayla ilgili hiçbir şey bilmemekte, tamamen istemi dışında e-posta almaktadır. Bununla beraber , e-posta sahibi eğer servis aldığı şirketle e-posta sayısına göre bir ücretlendirmeye anlaşırsa, gönderilen pazarlama amaçlı e-postalar e-posta sahibinin para harcamasına da sebep olacaktır.[3] e-posta ile pazarlamanın başarılı olabilmesi için 2 faktör vardır[3]:

1. Pazarlama amaçlı e-posta gönderebilmek için e-posta sahibinin izni.
2. Bir e-postaya sahip olmak.



Kişilere pazarlama amaçlı e-postalar gönderirken , herhangi bir anda bu tip e-postalar almayı engelleyici olanaklarında e-posta sahiplerine tanınması gereklidir.Böylelikle ,e-posta sahipleri istedikleri zaman kendilerini pazarlama sisteminin dışına çıkarabilirler

2.2.2. Geleneksel Pazarlama İle E-Posta İle Pazarlama Arasındaki Farklar

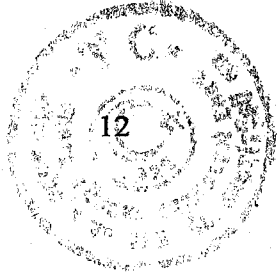
Geleneksel posta ile pazarlama pahalı bir yöntemdir ve gönderilen postaların %3 veya 4 üne yanıt alınırsa bu başarı sayılır.Oysa e-posta ile pazarlamada %15 -%30 arasında değişen bir yanıt alabilme oranı mevcuttur.[3]

Ayrıca :

- E-postaya yanıt vermek normal bir postaya yanıt vermekten çok daha kolaydır.
- E-posta ile pazarlama, eğer doğru yapılırsa , hedef müşteri kitlesiyle birebir ilişkiyi sağlar.

2.2.3. Müşteri İlgisini Anlayabilme, E-Posta İçeriği

Pazarlama amaçlı e-postanın içeriği belirli bir konu hakkında kişinin ne düşündüğünü anlamaya yöneliktir.Bu amaçla e-posta içerisine her türlü değişik tip veri konulabilir. Burada amaç kişinin ilgisini bu e-postaya yanıt verebilecek kadar çekebilmektir.



3. E-POSTA İLE PAZARLAMA ÇÖZÜMÜ

3.1. E-Posta İle Pazarlama Yazılımı Nedir?

E-posta ile pazarlama yazılımı web tabanlı olarak geliştirilmiş bir yazılımdır. Yazılım bir web tarayıcısı ile istenirse tek başına kullanılabilir veya bir ASP (uygulama sunucusu sağlayıcısı) içerisinden servis kiralanarak kullanılabilir. Geliştirilen yazılım, bir iş alanının pazarlama ihtiyaçlarını karşılamak amacıyla tasarlanmıştır. Yazılım müşteri ile iş arasındaki haberleşmeyi sağlamaktadır ve pazarlamayı Internet üzerinde mümkün kılmaktadır.

Yazılım hazırlanırken kullanıcıların Internet tarayıcılarını kullanmayı biliyor olmaları gerekliliği göz önüne alınmış ve yazılım mümkün olduğunca kolay kullanılır ve anlaşılır şekilde tasarlanmıştır.

Yazılımın yetenekleri , benzer çözümlerin iyi ve kötü yönleri incelenerek geliştirilmiş ve benzer yazılımların bu alandaki eksik kalan yönlerini tamamlamak amacıyla geliştirilmiştir.

3.2. E-Posta İle Pazarlama Yazılımının Mimarisi

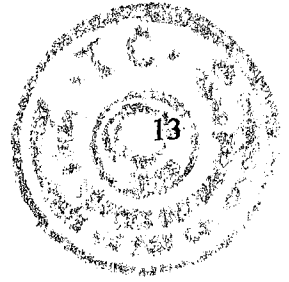
Yazılım , nesne-yönelimli bir sistemdir. Sistem analizi ve tasarımı nesne-yönelimli metodoloji kullanılarak yapılmıştır. Programlama dili olarak java kullanılmıştır. Mimari, çok-katmanlı mimariye göre tasarlanmıştır. Sistemin ihtiyaç duyduğu bileşenler aşağıdaki gibi sıralanabilir:

1. Kullanıcı

Internet explorer , netscape navigator veya benzerleri

2. Web Sunucusu

Java ve servlet çalıştırabilen herhangi bir web sunucusu



3.Uygulama Sunucusu

J2ee uyumlu web-uygulamam sunucuları.

4.Veritabanı Sunucusu

jdbc sürücüsü olan ilişkisel veya nesne veritabanları

Yazılım , veritabanı olarak informix,oracle,sqlserver üzerinde çalışmaktadır. Web uygulama suncusu olarak IBM-Websphere , BEA-Weblogic ,JRUN Tomcat sunucularında test edilmiştir.

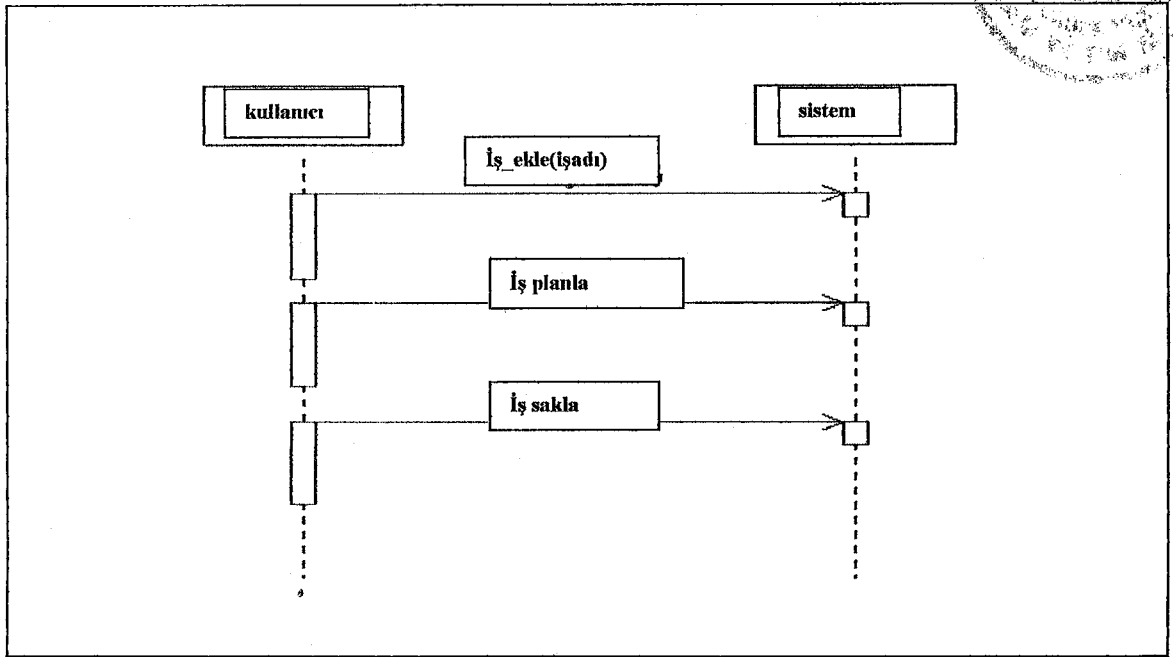
Java dilinin getirdiği esneklik ve standard sayesinde değişik ortamlara aktarılması edilmesi mümkündür.

3.3. E-Posta İle Pazarlama Yazılımının Sistem Analizi

Yazılımın analiz ve tasarımında UML, use case diyagramları , sistem sıra diyagramları kullanılmıştır.Şekil.3.1’de bir iş yaratmanın genel use case diyagramı gösterilmiştir.

Kullanıcı İşlemi	:Yeni bir iş yaratma
Aktörler	:Kullanıcı
Tip	:İlk, Temel
Tanım	:Bu işlem kullanıcı yeni bir iş yaratma opsiyonunu seçince başlar. Sistem, kullanıcının iş tipini,kategorileri, çalışma zamanını Seçmesini sağlar.Bitiminde , yaratılan iş , ilgili hedeflere gönderilmek üzere işleme konur .
Referanslar	:R1,R4,R7,R11,R12,R13,R14,R15
Diğer Use caseler:	Bu işlemin yapılabilmesi için kullanıcın “sisteme login olma use caseini” tamamlamış olması gerekir.

Şekil.3.1. Kullanıcı İşlemi Diyagramı



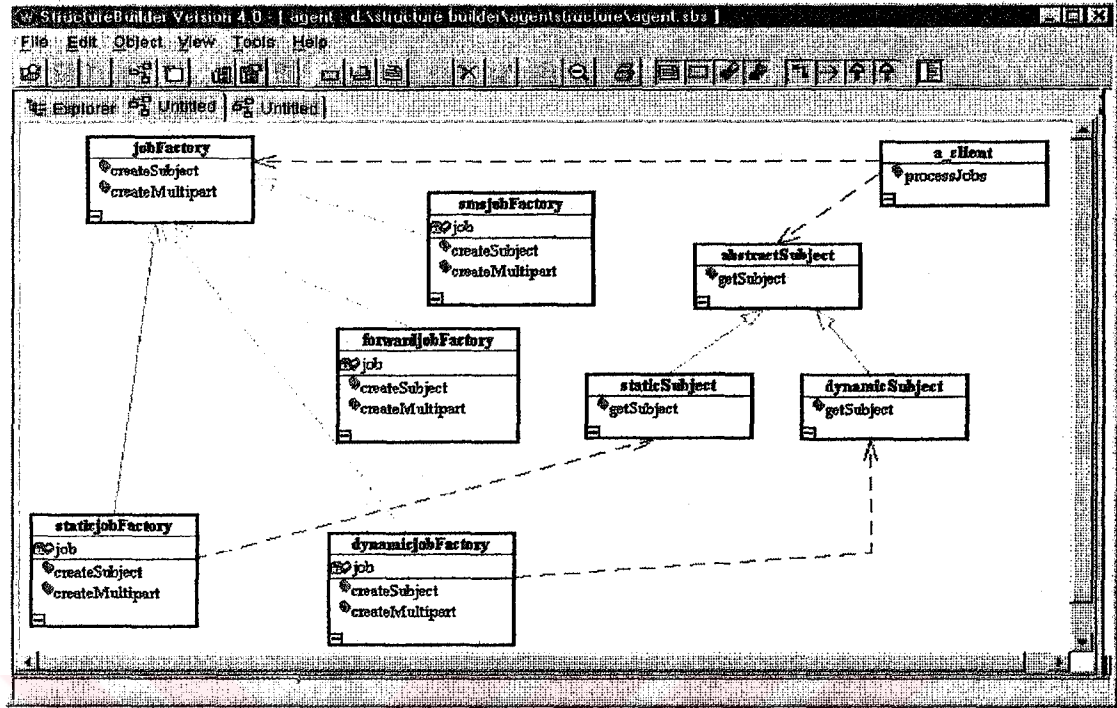
Şekil.3.2. Sistem Sıra Diyagramı

Sistemin kavramsal modeli Şekil.3.3'te görülmektedir:



Şekil.3.3. Kavramsal Model

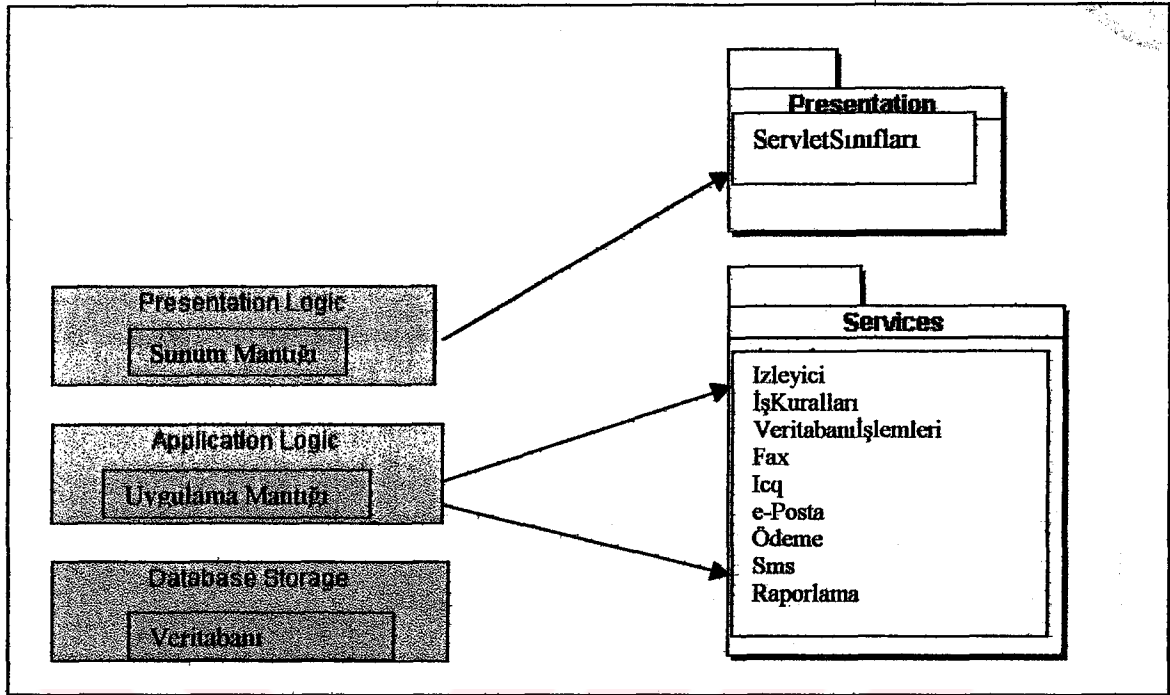
Sistemin grafik ara yüzü dışında kalan ve iletileri gönderme işini yapan diğer bir kısmı da mailAgent kısmıdır. Bu bölüm multi-threaded özelliklere sahip bir modüldür. Tasarımında yazılım tasarım modellerinden yararlanılmıştır. Abstract factory, Compositor ve Command tasarım modelleri kullanılan başlıca modellerdir. Şekil.3.4'te değişik iletilerin gönderilmesine ait UML diyagramı görülmektedir.



Şekil.3.4. İleti Gönderme UML Diyagramı

3.4. E-Posta İle Pazarlama Yazılımının Teorik Yapısı ve Algoritmalar

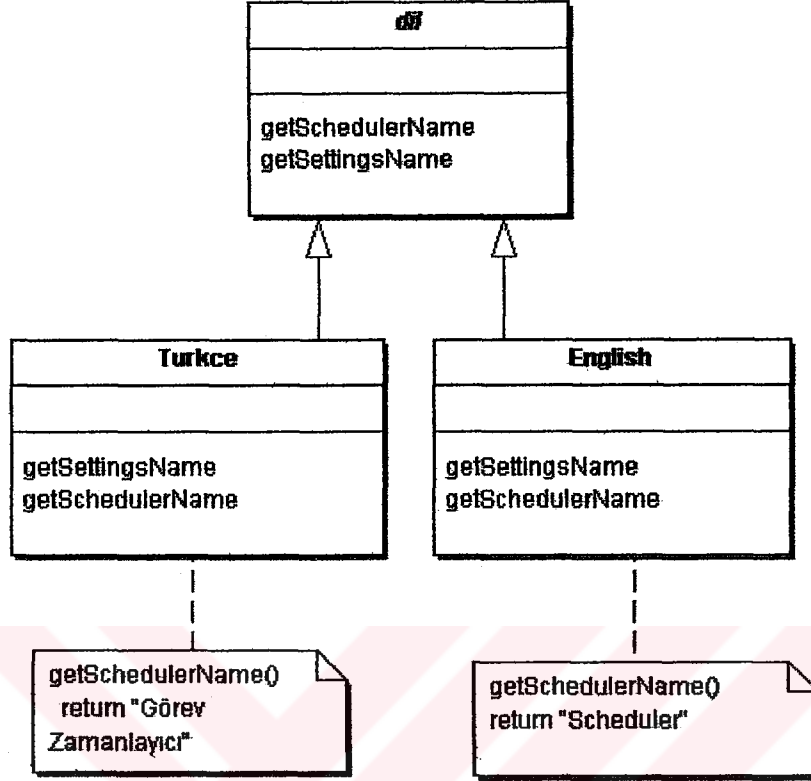
Yazılım n-katmanlı yapıya göre tasarlanmış bir mimariye sahiptir. Buna göre yazılımın temelde Sunum Mantığı, Uygulama Mantığı ve Veritabanı olmak üzere 3 dikey katmanı bulunurken, Uygulama katmanı kendi içerisinde de yatay olarak katmanlara ayrılmıştır.



Şekil.3.5. Yazılımın Genel Yapısı

Sunum mantığı içerisinde , kullanıcıların gördükleri ve daha önceki bölümlerde anlatılan arayüzlerin hazırlandığı ve java servlet oluşturan java classları bulunmaktadır. (ServletCorePackage)

Kullanıcıların arayüzleri oluşturulurken karşılaşılan en büyük sorun , uygulamanın değişik dillerde kullanımı desteklemesi amacına uygun bir şekilde nasıl tasarlanacağı olmuştur. Bu amaçla , geliştirilen algoritma , temel de “abstract factory” tasarım modelini kullanan ve nesne yönelimli tasarımın OCP (open closed principle – açık kapalı prensibi) mantığına uygun olarak , istenilen sayıda dile destek verme imkanını kullanıcı arayüzünü oluşturan kodlarda bir değişiklik yapmadan sağlayabilmektedir.



Şekil.3.6. Değişik Dillere Destek Verebilme

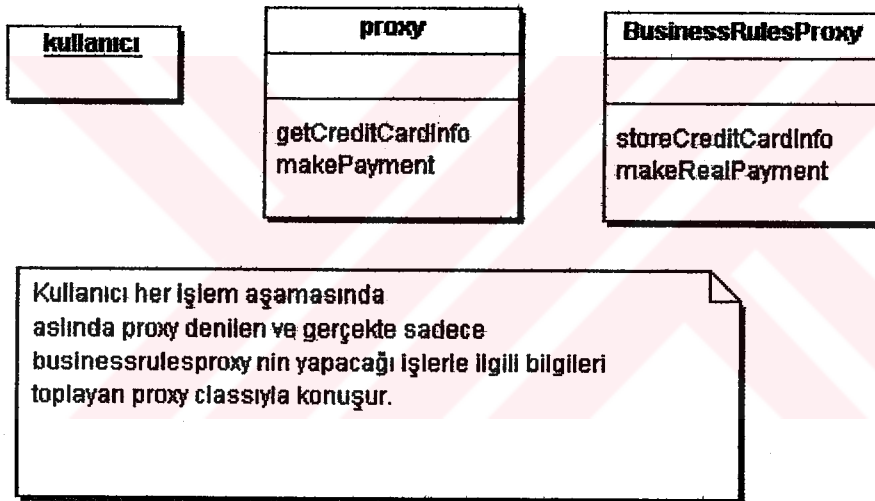
Şekil.3.6’da da gözüktüğü gibi arayüzleri oluşturan tüm class lar , dil adındaki soyut classın fonksiyonlarını kullanmaktadırlar. Uygulama çalışma sırasında, kullanıcının çalışmak istediği dile göre dil classından türeyen Turkce veya English classlarını yaratarak , istenilen dille çalışmayı herhangi bir şekilde kod tekrarı yapmadan sağlar.

Bu yapı içerisinde , sisteme yeni bir dil eklemek , sadece dil classından yeni bir class türetmekten ibarettir. Uygulama bu algoritmayı bir çok başka yerde de kullanmıştır.Uygulama Mantığı içerisinde yatay olarak bir katmanlama söz konusudur.Bu katmanlar içerisinde her bölüm kendisine ait algoritmalar içermektedir.

İş Kuralları: Bu katmanda bulunan classlar uygulama katmanından gelen istekleri gerçekleştirmekle görevlidir.Buna göre yeni bir iş yaratılmak istenildiğinde , sunum katmanında bulunan servletler , işkuralları katmanındaki classları çağırırlar.

Kullanıcı arayüzlerinde genelde birbirini takip eden işlemler bulunmaktadır. Yeni bir iş yaratma veya kredi kartı ile ödeme yapma aşamalı olarak gerçekleştirilen işlemlerdir. Yazılımda bu aşamada sunum katmanından , uygulama katmanına bir geçiş söz konusudur. Bu amaçla, tasarım aşamasında aşağıdaki sebeplerden dolayı proxy tasarım modelinin kullanılmasına karar verilmiştir:

- Aşama aşama gerçekleşen işlemlerin , bir veritabanı transactionı gibi değerlendirilebilmesi
- Uygulama katmanı ve sunum katmanının istenirse ayrı sunuculardan çalıştırılabilmesi
- Değişik dağıtık sistem ortamlarına (Corba veya Rmi) uyumluluk.



Şekil.3.7. İş Kuralları Çalışma Prensipleri

Veritabanı İşlemleri: Veritabanı işlemleri classları , işkuralları katmanındaki classlar tarafından kullanılır ve veritabanı ile ilgili sorgulama işlemlerini yapmakla yükümlüdür. Böyle bir katmanın tasarlanmasında amaç sistemi değişik veritabanlarıyla çalışabilir hale getirmektir. Teorik olarak , j2ee uyumlu olan web uygulama sunucuları, kendi içlerinde veritabanlarından bağımsız çalışabilmeyi , CMP (container managed persistency) mantığı ile sağlamakta olsalarda , bugünkü teknoloji içerisinde pratikte bir çok sorun çıkmaktadır. Yazılımın kullandığı veritabanı işlemleri katmanı , sadece bu yazılımın değişik veritabanlarıyla kullanımını sağlayabilmek amacıyla tasarlanmış olup ,

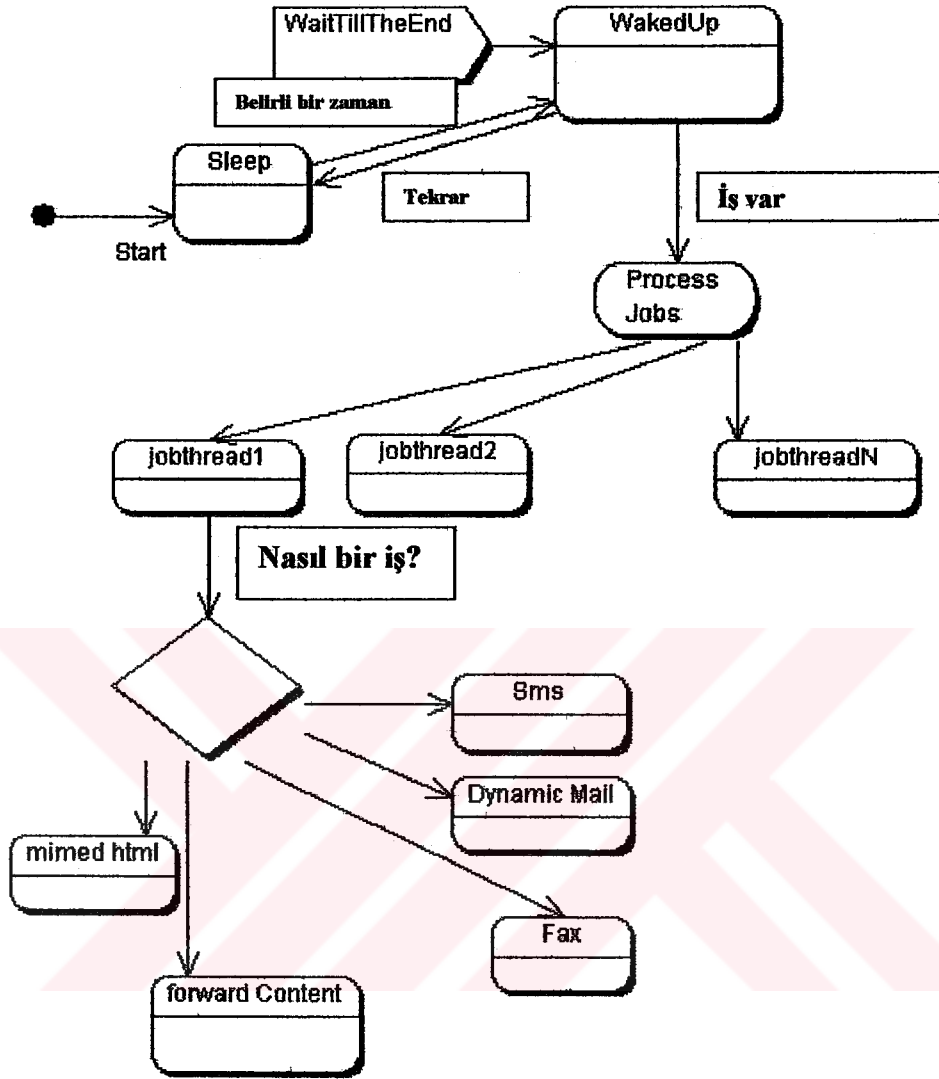
genel kullanımlı ve j2ee evamlılık kurallarına uygun değildir.Burada da abstract factory tasarım modeli kullanılarak , benzer algoritmalar geliştirilmiştir.

İzleyici: Yazılım , kendi içerisinde multithreaded yapıda çalışan bir gönderi yapma motoruna sahiptir.Bu motor , gönderileri paralel bir şekilde işleme yeteneğine sahip olduğu gibi,istenirse farklı gönderileri farklı sunuculara yönlendirme kabiliyetine de sahiptir.

Şekil.3.8’ de görüldüğü gibi , gönderileri yapmak amacıyla tasarlanan izleyici , multithreaded yapıda çalışan bir server programdır. 15 dak. lık periyotlarla çalışan izleyici,her bir çalışmada , işlenmesi gereken bir iş olup olmadığına bakar. Eğer işlenmesi gereken iş lar varsa, her bir iş için ayrı bir thread yaratır ve tekrar “sleep” durumuna geri döner.Bu aşamadan sonra her bir işthreadi , iş in türüne göre ilgili classı yaratır (her bir iş threadi bir factory yaratma özelliğine sahiptir) ve işlem daha sonra bu seviyeden itibaren gönderi türüne göre değişik aşamalara geçerek devam eder.Buna göre eğer bir e-posta gönderilecekse , e-posta servisleri, sms gönderilecekse sms servisleri,icq durumunda icq servisleri veya fax gönderilecekse fax servisleri kullanılır.Yine burada da ileride kullanılacak bilinen veya bilinmeyen gönderi iletme ortamlarında düşünülerek , abstract factory tasarım modeli kullanılmıştır.

HIZLI POSTA GÖNDERME ALGORİTMASI

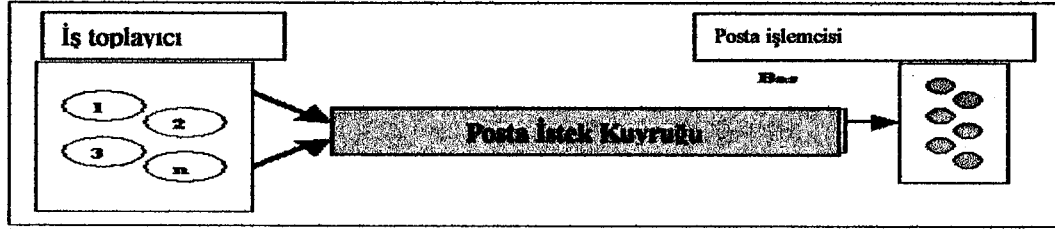
Yazılım geliştirilirken göz önünde bulundurulacak konulardan bir tanesi de, yüksek hacimde iletiyi veya iletileri hızlı bir şekilde yapabilme özelliğidir. Bu amaçla , işmine “hızlı posta gönderme algoritması” dediğimiz bir matematik model geliştirilmiştir.Bu model tamamen paralel işlem yapma prensibi üzerine kurulmuş olup , miktarı on binlerle belirtilen posta adreslerine makul sürelerde gönderi yapabilmeyi amaçlamıştır.



Şekil.3.8. İzleyici Yapısı

Geliştirilen algoritma , iletileri ; sayıları ne olursa olsun en kısa sürede gönderebilecek şekilde tasarlanmıştır.Algoritmanın temel prensibi paralel işleme mantığına dayanmaktadır.Posta gönderme işlemi blok bir işlemdir, gönderiyi yapan thread , işletim sistemi tarafından çalışır durumdan çıkartılır. Bu yüzden bu zaman süresini , diğer threadler tarafından efektif bir şekilde kullanmak mümkündür.Yazılımın çalıştığı sistem tek bir cpu ya sahip olsa bile , blok zamanını diğer threadlere kullandırmak büyük bir hız sağlamaktadır.

Bir n anında gönderilmek amacıyla yaratılan bir gönderi , t zaman aralıklarıyla çalışan pencereler içerisinde işlenmektedir.



Şekil.3.9. Hızlı Posta Gönderme Sistemi

Posta İstek Kuyruğu (MRQ), gönderilerin biriktirildiği kuyruk sistemi olup, buradaki her bir üye, bir domaini ve o domaine gidecek gönderileri temsil etmektedir. Böylece, sistem , domainlere göre gönderileri gruplamaktadır. İş toplayıcıları (JC) herhangi bir zamanda yapılan gönderi isteklerini analiz ederek , bu istekleri domainlere göre gruplamakta ve MRQ'ya iletmektedir. Posta prosesörleri (MP) ise, MRQ dan istekleri paralel olarak almakta ve işlemektedirler. Bir isteği gerçekleştiren MP, eğer MRQ da bekleyen bir başka iş varsa bunu işleyerek çalışmaya devam etmektedir. Sistem ayrıca domainlere kurulan tcp bağlantılarını saklayarak tekrar tekrar kullanmakta , böylece gereksiz bağlantılardan kaçınmaktadır. Sistem gönderi miktarına göre akıllı kararlar vererek optimum sayıdaki Mp yi konfigure edebildiği gibi , kullanıcının isteğine bağlı olarakta bu sayıyı belirleyebilmektedir. Tasarlanan algoritmanın matematik modeli şöyle açıklanabilir:

N : MP sayısı.

M : t anındaki MRQ eleman sayısı .

ti : bir MRQ isteğinin ortalama işlenme süresi

Tavg : MRQ daki tüm işlerin ortalama bitirilme

Zamanı

P : Paralel işlem yapabilme katsayısı

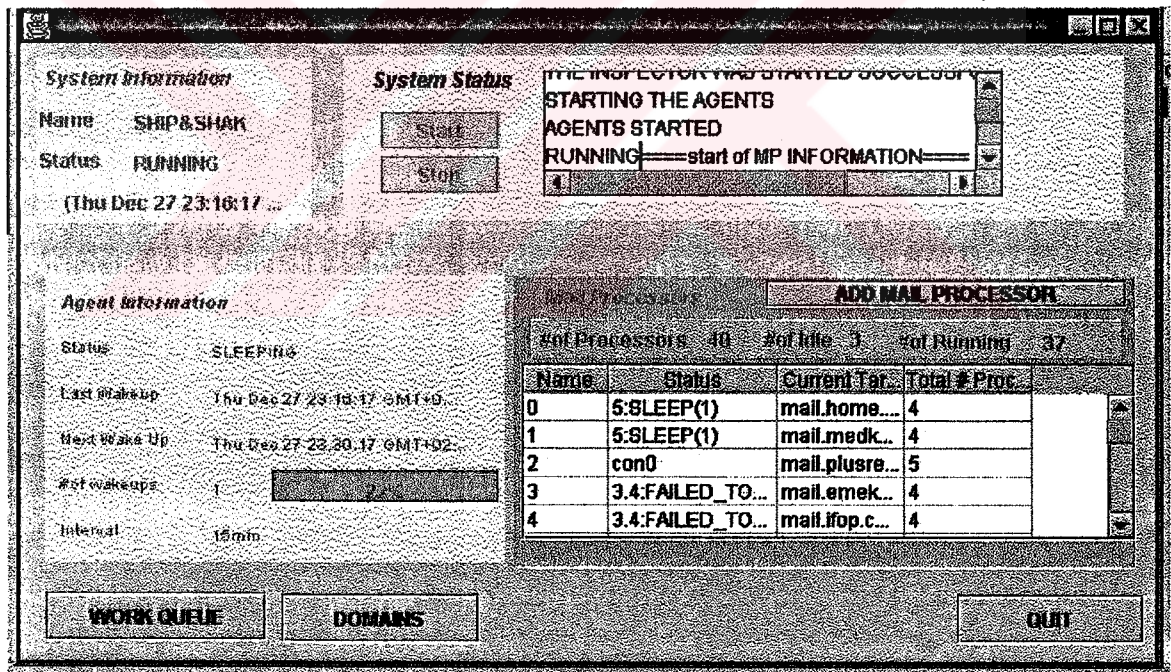
O : fazlalık

$$Tavg = ti * (M/N) * P + O \quad (N < M \text{ ve } N > 0)$$

Sistem, Tavg yi minimize ederek en kısa surede işlemleri bitirmeyi hedefler. Buna göre: Tavg değeri , ideal olarak (P=1) N, M'e yaklaştıkça azalır ve N=M olduğunda , yani sistem kuyruktaki iş kadar MP ile paralel işlem gerçekleştirdiğinde bir tek işi bitirme zamanına eşit olmaktadır. (P=1) Yapılan testler sonucunda , paralel işlemin belirli bir noktadan sonra hızlandırıcı etkisini kaybettiği ve aksine yavaşlatıcı bir etki yaptığı görülmüştür. Bu yüzden P adı verilen ve sistemin paralel işlem yapabilme gücünü belirleyen bir etki modelde yer almıştır. Bu katsayı sistemin çalıştığı makinanın CPU sayısına , CPU hızına , işletim sisteminin thread çalıştırma mantığına bağlı olmakla beraber , sistem ilgili katsayıyı değişik gönderilerin zamanlarına göre bulmaktadır:

$$P = t_i * M / (N * Tavg)$$

Bu değer sistem gönderi yaptıkça hesaplanmakta ve 1 e en yakın değere göre sistem MP sayısını belirleyebilmektedir.



Şekil.3.10. Sistem İzleme/Yönetme

Yazılımı, diğer örneklerinden ayıran en önemli özelliği izleyici kısmında sağlayabildiği esnekliktir. Hemen hemen diğer bütün e-posta ile pazarlama yazılımları , bir şekilde kullanıldıkları sistemlere entegre olurken , o sistemin yapısının değişmesini istemektedirler. En basit olarak, diğer yazılımların gönderi yaparken uyulmasını

istedikleri bir takım kurallar vardır.Buna göre , o yazılımların istedikleri şekillerde dökümanlar hazırlanmalı ve ancak bunlar gönderilmelidir.Bütün bunlar göz önüne alınarak, yazılımda herhangi bir kural tanımına fırsat verilmemiştir.Buna göre , kullanıcı kendi kaynaklarıyla hazırladığı bir HTML dökümanı , geliştirdiğimiz yazılımı kullanarak gönderebilir. Burada karşılaşılan en büyük sorun , html dökümanlarında bulunan linkler ve şekillerdir:

```
<html>
<body>

<a href=www.somewhere.com>Somewhere</a>
</body>
</html>
```

Yukarıda görülen html dökümanını eğer kullanıcı isterse, sistem olduğu gibi gönderebilir.Fakat görüldüğü gibi bu gönderiyi alan kişi , gönderinin posta olduğu durumu düşünürsek , hiç bir şekilde test.jpg dosyasını göremeyecektir.Çünkü , bu dosya iletiyi hazırlayan kişinin makinasında bulunan bir dosyadır.Yazılım , html dökümanlarını analiz ederek , bu tür referansları bulmakta , bunları kullanıcıya bildirmekte ve sonrada kullanıcı isterse bunları yüklemektedir. Bu aşamadan sonra 2 seçenek mevcuttur:

- Resimleri web server üzerinde linkler halinde göndermek.
- Resimleri de html ile beraber göndermek.(mimed html)

Geliştirdiğimiz yazılım , böylece bir html i analiz edip , içerisindeki grafikleri anlayabilme ve bu grafiklerle beraber hitmli tek bir posta ile gönderebilmektedir.

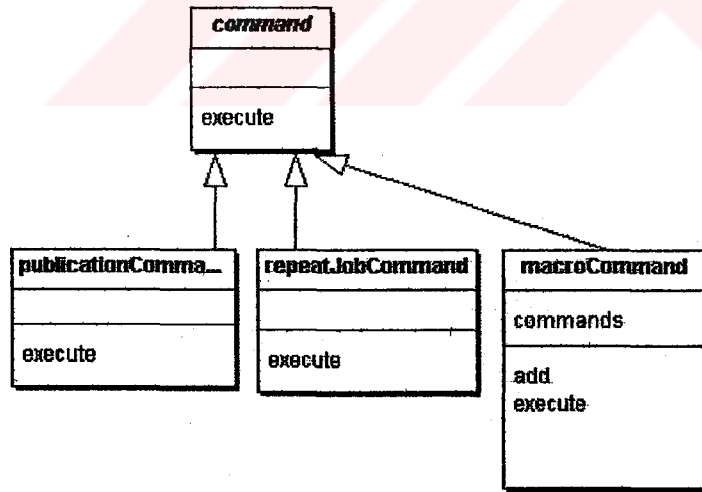
Bunların yanında, yukarıdaki örnekte görüldüğü gibi , eğer href ile gösterilen link olduğu gibi bırakılırsa , bu linkin iletiyi alan kişi tarafından ziyaret edilip edilmediğini anlamak mümkün değildir. Bu amaçla , yazılım bu tür linkleride değiştirip ,

bunları kendi üzerinden yönlendirmektedir. Böylece, bir iletiyi alan kişinin iletisi içerisindeki linklere kaç kere gittiği, ne zaman gittiği gibi bilgilerde ulaşılabilir. İzleyici servisleri ayrıca yeni işlerde yaratabilirler. Standard olarak bir ileti hazırlandığında, aşağıdaki 3 şekilde gönderilmesi mümkündür:

- Hemen Gönder
- Daha Sonra Gönder
- Tekrarlayarak gönder

İzleyiciler, tekrarlayarak gönderme işlemi denilen :

- Ayın belirli bir gününde ve/veya belirli bir saat veya saatlerde gönderme
 - Haftanın belirli bir gününde ve/veya belirli bir saat veya saatlerde gönderme
- durumlarını kontrol eder. Bununla beraber, burada geliştirilen algoritma, ayrıca bir gönderinin içeriğinde değişiklik meydana geldiği durumda da gönderi yapabilmek olanağını sağlamaktadır. Bu kısımda, command tasarım modeli kullanılmıştır:



Sırayla çalışması gereken "tekrarlı işler" yaratılırken kullanılan Command design patterni: Macrocommand classı içerisinde diğer command classlarını tutuyor ve bunları sırayla çalıştırıyor.

Şekil.3.11. Tekrarlı İş Yaratma Yapısı



Ödeme:

Yazılım , kullanıcıların aldıkları servislerin karşılığında , kredi kartları ile ödeme yapabilmelerine olanak tanır. Ödemeler 2 şekilde gerçekleştirilmektedir.

- Her ay sonunda yapılan gönderi miktarına bağlı olarak ücretlendirme karşılığı ödeme.
- Gönderi sırasında ödeme.

Bu şekilde , yüksek hacimlerdeki gönderiler için yapılan fiyatlandırma ile düşük hacimli gönderilerin fiyatlandırılması birbirinden ayrılarak cazip kullanımlar oluşturulması amaçlanmıştır.

3.5. E-Posta İle Pazarlama Yazılımının Özellikleri

Yazılımın özellikleri şöyle sıralanabilir:

- Değişik iş alanlarına adaptasyon
- Genişletilebilirlik
- Dinamik kategori işlemleri
- Dinamik e-posta işlemleri
- SMS gönderebilme
- FAX gönderebilme
- ICQ mesajı gönderebilme
- Özel durumlarla ilgili e-posta yaratabilme
- Var olan datayla entegrasyon
- Çoklu çalışabilme
- Sihirbazlarla kolay kullanım
- Kullanıcı hareketlerini izleyebilme
- Kullanıcılara sistemle ilgili seçenekler sunabilme

3.5.1. Değişik İş Alanlarına Adaptasyon

Daha önceki bölümlerde bahsedildiği gibi, her iş alanının pazarlama ihtiyaçları birbirinden farklılıklar göstermektedir. Yazılım sistemi, her iş alanının ihtiyacına göre konfigüre edilebilmektedir.

3.5.2 Genişletilebilirlik

Yazılım , eğer istenirse şirket bünyesinde kullanılabilir.Bununla beraber , bir ASP bünyesinde de kullanımı mümkündür.Böylece bilgi işlem konusunda uzmanlığı olmayan şirketlerde bu yazılımdan faydalanabileceklerdir.(Böyle bir çalışma sistemi , bünyesinde bilgi işlem personeli bulundurmayan şirketler için , profesyonel olmadıkları konuları uzmanlarına bıraktığı için avantajlıdır.)Buna örnek olarak veri saklamanın karmaşıklığı gösterilebilir “Veri saklama bir çok bağımsız ve karışık sistemi bir araya getirip düzensizlikten bir düzen yaratma işlemidir.”[6]

3.5.3. Dinamik Kategori İşlemleri

Kategori kişilerin ilgi duydukları mesela müzik gibi alanlardır.Yazılım , hedef kitlesini kategorilere göre belirler.Var olan çözümlerin kullandığı sabit kategori mantığından farklı olarak , yazılım kategorilerin silinmesine, yeni kategorilerin eklenmesine izin vermektedir.Böylece değişik iş alanları, kendileriyle ilgili kategorileri tanımlayabilirler.

3.5.4. Dinamik E-posta İşlemleri

Yazılım ,hedef kitleye e-posta gönderirken statik ve dinamik olmak üzere 2 seçenek sunmaktadır.Statik e-posta ile e-postanın içeriği olduğu gibi gönderilir. Dinamik postayla ise , e-posta içeriğine özel alanlar eklemek ve gönderme aşamasında bu alanların ilgili data kaynaklarından alınarak değiştirilmesi mümkündür. Böylelikle,

mesela her alıcının e-posta içerisinde kendisiyle ilgili kişisel bilgiler görmesi mümkündür.

3.5.5. SMS Gönderebilme

SMS'te e-posta kadar popüler bir iletişim aracıdır. Bu yüzden sistem hazırlanan e-postaların sms olarakta gönderilebilmesine olanak tanınmaktadır.

3.5.6 - 3.5.7. FAX , ICQ Gönderebilme

Benzer şekilde yazılım fax veya icq mesajlarında gönderebilmektedir.

3.5.8. Özel Durumlarla İlgili E-Posta Yaratabilme

Yazılım doğumgünü, bayram gibi özel günlerde ilgili alıcıya e-posta gönderebilme imkanı tanımaktadır. Sisteme yeni durumlar tanıtılabilmekte ve bunlar aktif veya pasif hale getirilebilmektedir.

3.5.9. Varolan Bilginin Sisteme Eklenmesi

Yazılım , her sistemin kullandığı hazır bir veri tabanı olduğu gerçeğini göz önünde tutarak , var olan veri tabanındaki bilgiyi sisteme aktarabilme özelliğine sahiptir. Şu anda , text formatındaki veya Microsoft-CSV formatındaki data sisteme sihirbazlar yardımıyla eklenebilmektedir.

3.5.10. Çoklu Çalışabilme

Eğer istenilirse, yazılım birden çok sistemi aynı anda çalıştırabilir, içerisinde temel veriler dışında veri olmayan sistemler yaratabilir.

3.5.11. Sihirbazlarla Kolay Kullanım

Bir yazılımın son kullanıcıları , o yazılımın iyi olup olmadığına karar verirken , yazılımın iç yapısıyla ilgilenmezler. Yazılımın karmaşıklığı kullanıcılar için bir şey ifade etmez.Önemli olan şey yazılımın istenilenleri yapabildiği ya da yapamadığıdır.Kullanımı kolaylaştırmak amacıyla, yazılım bir takım sihirbazlar kullanmaktadır.Adım adım e-posta veya sms hazırlayabilme e-posta sihirbazı veya yeni bir sistem oluşturmaya yarayan yeni sistem sihirbazı örnek olarak verilebilir. İlgili grupları veya müşteri ilgileri sistemin ana konularıdır, Sistemin temel amacı, kullanıcıların müşterilere pazarlama postaları göndermesidir. Sistem kullanıcıları kendi müşterilerin sisteme yükleyerek, ilgili gruplarını tanımlayabilir.

3.5.12. Kullanıcı Hareketlerini İzleyebilme

Yazılım , hedef kitleye gönderdiği e-posta ların içersinde bulunan linkleri kullanıcının ziyaret edip etmediğini izler , bu ziyaretlerle ilgili zaman ve adet sayısını rapor halinde sunar.

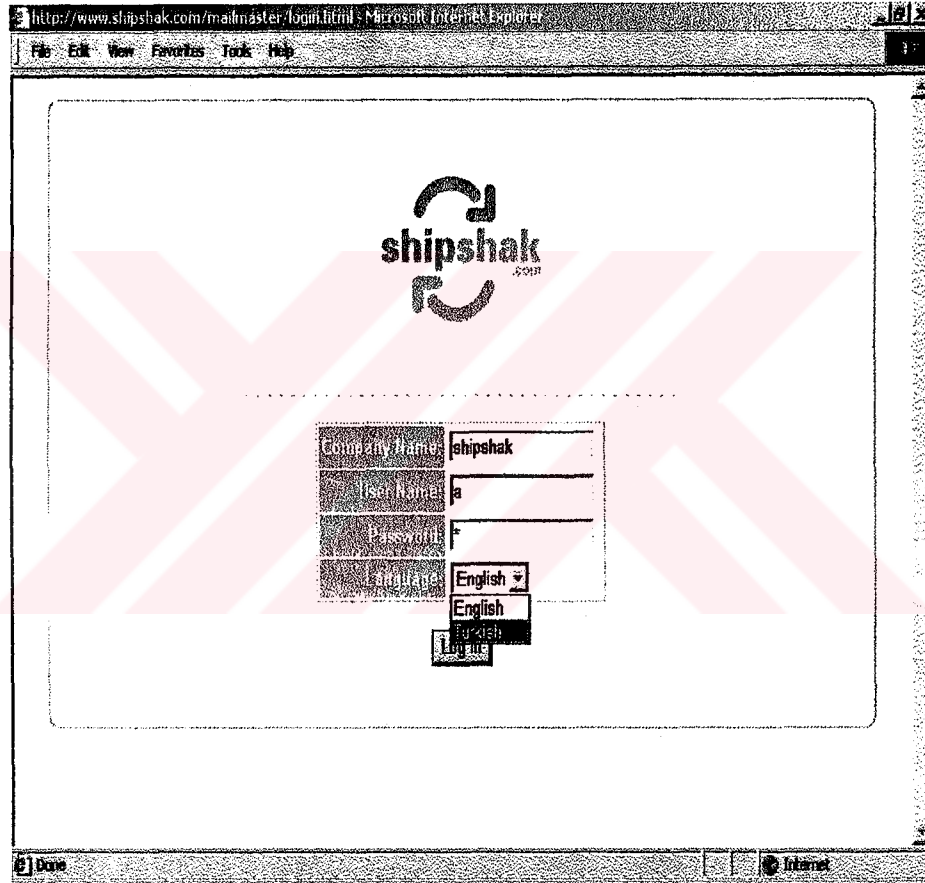
3.5.13. Kullanıcılara Sistemle İlgili Seçenekler Sunabilme

Yazılım , gönderdiği her mesajın sonuna kullanıcının sistemden çıkmasını , bilgilerini güncellemesini veya sisteme yeni bir kullanıcı kaydetmesini sağlayan bir menü ekleyerek , kullanıcının sistemi istediğinde bırakmasını veya bilgilerini değiştirebilmesini sağlar.

4. UYGULAMALAR VE SONUÇLAR

4.1. E-Posta İle Pazarlama Yazılımının Kullanıcı Ara Yüzü

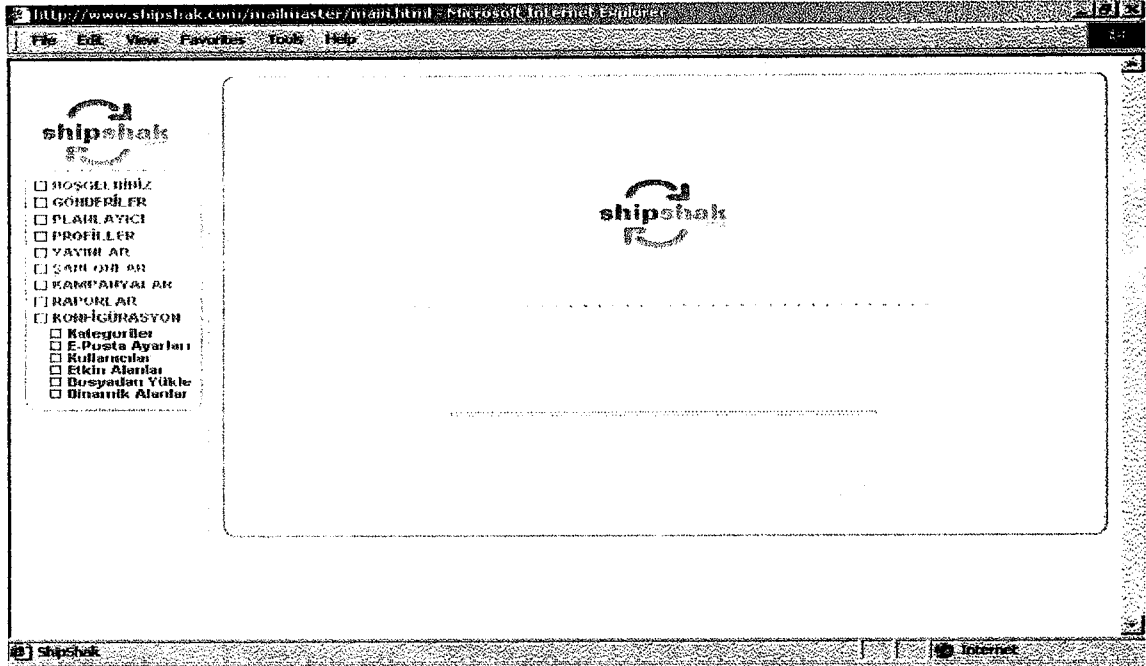
E-posta ile pazarlama yazılımının ilk ekranı kullanıcı adının ve şifresinin girilmek zorunda olduğu giriş ekranıdır. Bu ekran şekil.4.1.'de görülmektedir



Şekil.4.1. Giriş Ekranı

4.1.1. Kullanıcı Ana Menüsü Arayüzü

Arayüz aşağıdaki menülerden oluşmaktadır:



Şekil.4.2. Kullanıcı Ana Menüsü

Her bir menü içerisinde , ilgili işleme ait bir veya birden çok alt işlemler bulunmaktadır.

4.1.2. Posta Sihirbazı Bileşeni

Posta sihirbazı bileşeni ile kullanıcı yeni ileti hazırlar.Yeni bir ileti hazırlamanın 3 aşaması vardır:

- İleti tipini seçerek gönderinin içeriğini belirlemek.
- İletinin kimlere gideceğinin belirlendiği kategorileri seçmek.
- İletinin gönderilme zamanını ve aralığını belirlemek.

Şekil.4.3.'te görüldüğü gibi , bir ileti e-posta , sms ve fax olmak üzere 3 ayrı şekilde hazırlanabilir. Benzer şekilde , her bir ileti yönteminde statik ,dinamik ve yayın olmak üzere de 3 ayrı tip gönderi hazırlanabilir. Statik tip bir ileti , gönderini içeriğinde hiç bir değişiklik yapılmadığı durumdur. Dinamik iletideyse , ileti içerisinde şekil.4.3.'te gözüktüğü gibi bazı alanlar gönderi esnasında değiştirilebilir. Böylelikle

yukarıdaki iletiyi alan her alıcı , name ve lastname kısımlarında kendi isim ve soyadını görebilecektir. Burada kullanılan dinamik alanlar istenildiği gibi konfigür edilebilir.

The screenshot shows a web browser window with the URL <http://www.shiphaktom.com/mail/hesab/main.html>. The browser's menu bar includes File, Edit, View, Favorites, Tools, and Help. The page has a sidebar on the left with a logo and a list of options: ☐ HOŞGEL DİNİZ, ☐ GÖNDERİLER, ☐ E-Posta, ☐ Sabit, ☐ Dinamik, ☐ Şablon Kullan, ☐ Sms, ☐ İÇİ Mesajı, ☐ Faks Mesajı, ☐ PLANLAYICI, ☐ PROFİLLER, ☐ YAYINLAR, ☐ ŞABLOHLAR, ☐ KAMPANYALAR, ☐ RAPORLAR, and ☐ KONFIGÜRASYON. The main content area is titled 'GÖNDERİLER/Dinamik E-Posta'. It contains a form with two input fields, each with the value 'test'. To the right of these fields are buttons for 'Posta Kutusundan', 'HTML Yükle', and 'HTML Editor'. Below the input fields is a large text area containing the text 'Sn. [[name]]' and 'Yeni yılınızı kutlar , mutluluklar dileriz'. At the bottom of the form, there is a 'name' label, a 'Ek Dosya' section with radio buttons for 'Var' and 'Yok', and a 'Devam' button. The browser's status bar at the bottom shows 'Done' and 'Internet'.

Şekil.4.3. İleti Yaratma Bileşeni

Bununla beraber, iletinin içeriğini başka kaynaklardan oluşturmakta mümkündür. Bu amaçla, hazırlanan bir e-posta sisteme gönderilerek sistemin bu e-postayı aktarması yöntemi kullanılabilir. Bu amaç için şekil.4.3.'te ki "forward e-mail" seçeneği kullanılır.

Var olan bir html dökümanının kullanılabilmesi de "upload html" seçeneği ile mümkündür. Ayrıca sistem bu tip dökümanları analiz etmekte , içerisindeki grafikleri anlamakta ve bunları da kullanıcıdan istemektedir. Böylece herhangi bir html



dökümanını içerisindeki bütün grafiklerle beraber posta iletisi haline getirmek mümkün olmaktadır.

slipshak

- ☐ HOŞGELDİNİZ
- ☐ GÖNDERİLER
- ☐ E-Posta
 - ☐ Sabit
 - ☐ Dinamik
 - ☐ Şablon Kullan
- ☐ Sms
- ☐ İCQ Mesajı
- ☐ Faks Mesajı
- ☐ PLANLAYICI
- ☐ PROFİLLER
- ☐ YAYINLAR
- ☐ ŞABLONLAR
- ☐ KAMPANYALAR
- ☐ RAPORLAR
- ☐ KONFIGÜRASYON

GÖNDERİLER

Kategoriler

Operator ☐ Veya ☐ Ve

2) spor ☐ futbol ☐ basketbol

3) müzik ☐ jazz ☐ club

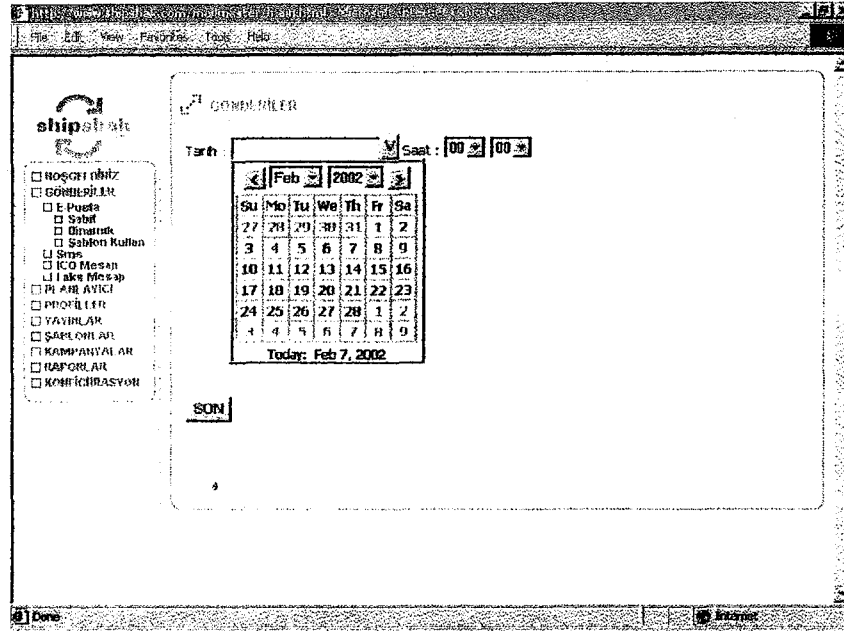
4) database ☐ db2 ☐ oracle ☐ informix

Yayın Adı ☐ test

Devam Herkese Gönder

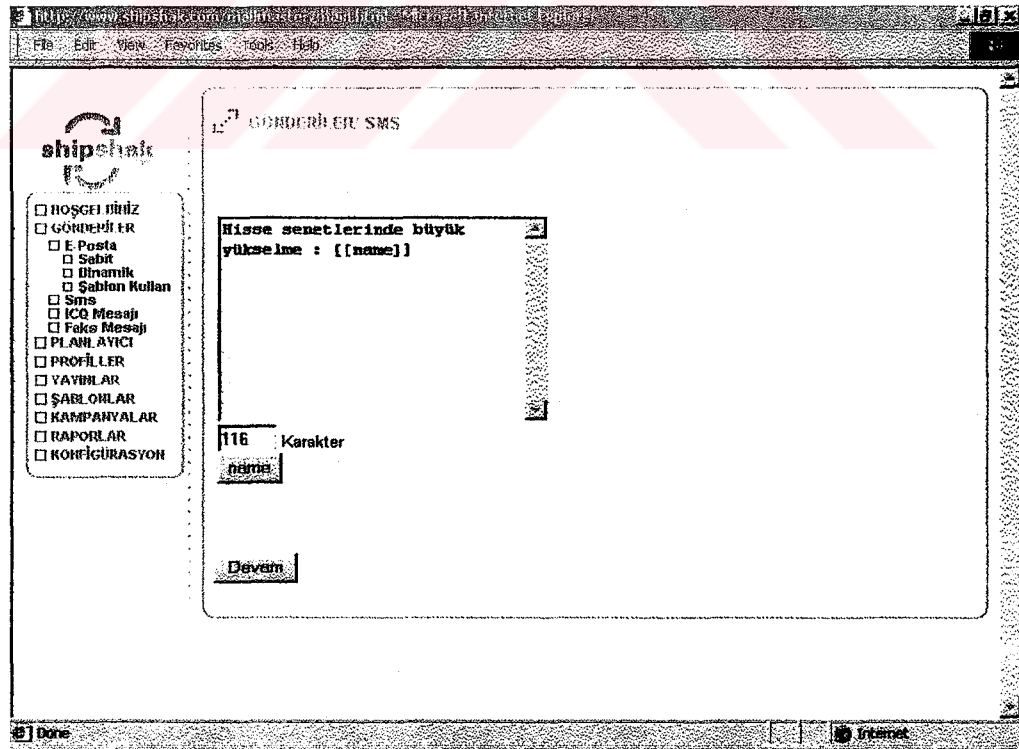
Şekil.4.4. İletinin Hedeflerini Belirleme

Şekil.4.4’de görüldüğü gibi bir iletinin kimlere gideceği kategorilerin seçimi ile mümkündür. İleti istenirse hemen gönderilebildiği gibi , belirlenen bir zamanda da gönderilebilir. Şekil.4.5.de ileti için zaman aralığının belirlenmesi gösterilmektedir.



Şekil.4.5. İletinin Gönderme Zamanını Belirleme

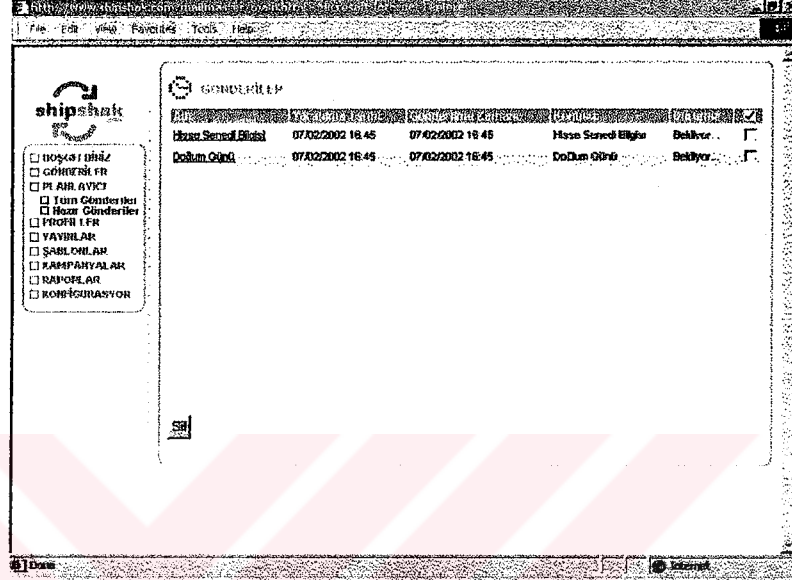
Posta sihirbazı içerisinde e-posta benzeri bir şekilde sms gönderisi de hazırlanabilir. Şekil.4.6.'da sms iletisinin hazırlanışı görülmektedir:



Şekil.4.6. SMS İletisi Hazırlama

4.1.3. Görev Zamanlayıcı Bileşeni

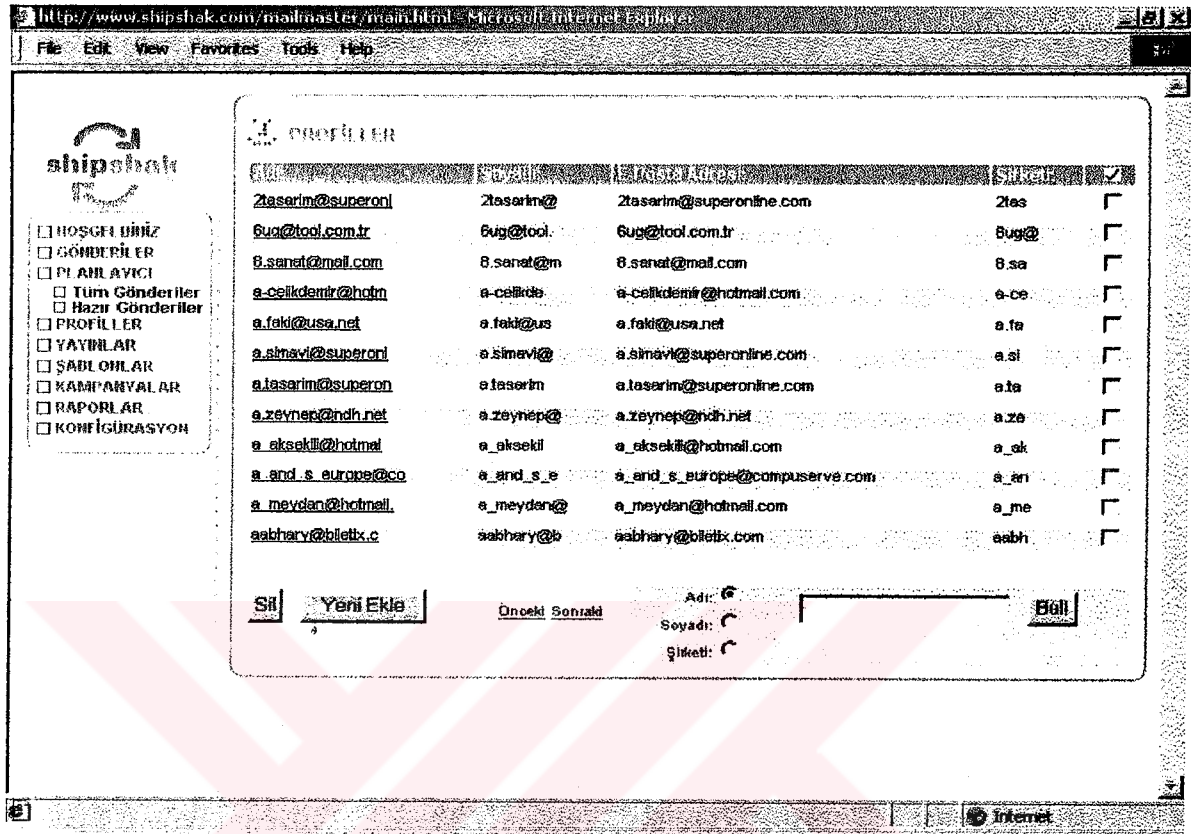
Görev zamanlayıcı ile bitirilmiş ve hazır durumda olan işler takip edilir:



Şekil.4.7. Görev Zamanlayıcı

4.1.4. Profiller Bileşeni

Profiller bileşeni ile yeni alıcıların sisteme eklenmesi veya var olan alıcıların bilgilerinde değişiklik yapılması , sisteme yeni alıcılar eklenmesi veya varolan alıcıların silinmesi gibi işlemlerin yapılmasını sağlar. Ayrıca , herhangi bir alıcıya kolayca ulaşmayı sağlayacak ad, soyad veya şirket adına göre arama yapılabilen bir menüde mevcuttur.



Şekil.4.8. Profil Bileşeni Ara Yüzü

4.1.5. Yayınlar bileşeni

Sistem , belirli aktivitelerin önceden tanımlanmasına olanak sağlar. Şekil.4.9.'da doğum günü yayını bir örnek olarak verilmiştir.

http://www.shipshak.com/mailmaster/main.html - Microsoft Internet Explorer

File Edit View Favorites Tools Help

shipshak

- ☐ HOŞGELDİNİZ
- ☐ GÖRÜŞLER
- ☐ PLAN AYICI
 - ☐ Tüm Gönderiler
 - ☐ Hazır Gönderiler
- ☐ PROFİLER
- ☐ YAYINLAR
- ☐ ŞABLONLAR
- ☐ KAMPANYALAR
- ☐ RAPORLAR
- ☐ KONFIGURASYON

YENİ YAYIN

Doğum Günü

Doğum Günü

Sn. [[name]]

Doğum Gününüz kutlu olsun.

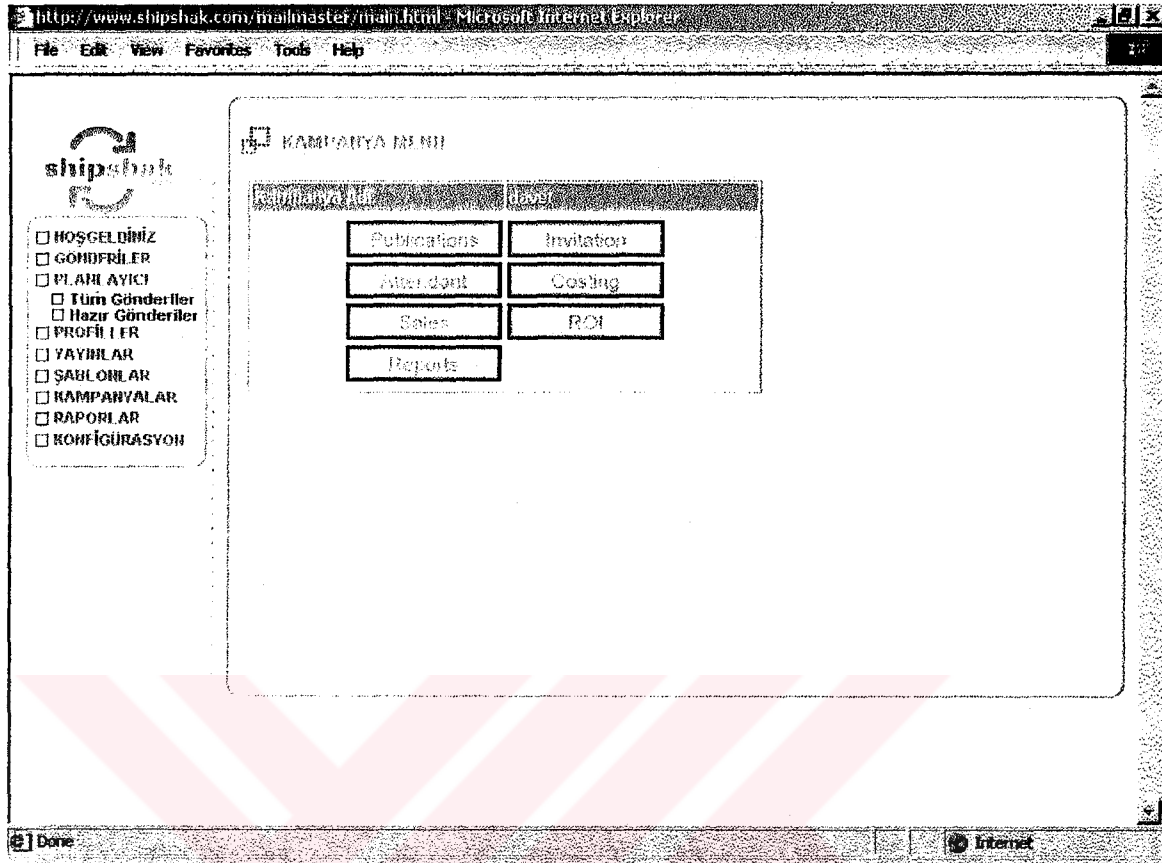
name

Done Internet

Şekil.4.9. Yayınlar Bileşeni

4.1.6. Kampanyalar Bileşeni

Sistem üzerinde birden çok aktiviteyi bir araya getirerek kampanyalar yaratmak mümkündür.

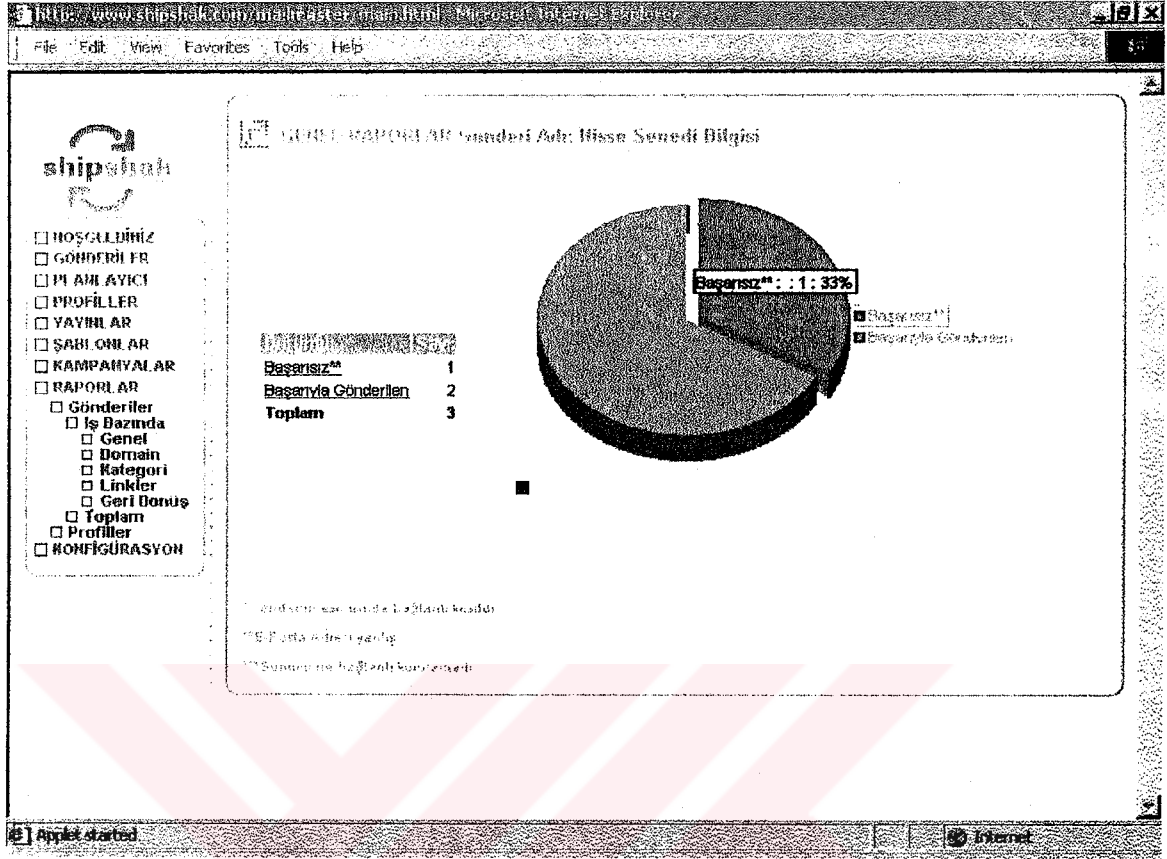


Şekil.4.10. Kampanyalar

Kampanyaların amacı , aktiviteler zinciri hazırlayarak , bu aktiviteleri arka arkaya çalıştırabilmektir. Bu sayede örnek olarak bir davet düzenlenebilir, bu davete katılması muhtemelen kişilere e-posta gönderilir, daha sonra davete geleceğini bildiren kişilere detaylar gönderilir vb. Ayrıca kampanyayla ilgili olarak geri dönüş bilgilerini de almak mümkündür.

4.1.7. Raporlar Bileşeni

Gönderilen iletilerin durumu hakkında grafiksel bilgiler şekil.4.11’de görüldüğü gibi alınabilir.



Şekil.4.11. Raporlar

Raporlar sayesinde , hazırlanan işlerin bitip bitmediğini ,işler içerisinde gerçekleştirilen gönderilerin durumlarını , başarısız iletilerin başarısız olma sebeplerini , zaman olarak iletilerin gönderilme sürelerini, domainlere göre iletilerin gönderilme zamanlarını vb. bir çok değişik bilgiye ulaşmak mümkündür. Ayrıca , sistem bir iş bitirildiğinde o işle ilgili detaylı bir raporu işi hazırlayan kişiye e-posta olarak göndermektedir.

4.1.8. Konfigürasyon Bileşeni

Konfigürasyon bileşeni ile sistemin kullanıcı tarafından konfigüre edilmesi sağlanmaktadır. Böylece kullanıcı kendisine ait çalışma ortamını yaratabilir.



4.1.8.1. Alıcı Alanları Bileşeni

Alıcıların var olan nitelikleri içerisinde seçim yaparak sadece istenilen alanlarla çalışmak mümkündür:

http://www.shipshak.com/mailmaster/main.html - Microsoft Internet Explorer

File Edit View Favorites Tools Help

shipshak

☐ HOŞGELDİNİZ
☐ GÖNDERİLER
☐ PLANLAVICI
☐ PROFİLLER
☐ YAYINLAR
☐ ŞABLONLAR
☐ KAMPANYALAR
☐ RAPORLAR
☐ KONFIGÜRASYON
☐ Kategoriler
☐ E-Posta Ayarları
☐ Kullanıcılar
☐ Etkin Alanlar
☐ Dosyadan Yükle
☐ Dinamik Alanlar

Etkin Alanlar

E-Posta Adres: (2) ☒ Web Adres: ☒
Ünvan: ☒ İQ Numarası: ☒
Cinsiyet: ☒ Cep Telefonu: ☒
Şirket: ☒ Doğum Tarihi: ☒
Adres: ☒ Adres: (2) ☒
Şehir: ☒ Şehir Kodu: ☒
Posta Kodu: ☒ Ülke: ☒
Ev Telefonu: ☒ İş Telefonu: ☒
İş Telefonu: (2) ☒ Faks: ☒
Faks: (2) ☒ Üyelik Başlangıç: ☒
Üyelik Bitiş: ☒
domainid: ☐

Lütfen profiller için ilgilendiğiniz alanları seçiniz.
Seçili olmayan alanlar program tarafından kullanılmaz.

Kaydet

Done Internet

Şekil.4.12. Kullanıcı Alanları Arayüzü

Burada görülen alanlar , aslında sistemde kullanılacak olan alıcı bilgileridir.Eğer istenirse burada görülen alanların hepsi veya bir kısmı sisteme dahil edilerek , kullanılmayan alanlarla işlem yapılması engellenmiş olur.

4.1.8.2. Kategori Bileşeni

Sistem istenilen kategorinin eklenip çıkartılması olanağını sağlamaktadır:

Şekil.4.13. Kategori Bileşeni

Şekilde de görüldüğü gibi, 2 seviyeli olmak üzere istenildiği kadar kategori yaratılması mümkündür.

4.1.8.3. Dinamik Alanlar Bileşeni

Sistem istenirse dinamik ileti hazırlamak amacıyla kullanılan yeni dinamik alanların tanıtılmasına olanak tanır.

http://www.shipshak.com/mailmaster/main.html - Microsoft Internet Explorer

File Edit View Favorites Tools Help

shipshak

- ☐ HOŞGELDİNİZ
- ☐ GÖNDERİLER
- ☐ PLANLAYICI
- ☐ PROFİLLER
- ☐ YAYINLAR
- ☐ ŞABLONLAR
- ☐ KAMPANYALAR
- ☐ RAPORLAR
- ☐ KONFIGÜRASYON
 - ☐ Kategoriler
 - ☐ E-Posta Ayarları
 - ☐ Kullanıcılar
 - ☐ Etkin Alanlar
 - ☐ Dosyadan Yükle
 - ☐ Dinamik Alanlar

YENİ DİNAMİK ALAN

Ad	soyad
Alan	musteri.soyad
Kolon	Kolon
Fonksiyon	Fonksiyon

Kaydet

Done Internet

Şekil.4.14. Dinamik Alanlar

Böylece alıcıların e-posta mesajlarında alıcılara ait alanlar istenildiği gibi özelleştirilmiş olur.

Bu çalışmada geliştirilen e-posta ile pazarlama yazılımı şu anda deneme amaçlı kullanılmaktadır:

Biletix-İstanbul

Buradaki kullanımda müşteriler yeni etkinliklerden haberdar edilmekte , müşterilerin etkinliklerle ilgili sayfaları ziyaret ediş zamanları ve sayıları görülebilmektedir.

Ericcson –İstanbul , Sun.-İstanbul

Her iki firmada yazılım yüksek hacimli gönderilerle test edilmektedir.

Meteoroloji Merkezi – Ankara

Buradaki kullanımda , her gün değişik zamanlarda alınan hava durumu tahminleri değişik merkezlere gönderilmektedir. Daha önceden böyle bir sistemi olmadığı belirlenen merkezin bu işi yapmak için 70 kişi görevlendirdiği göz önüne alınırsa , sistemin avantajları kaçınılmaz olarak ortaya çıkar.

Bu ana kadar yapılan çalışmalarda ortaya çıkan sonuçlar şöyle sıralanabilir:

- e-posta ile pazarlama ülkemizde yeni bir konudur.Bir çok orta ölçekli firma pazarlamayla ilgili olarak yaptığı yatırımlara e-posta ile pazarlama konusunu almayı ilerisi için düşünmekte olduklarını söylerken , büyük firmalar bu konuyla ilgili yatırım yapma kararı almış durumdadırlar.
- e- posta ile pazarlama ile e-ticaret i bir araya getiren bu yazılım çözümü,kendi rakiplerine göre konuya getirdiği bilimsel perspektif açısından oldukça ileri bir durumdadır.
- Her ne kadar e-posta ile pazarlama konusu altında olsa da , yazılımın sms,fax,icq gibi e-posta dışında ki ortamlara da ileti yapabilmesi , bu alandaki ihtiyacı karşılayacak niteliktedir. Bir çok firma , kendi bünyelerinde kullandıkları fax sunucusu,posta sunucusu programlarıyla benzer işlemleri yapmaya

çalışmaktadırlar .Bu yazılım , tam bir platform sağlaması özelliğiyle avantaja sahiptir.

- Yazılımın CRM paketleriyle beraber bir ek modül olarak kullanılabilirliği ihtimali , ayrı bir entegrasyon kapısını aralamaktadır.
- Bilimsel olarak e- posta ile pazarlama , araştırmaya açık bir alandır.Dolayısıyla , bu çalışmanın uygulamalarından ortaya çıkan en önemli sonuç , konu üzerinde daha yapılabilecek bir çok şeyin olduğu gerçeğidir.Bunlar arasında örnek olarak,
- cep telefonu kullanıcılarının sinyal aldıkları merkezleri kontrol ederek,oldukları yeri saptama ve bulundukları yere giren ilgili pazarlama mesajını kişiye sms olarak gönderme ve
- video konferans olanağı sağlayabilen cihazlara görüntü gönderebilme
- yeni e-posta formatları geliştirme.

sayılabilir.

Yazılım çözümü , bunlar gibi tahmin edilebilen veya bunların dışında tahmin edilemeyen durumları destekleyebilecek şekilde esnek bir yapıda tasarlanmıştır.

5. SONUÇLAR VE ÖNERİLER

5.1. Sonuçlar

Bu tez çalışmasında, e-posta ile pazarlama yapmak amacıyla bir yazılım geliştirilmiştir. Bilgi toplumunun kendisine ait kuralları vardır. E-posta ile pazarlama, iş alanlarına müşterilerinin ne düşündüğü ve ne istediklerini anlama olanağını tanır. Diğer iş alanları gibi, internet üzerinde pazarlama yapmanın da normal pazarlama tekniklerinden farkı vardır. İnternet üzerinde pazarlamanın hızlı sonuca daha az masrafla ve daha doğru olarak ulaşılmasını sağladığını söylemek yanlış olmaz. Şirketler, bu bilgilerle geleceğe yönelik kararlarını daha sağlıklı şekilde alabilirler.

Yazılım geliştirilirken tekrar kullanılabilir kodlar yazılmasına özen gösterilmiştir. Bu amaçla “design pattern” lerden yararlanılmıştır. Böylece ileride ki isteklere yazılımın cevap verebilmesi amaçlanmıştır.

Sonuç olarak , yazılım java programlama diliyle kodlanmış bir object-oriented sistemdir. Bu yazılımın e-posta ile pazarlama konusunun problemlerine çözüm olacağı düşünülmektedir

5.2. Öneriler

E-posta ile pazarlama konusu üzerine yapılan bu tez çalışmasında , teknolojik alt yapının mümkün olduğunca esnek olmasına önem verilmiştir. Bu yüzden sistem , yazılım teknolojisinin bütün olanaklarını kullanmayı amaç edinmiş , oldukça esnek bir yapıya sahiptir. İleti hacmi düşük olan şirketler için ASP modeli uygun görülmekte , yüksek hacimli firmalar içinse şirket bünyesine dahil ederek kullanılması önerilir.

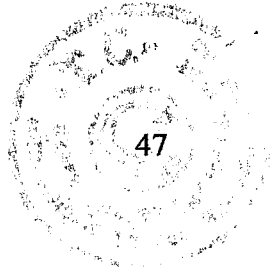
KAYNAKLAR

- [1].Daniel Amor,"The E-business revolution-2000",Prentice Hall
2000,www.ebusinessrevolution.com
- [2].Jim Hu,"When is email marketing spam?", 31/08/1998,www.cnet.com
- [3].The eMarketing Digest, 2001,www.webbers.com
- [4].Booch,G., "Object Solutions:Managing the Object-oriented project", 1996
- [5].Craig Larman,"Applying Uml and patterns", 1998,Prentice Hall PTR,2000,
www.applyumlpatterns.com
- [6].John William Toigo,"the Holly Grail of Data Storage Management", Prentice Hall
PTR,1999,www.phptr.com
- [7]Microsoft\Microsoft Commerce Solutions Web Technology ,
1999,Mspress.microsoft.com
- [8]Malone T.,Yates J. and Benjamin R., "Electronic markets and electronic hierarchies.
Commun. ACM",30/06/1987
- [9]Rayport J.F. and Sviokla, J.J,"Managing in the marketspace.", Harvard Business Rev.
72, 6 Nov.-Dec.1994), 141-150.
- [10]Stern LW, El-Ansary AI, Coughlan AT," Marketing Channels", Prentice
Hall, NJ 07458,1996:1±2.
- [11]Internet Economy Indicators, Eylül 22, 1999,[http://www.inter.netindicators.com/
features.html](http://www.inter.netindicators.com/features.html)
- [12]Chandrasekar Subramaniam,Michael J.Shaw and David M.Gardner," Product
Marketing and Channel Management in Electronic Commerce", Information System
Frontiers 1:4,363-378 , 2000
- [13] The American Marketing Association , www.ama.com

ÖZGEÇMİŞ

1974 yılında Tekirdağ'da doğdu.1990 yılında girdiği Yıldız Teknik Üniversitesi Bilgisayar Bilimleri Mühendisliği bölümünden 1997 yılında mezun oldu.1999 yılında Gebze Yüksek Teknoloji Enstitüsü Bilgisayar Mühendisliği Bölümünde yüksek lisans eğitimine başladı.1996 yılında girdiği Login Bilgisayar A.Ş. de halen Informix veritabanı uzmanı olarak görev yapmakta,veritabanı projelerinde danışman olarak bulunmakta ve yazılım projelerinde yazılım mühendisi olarak görev almaktadır. E-iş , e-ticaret ilgilendiği temel alanlar'olup , bu konularda çalışmalarına devam etmektedir.





EKLER

Ek 1: mailAgent.java kaynak kodu

Ek 2: jobFactory.java kaynak kodu

EK 3 : staticJobFactory.java kaynak kodu

EK 4 : dynamicJobFactory.java kaynak kodu

EK 5 : smsJobFactory.java kaynak kodu

EK 6: mailWorks.java kaynak kodu



EK 1 : mailAgent.java

```
/*  
    This class is responsible for sending mails, in other words  
    processing the Jobs.  
    At specified intervals , a thread is forked and it checks there exists  
    ready jobs or not.If so it"runs the jobs" by sending mails to appropriate  
    destinations.  
*/  
  
import java.util.*;  
import java.net.*;  
import javax.mail.*;  
import javax.mail.internet.*;  
import java.sql.*;  
  
public class mailAgent  
{  
    public static void main(String args[])  
    {  
        while (true)  
        {  
            try  
            {  
                (new mailBrowser()).start();  
                Thread.sleep(60000*5);  
            }  
            catch (InterruptedException e)  
            {  
                e.printStackTrace();  
            }  
        }  
    }  
}
```



/*

This thread is responsible for

1. checking the ready jobs
2. sending ready jobs to the destination

*/

class mailBrowser extends Thread

{

private Hashtable data;

public mailBrowserVer2 () { data = new Hashtable(); }

public int getData()

// return 1 if there is ready job, 0 if an error occurs, -1 if there is no

// ready

{

mailOperations mo = new mailOperations(); // mail send class
return mo.findReadyJobs(data);

}

// create a thread for each job.

public void run()

{

System.out.println("a new mailbrowser was started");

int result = getData();

if (result == -1)

System.out.println("no ready jobs:");

else if (result == 0)

System.out.println("error");

else if (result == 1)

{

Enumeration e = data.elements();

while (e.hasMoreElements())

{

job jb=(job)e.nextElement();

jobThread jTh = new jobThread(jb);

jTh.start();

}

}

System.out.println("mailbrowser was finished");

}

```
}

```

```
class jobThread extends Thread

```

```
{

```

```
    job jb;

```

```
    public jobThread(job pJob)

```

```
    {

```

```
        jb = pJob;

```

```
    }

```

```
    public void processJobs(jobFactory jF)

```

```
    {

```

```
        // run factory methods and do whatever.

```

```
        //write the story here

```

```
        int mailSuccess = 0;

```

```
        int success = 1;

```

```
        int count=0;

```

```
        int total = 0;

```

```
        int size=jb.toList.size();

```

```
        mailOperations mo = new mailOperations();

```

```
        mo.fillJobDetailWithCustomers(jb);

```

```
        Vector customerResults = new Vector();

```

```
        mailWorks mW=new mailWorks("www.login.com.tr",1);

```

```
        mW.setTransport();

```

```
        Connection conn=mo.getConnection();

```

```
        PreparedStatement pstmt=mo.prepUpdateJobs(conn);

```

```
        PreparedStatement pstmtCustomerState = mo.prepUpdateCustomer(conn);

```

```
        Enumeration e1=jb.toList.elements();

```

```
        while (e1.hasMoreElements()) // for all addresses

```

```
        {

```

```
            mailAdr mA = (mailAdr)e1.nextElement();

```

```
            String s1=mA.getAddr();

```

```
            int per_id = mA.getId();

```

```
            int detail_id = mA.getDetailId();

```

```
            int customerSuccess = 0;

```

```
            try

```

```
            {

```

```
                mW.setAddress(count,new InternetAddress(s1));

```

```
            }

```

```
            catch(AddressException ae)

```

```
            {

```

```

        System.out.println("ex:"+ae.getMessage());
    }
    mailSuccess=mW.sendMail("e-marketing",per_id,conn,jF);
    if (mailSuccess!=1)
    {
        if (mailSuccess == -1)
        {
            customerSuccess = 1;
        }

        else if (mailSuccess == -2)
        {
            customerSuccess = 2;
        }
    }
    else customerSuccess = 3;
    success = 0;
    }
    else
    {
        customerSuccess = 4;
        total++;
    }
    customerResults.addElement(new detail_Status(detail_id,customerSuccess));
} // end of while for all mail addresses
if (pstmt!=null)
{
    int tot = customerResults.size();
    if (total==tot) success = 0; // success
    else if (total==0) success = 0; // 3 - unsuccessful completely
    else success = 0; // 2 partially completed
    //1. change individual job_details:
    if ((mo.changeJobDetailState(customerResults,conn,pstmtCustomerState))==0)
    System.out.println("some customer mail statuis cant be changed!");
    //mail sent returns with success , so this jobs state must be changed //to done state
    if (mo.changeJobState(jb.getId(),success,conn,pstmt)==0)
        System.out.println("Job status "+jb.getId() +" cant be changed!");
    }
    else
        System.out.println("db connection has a problem");

    // release the db connection
    mo.releaseConnection(conn,pstmt);

    System.out.println("EXIT job");
}

public void run()
{

```

```
String job_type = jb.getJtype();
if ("S".equalsIgnoreCase(job_type))
{
    //static job
    staticjobFactory sjFactory = new staticjobFactory(jb);
    processJobs(sjFactory);
}
else if ("D".equalsIgnoreCase(job_type))
{
    //dynamic job
    dynamicjobFactory djFactory = new dynamicjobFactory(jb);
    processJobs(djFactory);
}
else if ("F".equalsIgnoreCase(job_type))
{
    //forward mail dynamic job
    forwardjobFactory fjFactory = new forwardjobFactory(jb);
    processJobs(fjFactory);
}
else if ("M".equalsIgnoreCase(job_type))
{
    //sms job
    smsjobFactory smsjFactory = new smsjobFactory(jb);
    processJobs(smsjFactory);
}
else if ("T".equalsIgnoreCase(job_type))
{
    //mhtml job
    mhtmljobFactory mhtmljFactory = new mhtmljobFactory(jb);
    processJobs(mhtmljFactory);
}
else
{
    // just a job
    //jobFactory jFactory = new jobFactory(jb);
    //processJobs(jFactory);
    System.out.println("unknown job");
}
}

}
```

EK 2 : jobFactory.java

```
/*  
    This is an implementation of Abstract Factory design pattern  
*/  
import java.sql.*;  
import javax.mail.internet.*;  
public abstract class jobFactory  
{  
    //constructor  
  
    // factory methods .create some products  
  
    abstract MimeMultipart createMultiPart(int person_id,Connection conn);  
    abstract abstractSubject createSubject(int person_id,Connection conn);  
  
}
```

EK 3 : staticJobFactory.java

```
import java.sql.*;
import java.util.*;
import javax.mail.*;
import javax.mail.event.*;
import javax.mail.internet.*;
import javax.activation.*;
import java.io.*;

public class staticJobFactory extends JobFactory
{
    Job jb;
    public staticJobFactory(Job pjb)
    {
        System.out.println("static job");
        jb = pjb;
    }

    public AbstractSubject createSubject(int person_id, Connection conn)
    {
        return new staticSubject(jb.getSubject());
    }

    public MimeMultipart createMultiPart(int person_id, Connection conn)
    {
        MimeMultipart mp = new MimeMultipart();
        try
        {
            MimeBodyPart b1 = new MimeBodyPart();
            b1.setContent(jb.getBody(), "text/plain");
            b1.setDisposition(b1.INLINE);
            mp.addBodyPart(b1);

            Enumeration e2 = jb.getAttachmentsVer2();
            while (e2.hasMoreElements())
            {
                MimeBodyPart b2 = new MimeBodyPart();
                ByteArrayDataSource bbas = null;
                AttachmentVer2 atv = (AttachmentVer2) e2.nextElement();
```

```
String fileName = atv.getFileName();
String ctype = atv.getCtype();
ByteArrayInputStream byteValue=null;
byteValue= (ByteArrayInputStream)atv.getByteValue();
bbas =new ByteArrayDataSource(byteValue,ctype );

DataHandler dh = new DataHandler(bbas);
b2.setDataHandler(dh);
b2.setFileName(fileName);
b2.setDisposition(b2.INLINE);

mp.addBodyPart(b2);
byteValue.reset();
}
}

catch (MessagingException msge)
{
    System.out.println("error in staticjobfactory(createmultipart):"+msge.getMessage());
}
System.out.println("static job is ready to go ");
return mp;
}
}
```

EK 4 : dynamicJobFactory.java

```
import java.sql.*;
import java.util.*;
import javax.mail.*;
import javax.mail.event.*;
import javax.mail.internet.*;
import javax.activation.*;
import java.io.*;

public class dynamicJobFactory extends jobFactory
{
    job jb;
    public dynamicJobFactory(job pjb)
    {
        System.out.println("dynamic job");
        jb = pjb;
    }

    public MimeMultipart createMultiPart(int person_id, Connection conn)
    {
        String body = jb.getBody();
        body = myutility.fillDynamicFields(body, conn, person_id);
        MimeMultipart mp = new MimeMultipart();
        System.out.println("body:" + body);
        try
        {
            MimeBodyPart b1 = new MimeBodyPart();
            b1.setContent(body, "text/plain");
            b1.setDisposition(b1.INLINE);
            mp.addBodyPart(b1);

            Enumeration e2 = jb.getAttachmentsVer2();
            while (e2.hasMoreElements())
            {
                MimeBodyPart b2 = new MimeBodyPart();
                ByteArrayDataSource bbas = null;
                AttachmentVer2 atv = (AttachmentVer2) e2.nextElement();
                String fileName = atv.getFileName();
                String ctype = atv.getCtype();
                ByteArrayInputStream byteValue = null;
                byteValue = (ByteArrayInputStream) atv.getByteValue();
```

```

        "text/html".equalsIgnoreCase(cttype))

        if ("text/plain".equalsIgnoreCase(cttype)) {

            //if attachment is editable do necessary process.
            String textValue = myutility.getDBByte(byteValue);

            textValue = myutility.fillDynamicFields(textValue, conn, person_id);

            bbas = new ByteArrayDataSource(textValue, cttype);

        }

        else

            bbas = new ByteArrayDataSource(byteValue, cttype);

        DataHandler dh = new DataHandler(bbas);
        b2.setDataHandler(dh);
        b2.setFileName(fileName);
        b2.setDisposition(b2.INLINE);

        mp.addBodyPart(b2);
        byteValue.reset();

    }

    catch (MessagingException msge)
    {

        System.out.println("error in staticjobfactory(createmultipart):" + msge.getMessage());

    }

    System.out.println("dynamic job is ready to go ");
    return mp;

}

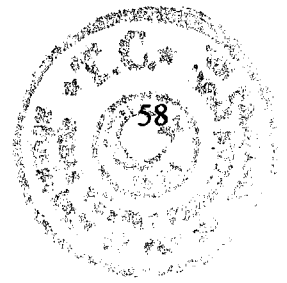
public abstractSubject createSubject(int person_id, Connection conn)
{

    return new dynamicSubject(jb.getSubject(), person_id, conn);

}

}

```

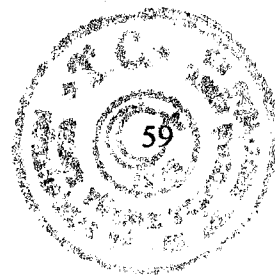


EK5 : smsJobFactory.java

```
import java.sql.*;
import java.util.*;
import javax.mail.*;
import javax.mail.event.*;
import javax.mail.internet.*;
import javax.activation.*;
import java.io.*;

public class smsJobFactory extends JobFactory
{
    Job jb;
    public smsJobFactory(Job pjb)
    {
        System.out.println("sms job");
        jb = pjb;
    }
    public MimeMultipart createMultipart(int person_id, Connection conn)
    {
        String body = jb.getBody();
        body = myutility.fillDynamicFields(body, conn, person_id);
        MimeMultipart mp = new MimeMultipart();
        System.out.println("body:" + body);
        try
        {
            MimeBodyPart b1 = new MimeBodyPart();
            b1.setContent(body, "text/plain");
            b1.setDisposition(b1.INLINE);
            mp.addBodyPart(b1);

            Enumeration e2 = jb.getAttachmentsVer2();
            while (e2.hasMoreElements())
            {
                MimeBodyPart b2 = new MimeBodyPart();
                ByteArrayDataSource bbas = null;
                AttachmentVer2 atv = (AttachmentVer2) e2.nextElement();
                String fileName = atv.getFileName();
                String ctype = atv.getCtype();
                ByteArrayInputStream byteValue = null;
                byteValue = (ByteArrayInputStream) atv.getByteValue();
                bbas = new ByteArrayDataSource(byteValue, ctype);
                DataHandler dh = new DataHandler(bbas);
```



```
b2.setDataHandler(dh);  
b2.setFileName(fileName);  
b2.setDisposition(b2.INLINE);  
mp.addBodyPart(b2);  
byteValue.reset();
```

```
}
```

```
}
```

```
catch (MessagingException msge)
```

```
{
```

```
    System.out.println("error in staticjobfactory(createmultipart):"+msge.getMessage());
```

```
}
```

```
System.out.println("sms job is processed ");
```

```
return mp;
```

```
}
```

```
public abstractSubject createSubject(int person_id,Connection conn)
```

```
{
```

```
    return new dynamicSubject(jb.getSubject(),person_id,conn);
```

```
}
```

```
}
```



EK 6: mailWorks.java

```
/*
    This class is responsible for sending pop3 mails.
*/
import java.util.*;
import javax.mail.*;
import javax.mail.event.*;
import javax.mail.internet.*;
import javax.activation.*;
import java.io.*;
import java.lang.reflect.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class mailWorks
{
    // send mail variables

    private String smtpHost;                // mail server host name in dot format
                                           // such as www.xxx.yyy
    private Address[] toAddress;            // destination addresses
                                           // for sending mail

    // read mail variables

    private Session session;                // session for reading mail
    private Store store;                    // store for reading mail
    private Folder folder;                  // folder for reading mail
    private Transport tr;
    private Message[] messages=null;        // mail messages
    private int numOfMessages;              // number of messages
    private int numOfOldMessages;            // number of new mails
    private MimeBodyPart[] bodypart;        // mail attachments
    private int numOfBodyPart;              // number of attachments
    private String mailData;                // mail body text data

    /*
    Creates an empty mailWorks instance
    */
    public mailWorks()
    {
```



```
store = null;
folder = null;
}
/*
Creates mailWorks object with smtpHost and number of to address.
It is used for sending mail
*/
public mailWorks(String pHost,int size)
{
    this.smtpHost=pHost;
    this.toAddress = new InternetAddress[size];
}

// get & set methods for variables

public Message[]    getMessages()                { return this.messages;
}
public Message      getMessage(int num) { return this.messages[num];    }
public int          getMessageCount()  { return this.numOfMessages;    }
public int          getOldMessageCount() { return this.numOfOldMessages; }

public String getSubject(Message msg) throws MessagingException
{
    return msg.getSubject();
}

public String getFrom(Message msg)                throws MessagingException
{
    String retVal = new String();
    try
    {
        retVal = msg.getFrom()[0].toString();
    }
    catch (NullPointerException ne)
    {
        System.out.println("hata:"+ne.getMessage());
        retVal = "";
    }
    return retVal;
}
}
/*
```

This method gets the to the addresses and store them in addr structure.

This structure holds the to addresses in string array and store the number of addresses.

```

    */
    public addr getTo(Message msg) throws MessagingException
    {
        InetAddress[] iadd = (InetAddress[])msg.getRecipients(Message.RecipientType.TO);
        int length = 0;
        // number of to addresses
        if (iadd != null)
            length = java.lang.reflect.Array.getLength(iadd);

        addr to = new addr(length);

        // create a new addr structure

        // to store these addresses
        int next = 0;

        // the number of to addresses

        int i = 0;

        // after eliminating duplicates
        for (i=0;i<length;i++)
        {
            if ((to.setAddress(next,iadd[i].getAddress()))==1) // get the address
            {
                // match found
            }
            else
            {
                // no match
                next++;
            }
        }
        if (next<i)
            to.setLength(next);

        return to;
    }

    /*
    This method gets the CC the addresses and store them in addr structure.
    This structure holds the to addresses in string array and store the
    number of addresses.
    */
    public addr getCC(Message msg) throws MessagingException
    {
        InetAddress[] iadd = (InetAddress[])msg.getRecipients(Message.RecipientType.CC);

```

```

        int length = 0;
addresses
        if (iadd != null)
            length = java.lang.reflect.Array.getLength(iadd);
        addr to = new addr(length);
// create a new addr structure

// to store these addresses

int next = 0;
addresses
int i = 0;
duplicates

        for (i=0;i<length;i++)
        {
            if ((to.setAddress(next,iadd[i].getAddress()))==1)
                { // match found
                    // get the address
                }
            else
            {
                // no match
                next++;
            }
        }
        if (next<i)
            to.setLength(next);

        return to;
    }

    /*
    This method gets the bcc the addresses and store them in addr structure.
    This structure holds the to addresses in string array and store the
    number of addresses.
    */
    public addr getBCC(Message msg) throws MessagingException
    {

        InetAddress[] iadd = (InetAddress[]) msg.getRecipients(Message.RecipientType.BCC);
        int length = 0;
        if (iadd!=null)
            length = java.lang.reflect.Array.getLength(iadd);
// number of to addresses

        addr to = new addr(length);
// create a new addr structure

        // to store these addresses

```

```

        int next = 0;
        // the number of to addresses
        int i = 0;
        // after eliminating duplicates

        for (i=0;i<length;i++)
        {
            if ((to.setAddress(next,iadd[i].getAddress()))==1) // get the address
            {
                // match found
            }
            else
            {
                // no match
                next++;
            }
        }
        if (next<i)
            to.setLength(next);

        return to;
    }

    public int getMessageNumber(Message msg)
    {
        return msg.getMessageNumber();
    }

    public java.util.Date getSentDate(Message msg) throws MessagingException
    {
        return msg.getSentDate();
    }

    public void setDeletedFlags(int msgnum) throws MessagingException
    {
        folder.setFlags(msgnum,msgnum,new Flags(Flags.Flag.DELETED),true);
    }

    public java.util.Date getReceivedDate(Message msg) throws MessagingException
    {
        return msg.getReceivedDate();
    }

    public String getBodyPartContentType(int num) throws MessagingException
    {

```

```

        return bodypart[num].getContentType();
    }

    public InputStream getBodyPartInputStream(int num) throws MessagingException,IOException
    {
        return bodypart[num].getInputStream();
    }

    public String getBodyPartFileName(int num) throws MessagingException
    {
        return bodypart[num].getFileName();
    }

    public int getBodyPartSize(int num) throws MessagingException
    {
        return bodypart[num].getSize();
    }

    public void getHeaders(int num) throws MessagingException
    {
        Enumeration e=bodypart[num].getAllHeaders();
        while (e.hasMoreElements())
        {
            Header h = (Header)e.nextElement();
            System.out.println("name:"+h.getName()+"val:"+h.getValue());
        }
    }

    public int getNumOfBodyPart() { return this.numOfBodyPart; }

    public String getMailData()
    {
        return this.mailData;
    }

    public void setAddress(int count,InternetAddress addr1)
    {
        this.toAddress[count]=addr1;
    }

    /*
    This method send mail to the appropriate destination.
    Return values: 1 means success (this doesnt mean that the message reaches the destination
    0 failure

```

```
*/  
public int sendMail(String pFrom,String pData,String pSubject)  
{  
  
    try  
    {  
        java.util.Properties properties=System.getProperties();  
        properties.put("mail.smtp.host",smtpHost);  
        Session ses = Session.getInstance(properties,null);  
        MimeMessage message=new MimeMessage(ses);  
        Address fromAddress=new InternetAddress(pFrom);  
        message.setFrom(fromAddress);  
  
        message.setRecipients(Message.RecipientType.TO,toAddress);  
message.setSubject(pSubject);  
  
        // add attachements:  
  
        MimeMultipart mp = new MimeMultipart();  
  
        MimeBodyPart b1 = new MimeBodyPart();  
        b1.setContent(pData,"text/plain");  
        mp.addBodyPart(b1);  
  
        /*Enumeration e1 = jb.getAttachments();  
        while (e1.hasMoreElements())  
        {  
            String fileName=(String)e1.nextElement();  
            MimeBodyPart b2 = new MimeBodyPart();  
            FileDataSource fds = new FileDataSource(fileName);  
            b2.setDataHandler(new DataHandler(fds));  
            b2.setFileName(fds.getName());  
            mp.addBodyPart(b2);  
        }*/  
  
        message.setContent(mp);  
  
        Transport.send(message);  
        System.out.println("mail gonderildi:");  
    }  
    catch(AddressException ae)
```



```
{
    System.out.println("h1:"+ac.getMessage());
    return 0;
}
catch(SendFailedException se)
{
    System.out.println("h2:"+se.getMessage());
    return 0;
}
catch(MessagingException me)
{
    System.out.println("h3:"+me.getMessage());
    return 0;
}
return 1;
}
```

```
public int sendMail(mail m,int mailType)
```

```
/*
```

```
This method send mail to the appropriate destination.
```

```
Return values: 1 means success (this doesnt mean that the message reaches the destination
```

```
0 failure
```

```
*/
```

```
{
```

```
System.out.println("smtp host:"+smtpHost);
```

```
try
```

```
{
```

```
java.util.Properties properties=System.getProperties();
```

```
properties.put("mail.smtp.host",smtpHost);
```

```
Session ses = Session.getInstance(properties,null);
```

```
MimeMessage message=new MimeMessage(ses);
```

```
Address fromAddress=new InternetAddress(m.getFrom());
```

```
message.setFrom(fromAddress);
```

```
InternetAddress[] toAdr = new InternetAddress[m.getNumTo()];
```

```
//System.out.println("numto sayisi:"+m.getNumTo());
```

```
for (int i=0;i<m.getNumTo();i++)
```

```
{
```

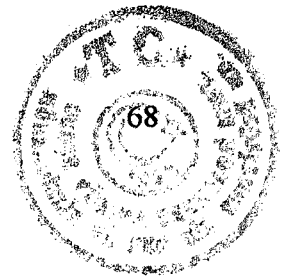
```
toAdr[i] = new InternetAddress(m.getToAddr(i));
```

```
}
```

```
message.setRecipients(Message.RecipientType.TO,toAdr);
```

```
InternetAddress[] ccAdr = new InternetAddress[m.getNumCc()];
```

```
for (int i=0;i<m.getNumCc();i++)
```



```
{

    ccAdr[i] = new InternetAddress(m.getCcAddr(i));
}
message.setRecipients(Message.RecipientType.CC,ccAdr);

InternetAddress[] bccAdr = new InternetAddress[m.getNumBcc()];
for (int i=0;i<m.getNumBcc();i++)
{

    bccAdr[i] = new InternetAddress(m.getBccAddr(i));
}
message.setRecipients(Message.RecipientType.BCC,bccAdr);

message.setSubject(m.getSubject());

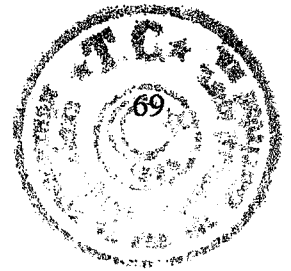
// add attachements:

MimeMultipart mp = new MimeMultipart();

MimeBodyPart b1 = new MimeBodyPart();
b1.setContent(m.getData(),"text/plain");
mp.addBodyPart(b1);

int numAttach = m.getNumAttach();
if ((mailType == 0) || (mailType == 1)) // compose a new mail or reply
{
    for (int j=0;j<numAttach;j++)
    {

        attachment a = m.getAttachment(j);
        String fileName = a.getName();
        fileName = "c:\\inetpub\\tmp\\"+fileName;
        MimeBodyPart b2 = new MimeBodyPart();
        FileDataSource fds = new FileDataSource(fileName);
        System.out.println("mail sendet type:"+fds.getContentType());
        b2.setDataHandler(new DataHandler(fds));
        //b2.setContent(fds,"image/gif");
        b2.setFileName(fds.getName());
        b2.setDisposition(b2.INLINE);
        mp.addBodyPart(b2);
    }
}
else
```



```
        if (mailType == 2) // forward
        {

        }

        message.setContent(mp);

        Transport.send(message);
        System.out.println("mail gonderildi:");

    }
    catch(AddressException ae)
    {
        System.out.println("h1:"+ae.getMessage());
        return 0;
    }
    catch(SendFailedException se)
    {
        System.out.println("h2:"+se.getMessage());
        return 0;
    }
    catch(MessagingException me)
    {
        System.out.println("h3:"+me.getMessage());
        return 0;
    }
    return 1;
}

public void setTransport ()
{
    java.util.Properties properties=System.getProperties();
    properties.put("mail.smtp.host",smtpHost);
    Session ses = Session.getInstance(properties,null);
    try
    {
        this.tr = ses.getTransport("smtp");
    }
    catch (NoSuchProviderException nos) { System.out.println("error:"+nos.getMessage());
    }

}

public Transport getTransport()
{
    return this.tr;
}

/*
```

This method send mail to the appropriate destination.

Return values: 1 means success (this doesnt mean that the message reaches the destination

0 failure

*/

```
public int sendMail(String pFrom,String pData,String pSubject,job jb,String job_type,int per_id,Connection
conn)
```

```
{
```

```
try
```

```
{
```

```
java.util.Properties properties=System.getProperties();
```

```
properties.put("mail.smtp.host",smtpHost);
```

```
Session ses = Session.getInstance(properties,null);
```

```
MimeMessage message=new MimeMessage(ses);
```

```
Address fromAddress=new InternetAddress(pFrom);
```

```
message.setFrom(fromAddress);
```

```
message.setRecipients(Message.RecipientType.TO,toAddress);
```

```
message.setSubject(pSubject);
```

```
MimeMultipart mp = null;
```

```
Enumeration e2 = jb.getAttachmentsVer2();
```

```
if ("F".equalsIgnoreCase(job_type))
```

```
{
```

```
// forward mail
```

```
mp = new MimeMultipart("alternative");
```

```
ByteArrayDataSource bbas =null;
```

```
AttachmentVer2 atv = (AttachmentVer2) e2.nextElement();
```

```
String fileName = atv.getFileName();
```

```
String ctype = atv.getCtype();
```

```
ByteArrayInputStream byteValue = null;
```

```
byteValue= (ByteArrayInputStream)atv.getByteValue();
```

```
String textValue=getDBByte(byteValue);
```

```
textValue=myutility.fillDynamicFields(textValue,conn,per_id) ;
```

```
//textValue=myutility.fillDynamicFieldsInForwardedContent(textValue,conn,per_id) ;
```

```
ByteArrayInputStream byteValueConverted = new
ByteArrayInputStream(textValue.getBytes());
```

```
byteValue.reset();
```

```
ByteArrayDataSource forwardSource = new
ByteArrayDataSource(byteValueConverted,ctype);
```

```
DataHandler forwardHandler = new DataHandler(forwardSource);
```

```

        BodyPart mbodypart = new MimeBodyPart();
        mbodypart.setDataHandler(forwardHandler);
        mp.addBodyPart(mbodypart);

    }

    else
    {

        mp = new MimeMultipart();

        MimeBodyPart b1 = new MimeBodyPart();

        b1.setContent(pData,"text/plain");
        b1.setDisposition(b1.INLINE);
        mp.addBodyPart(b1);

        while (e2.hasMoreElements())
        {

            MimeBodyPart b2 = new MimeBodyPart();
            ByteArrayDataSource bbas =null;
            AttachmentVer2 atv = (AttachmentVer2) e2.nextElement();
            String fileName = atv.getFileName();
            String ctype = atv.getCtype();
            ByteArrayInputStream byteValue=null;
            byteValue= (ByteArrayInputStream)atv.getByteValue();

            //System.out.println("ctype:"+ctype);

            if ("D".equalsIgnoreCase(job_type))
            {

                // if dynamic mail,process attachments contents.
                if ("text/plain".equalsIgnoreCase(ctype)) ||

                {

                    //if attachment is editable do necessary process.
                    String textValue =getDBByte(byteValue);

                    textValue=myutility.fillDynamicFields(textValue,conn,per_id);

                    bbas =new ByteArrayDataSource(textValue,ctype );

                }
                else

```

```

        {

            bbas =new ByteArrayDataSource(byteValue,cttype );

        }

    else
    {

        bbas =new ByteArrayDataSource(byteValue,cttype );

    }

    DataHandler dh = new DataHandler(bbas);
    b2.setDataHandler(dh);
    //b2.setHeader("Content-ID", "<1>");
    b2.setFileName(fileName);
    //b2.setDisposition(b2.INLINE);

    mp.addBodyPart(b2);
    byteValue.reset();
    }

    message.setContent(mp);

    tr.send(message);
    System.out.println("mail gonderildi:");

    return 1;
}
catch(AddressException ae)
{
    System.out.println("inside send mail h1:"+ae.getMessage());
    return 0;
}
catch(SendFailedException se)
{
    System.out.println("bi sey var:"+se.getMessage());

    //her seferinde 1 tek destination address oldugu icin
    //tek bir adresle ilgili bilgi soz konusu.Bu yuzden
    // her uc methoddan sadece 1 sinden data donmeli
    if (se.getValidUnsentAddresses() != null)
    {

```

```

        // adres dogru fakat mail gonderilemedi
        return -1;
    }
    else if (se.getInvalidAddresses() != null)
    {
        // adres yanlis
        return -2;
    }
    else return -3;

    /*System.out.println("inside send mail h2:"+se.getMessage());

    System.out.println("success:"+se.getValidSentAddresses());
    System.out.println("email valid but unsent:"+se.getValidUnsentAddresses());
    Address[] xx=se.getInvalidAddresses();
    System.out.println("invalid address:"+xx[0].toString());
    */
    //return 0;
}
catch(MessagingException me)
{
    System.out.println("inside send mail h3:"+me.getMessage());
    return 0;
}
catch (Exception ee) { System.out.println("error:"+ee.getMessage());
                        return 0; }
}

/*
This method sends static mail to the appropriate destination.
Return values: 1 means success (this doesnt mean that the message reaches the destination
               0 failure
*/
public int sendMail(String pFrom,int per_id,Connection conn,JobFactory jF)
{
    try
    {
        java.util.Properties properties=System.getProperties();
        properties.put("mail.smtp.host",smtpHost);
        Session ses = Session.getInstance(properties,null);

        MimeMessage message=new MimeMessage(ses);
        Address fromAddress=new InternetAddress(pFrom);

```

```
message.setFrom(fromAddress);
message.setRecipients(Message.RecipientType.TO,toAddress);

abstractSubject aSub = jF.createSubject(per_id,conn);
System.out.println("subject:"+aSub.getSubject());
message.setSubject(aSub.getSubject());

MimeMultipart mp = jF.createMultiPart(per_id,conn);
message.setContent(mp);
tr.send(message);
System.out.println("mail gonderildi: " );
return 1;
}
catch(AddressException ae)
{
    System.out.println("inside send mail h1:"+ae.getMessage());
    return 0;
}
catch(SendFailedException se)
{
    System.out.println("bi sey var:"+se.getMessage());

    //her seferinde 1 tek destination address oldugu icin
    //tek bir adresle ilgili bilgi soz konusu.Bu yuzden
    // her uc methoddan sadece 1 sinden data donmeli
    if (se.getValidUnsentAddresses() != null)
    {
        // adres dogru fakat mail gonderilemedi
        return -1;
    }
    else if (se.getInvalidAddresses() != null)
    {
        // adres yanlis
        return -2;
    }
    else return -3;
}
catch(MessagingException me)
{
    System.out.println("inside send mail h3:"+me.getMessage());
    return 0;
}
catch (Exception ee) { System.out.println("error:"+ee.getMessage());
                                return 0; }
}
```

/*

This method send mail to the appropriate destination.

Return values: 1 means success

0 failure

*/

public int sendMail(String pFrom,String pTo,String pData,String pSubject)

{

try

{

java.util.Properties properties=System.getProperties();

properties.put("mail.smtp.host",smtpHost);

Session ses = Session.getInstance(properties,null);

MimeMessage message=new MimeMessage(ses);

Address fromAddress=new InternetAddress(pFrom);

message.setFrom(fromAddress);

Address[] toAddress=InternetAddress.parse(pTo);

message.setRecipients(Message.RecipientType.TO,toAddress);

message.setSubject(pSubject);

message.setText(pData);

Transport.send(message);

}

catch(AddressException ae)

{

System.out.println("h1:"+ae.getMessage());

return -1;

}

catch(SendFailedException se)

{

System.out.println("h2:"+se.getMessage());

return -2;

}

catch(MessagingException me)

{

System.out.println("h3:"+me.getMessage());

return -3;

}

return 1;

}

public int logout()

{

System.out.println("inside logout");

```

int retVal=1;
try
{
    folder.close(true);
    //folder.close(false);
    folder = null;
}
catch(MessagingException msg)

{
    System.out.println(msg.getMessage());
    retVal = -1;
}

try
{
    store.close();
    store = null;
}
catch(MessagingException msg)
{
    System.out.println(msg.getMessage());
    retVal = -1;
}
return retVal;
}
/*
This method checks new mail , puts data into a new vector.
*/
public int readMail(String userName,String password,String server,int port,Vector dataVector)
{
    int returnVal = -1;
    java.util.Properties prop = System.getProperties();
    session = Session.getInstance(prop,null);

    try
    {
        store = session.getStore("pop3");
        store.connect(server,port,userName,password);
        folder = store.getFolder("INBOX");
        folder.open(Folder.READ_ONLY);
        messages=folder.getMessages();
        System.out.println("num of messages:"+messages.length);
        for (int i=0;i<messages.length;i++)
    }

```



```
{
mail m = new mail(messages[i].getMessageNumber(),(messages[i].getFrom()[0]).toString(),
                  messages[i].getSubject());
                  dataVector.addElement(m);
}
//folder.close(false);
//store.close();
returnVal = 1;

}
catch (NoSuchProviderException nos)
{
    System.out.println(nos.getMessage());
    returnVal = -1;

}
catch (MessagingException msg)
{
    System.out.println(msg.getMessage());
    returnVal = -1;

}

return returnVal;

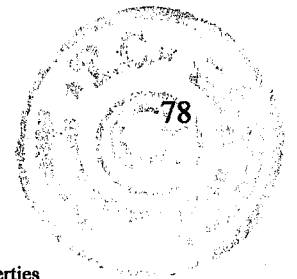
}
```

```
/*
This method is responsible for connecting to the mail server
*/
public int login(String userName,String password,String server,int port)
{

    int returnVal = -1;
    // This is the return value.

    //1 means success

    // -1,-2,-3,-4 means failure
```



```
java.util.Properties prop = System.getProperties(); // System properties
session = Session.getInstance(prop,null);           // session variable

try
{

    store = session.getStore("pop3");
    store.connect(server,port,userName,password);
    folder = store.getFolder("INBOX");
    folder.open(Folder.READ_WRITE);
    returnVal =1;
}
catch (NoSuchProviderException nos)
{

    System.out.println(nos.getMessage());

    returnVal = -2;

}
catch (MessagingException msg)
{

    System.out.println(msg.getMessage());

    returnVal = -3;

}
catch (Exception ie)
{

    System.out.println(ie.getMessage());

    returnVal = -4;

}

return returnVal;

}

/*
```

This method checks new mail and put data into messages variable
It is used after the login method.

```

    */
    public int readMail()
    {
        int returnVal = -1; // return

        try
        {
            //if (folder!=null) // if

            if (folder.isOpen())
            {
                folder.close(false); // to refresh
                numOfOldMessages = numOfMessages; // orijinal number
            }
            else
            {
                System.out.println("ya folder null beyav");
                folder.open(Folder.READ_WRITE); // open it and read

                messages=folder.getMessages(); // get the messages
                numOfMessages = messages.length; // get the number of messages

                returnVal = 1;

                // success
            }
        }
        catch (NoSuchProviderException nos)
        {
            System.out.println(nos.getMessage());
            returnVal = -2;
        }
        catch (MessagingException msg)
        {
            System.out.println(msg.getMessage());
            returnVal = -3;
        }
        catch (Exception ie)
        {
            System.out.println(ie.getMessage());
            returnVal = -4;
        }

        return returnVal;
    }

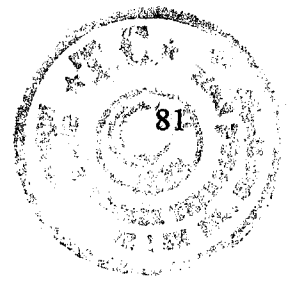
```

```

    }

    /* reads the mails data, gets attachments.
       num is the attachment num.
       this is designed for applet use
    */
    public void readMailData(int num,mail m)
    {
        try
        {
            MimeMessage mes = (MimeMessage) messages[num];
            ContentType ct = new ContentType(mes.getContentType());
            if ("text/html".equalsIgnoreCase(ct.getBaseType()))
            {
                BufferedReader reader = new BufferedReader(new InputStreamReader(mes.getInputStream()));
                String s ;
                StringBuffer sb = new StringBuffer();
                while ((s=reader.readLine())!= null)
                    sb.append(s);
                m.setData(sb.toString());
            }
            else
            {
                Object o = mes.getContent();
                if (o instanceof String)
                {
                    m.setData((String)o);
                    System.out.println("data:"+o);
                }
                else
                {
                    // print the content type
                    //System.out.println("content type:"+mes.getContentType());
                    if (o instanceof MimeMultipart)
                    {
                        MimeMultipart mp = (MimeMultipart) o ;
                        System.out.println("mime multi part");
                        for (int i=0;i<mp.getCount();i++)
                        {
                            MimeBodyPart bp = (MimeBodyPart) mp.getBodyPart(i);
                            if ("text/plain; charset=us-ascii".equalsIgnoreCase(bp.getContentType()))
                            {
                                String data = (String) bp.getContent();
                                m.setData(data);
                            }
                        }
                    }
                }
            }
        }
    }

```



```
    }
    }
}

catch (MessagingException msg) { System.out.println(msg.getMessage()); }
catch (IOException ioe) { System.out.println(ioe.getMessage()); }

}

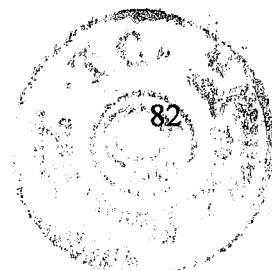
/*
    This method reads the attachemnts
    Num is the message num.
    It stores the text mail data inside mailData,
    other attachments are stored in bodypart array.
*/

public void readMailData(int num)
{
    mailData = null;
    // destroy old values
    bodypart = null;
    numOfBodyPart = 0;
    try
    {
        MimeMessage mes = (MimeMessage) messages[num];           // get the mime message

        ContentType ct = new ContentType(mes.getContentType());   // get the content type

        System.out.println("read mail data:content type"+ct.getBaseType());

        if ("text/html".equalsIgnoreCase(ct.getBaseType()))
        {
            //|| "multipart/related".equalsIgnoreCase(ct.getBaseType()) // if it is text/html
            {
                // set mailData
                //System.out.println("text/html");
                BufferedReader reader = new BufferedReader(
                    new InputStreamReader(mes.getInputStream()));
                String s ;
                // each line of data
                StringBuffer sb = new StringBuffer();               // buffer for all data
                while ((s=reader.readLine())!= null) {               // read till end of it
                    sb.append(s+"\r\n");
                }
                // add data to buffer
            }
        }
    }
}
```



```
mailData=sb.toString();

// set mailData

System.out.println("data:"+mailData);

}
else
{

// data is not text/html

Object o = mes.getContent();
if (o instanceof String)
// if it is string
{

mailData =(String)o;

// set mailData

}
//else

// print the content type
// System.out.println("bb");
String textData = new String();
if (o instanceof MimeMultipart)
{

MimeMultipart mp = (MimeMultipart) o ;
bodypart = new MimeBodyPart[mp.getCount()];
numOfBodyPart = mp.getCount();
int index =0;
System.out.println("number of
parts:"+mp.getCount());

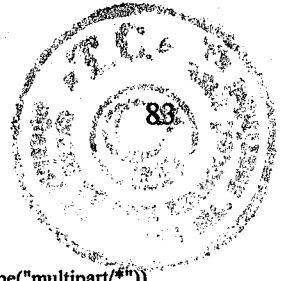
MimeBodyPart bp = (MimeBodyPart) mp.getBodyPart(i);

if (bp.isMimeType("text/plain"))
{

textData=(String)bp.getContent();

if (mailData != null)
mailData=mailData+textData;
else
mailData = textData;
numOfBodyPart--;
}

else
{
```



```
if (bp.isMimeType("multipart/*"))

bodypart[index] = bp;

index++;

}

}

}

}

}

catch (MessagingException msg) { System.out.println(msg.getMessage()); }
catch (IOException ioe) { System.out.println(ioe.getMessage()); }

}

public void forwardMessage(Session sesl,Message mes)
{
    String smtpHost="www.login.com.tr";
    try {
        java.util.Properties properties=System.getProperties();
        properties.put("mail.smtp.host",smtpHost);
        Session ses = Session.getInstance(properties,null);

        System.out.println("message:");
        //ByteArrayOutputStream aa=new ByteArrayOutputStream();
        //mes.writeTo(aa);
        //ByteArrayInputStream bins = new ByteArrayInputStream(aa.toByteArray());
        //System.out.println("msggg:"+aa.toString());

        //ByteArrayDataSource bb=new ByteArrayDataSource(bins,"multipart/mixed" );
        DataHandler ddh=mes.getDataHandler();
        DataSource dsou=ddh.getDataSource();
        System.out.println("c-t:"+dsou.getContentType() );

        ByteArrayInputStream basd=(ByteArrayInputStream)dsou.getInputStream();

        //System.out.println("data:"+getDBByte(basd));

        ByteArrayDataSource hs = new ByteArrayDataSource(basd,dsou.getContentType() );

        Message forward = new MimeMessage(session);
        forward.setSubject(mes.getSubject());
```

T.C. YÜKSEKÖĞRETİM KURULU
BİLGİ YÖNETİMİ MERKEZİ



```
forward.setFrom(new InternetAddress("tolga@login.com.tr"));
forward.addRecipient(Message.RecipientType.TO, new InternetAddress("informix@login.com.tr"));

BodyPart mbodypart = new MimeBodyPart();
//mbodypart.setDataHandler(mes.getDataHandler());
DataHandler dh = new DataHandler(hs);
mbodypart.setDataHandler(dh);
Multipart mp = new MimeMultipart();
mp.addBodyPart(mbodypart);
forward.setContent(mp);
Transport.send(forward);
}

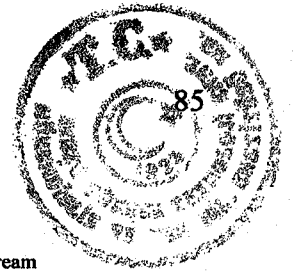
catch(AddressException ae)
{
    System.out.println("h1:"+ae.getMessage());
}
catch(SendFailedException se)
{
    System.out.println("h2:"+se.getMessage());
}
catch(MessagingException me)
{
    System.out.println("h3:"+me.getMessage());
}
catch(IOException ioe)
{
    System.out.println("h3:"+ioe.getMessage());
}

}

public String getDBByte(ByteArrayInputStream value)
{
    StringBuffer result=new StringBuffer();

    // It stores the byte data that comes from database
    int b=0; // The single value that is read from the db
    while ( (b=value.read())!= -1) // Read till the end of the data
    {
        char c = (char)b;

        result.append(c);
    }
}
```



```
}  
//value.close();  
System.out.flush();  
return result.toString();
```

// close the stream

// flush the contents

```
}
```

```
}
```

