MoReCon: MOBİL YAZILIM UYGULAMALARI İÇİN BAĞLAM-DUYARLI VE SERVİS TABANLI BİR ARAKATMAN YAZILIMI

ONUR SOYER

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ

TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

EYLÜL 2010

ANKARA

Fen Bilimleri Enstitü onayı

_____

Prof. Dr. Ünver KAYNAK

Müdür

Bu tezin Yüksek Lisans derecesinin tüm gereksinimlerini sağladığını onaylarım.

_____

Doç. Dr. Erdoğan DOĞDU

Anabilim Dalı Başkanı

Onur SOYER tarafından hazırlanan "MoReCon: Mobil Yazılım Uygulamaları İçin Bağlam-Duyarlı ve Servis Tabanlı Bir Arakatman Yazılımı" adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

_____

Doç. Dr. Erdoğan DOĞDU

Tez Danışmanı

Tez Jüri Üyeleri

Başkan :Doç. Dr. Erdoğan DOĞDU _____

Üye : Yrd. Doç. Dr. Esra KADIOĞLU ÜRTİŞ _____

Üye : Doç. Dr. Kemal BIÇAKCI _____

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada orijinal olmayan her türlü kaynağa eksiksiz atıf yapıldığını bildiririm.

…….

Onur SOYER

Üniversitesi : **TOBB Ekonomi ve Teknoloji Üniversitesi**

Enstitüsü : **Fen Bilimleri**

Anabilim Dalı : **Bilgisayar Mühendisliği**

Tez Danışmanı : **Doç. Dr. Erdoğan DOĞDU**

Tez Türü ve Tarihi : **Yüksek Lisans – Ağustos 2010**

Onur SOYER

MoReCon: MOBİL YAZILIM UYGULAMARI İÇİN BAĞLAM-DUYARLI ve SERVİS TABANLI BİR ARAKATMAN YAZILIMI

ÖZET

Mobil cihazların hızla yaygınlaşması ile birlikte bu cihazlar için geliştirilen uygulamalar da çeşitlilik kazanmaktadır. Bu cihazlarla ilgili en önemli kavramlardan birisi "bağlam-duyarlılık" (context-awareness) dir. Bu kavram, mobil cihazların taşıyan kişinin özelliklerine ve cihazın bulunduğu ortama ve çevreye göre tepki vermesi, üzerindeki uygulamaları duruma göre uyarlamasını ifade eder. En bilinen ve yaygın kullanılan bağlam-duyarlılık bilgilerinden birisi "yer" (location) bilgisidir. Örneğin, mobil cihaz bulunduğu yere göre (GPS bilgisi) kişiye yapılacak işlerini bildirebilir, ya da yerel reklam bilgileri sunabilir.

Bu tez çalışmasında bağlam-duyarlılık konusunda daha önce yapılan çalışmalar ve mobile cihazlarda bağlam-duyarlı uygulama geliştirebilmek için geliştirdiğimiz MoReCon adlı arakatman yazılımı (middleware) sunulmuştur. MoReCon, uygulamalardan bağımsız olarak bağlam-duyarlılık konusunda gerekebilecek her türlü bilgiyi sağlayan ve saklayan kapsamlı bir arakatman yazılımdır. Bu yazılım RESTful web servisleri ile esnek bir şekilde erişilebilir, bağlam bilgilerini esnek bir şekilde saklama, erişme ve güncellemeyi sağlayan basit bir protokole sahiptir. MoReCon, mobil cihazlar üzerinde çalışmak üzere küçük olacak (compact) şekilde tasarlanmıştır.

Anahtar Kelimeler: Bağlam duyarlılık, RESTful web servisleri, J2ME, mobil uygulamalar

University : **TOBB University of Economics and Technology**

Institute : **Institute of Natural and Applied Sciences**

Science Programme : **Computer Engineering**

Supervisor : **Assoc. Prof. Dr. Erdoğan DOĞDU**

Degree Awarded and Date : **M.Sc. – August 2010**

Onur SOYER

MoReCon: SERVICE-BASED and CONTEXT-AWARE MIDDLEWARE FOR MOBILE APPLICATIONS

ABSTRACT

With the rapid growth of mobile devices and their usage, now there are also a wide variety of applications developed for these devices. One of the most important concepts for these devices is "context-awareness". This concept refers to the fact that these devices and the applications on them react and respond according to the context in which the person carying device is, such as the surrounding environment or the person's personal preferences. One of the most known and common usage of context-aware information is the location information. For example, mobile device can notify the user according to its location and present the tasks or make advertisement suggestions, etc.

In this thesis, we surveyed some of the previous work about context-awareness and presented MoReCon middleware that we developed for building context-aware application on mobile devices. MoReCon is helpful in providing and maintaning the context information for the mobile applications. It is a modular and servis-oriented middleware, serves the context information via RESTFul web services via a simple protocol. MoReCon is designed as a compact middleware to work on mobile devices.

Keywords: Context-awareness, RESTful web services, J2ME, mobile applications

TEŞEKKÜR

Çalışmalarım boyunca değerli yardım ve katkılarıyla beni yönlendiren hocam Doç. Dr. Erdoğan DOĞDU'ya ve her türlü fedakârlığı yaparak benim bu günlere gelmemi sağlayan Anneme ve Babama teşekkürü bir borç bilirim.

İÇİNDEKİLER

ŞEKİLLERİN LİSTESİ

ÇİZELGELERİN LİSTESİ

1. GİRİŞ

Gelişen teknoloji ile birlikte mobil cihazlar birçok özellik kazanmıştır. Bu özellikler kullanıcıların bir takım işlemlerini hareket halindeyken gerçekleştirmelerini sağlamaktadır.

Günümüz mobil cihazlarında geliştirilen uygulamalar cihazdaki alıcılar ile desteklenmektedir. Cihazlardaki alıcılara ve servislere bağlı olarak, GPS, sıcaklık, gürültü, hava kirliliği, hava durumu, trafik bilgisi vs. gibi bilgiler elde edilebilir. Bu bilgilerin her biri birer bağlamdır (context). Bu bağlam bilgilerini kullanarak hizmet sunan bir uygulama ise bağlam-duyarlı sistem olarak tanımlanır. Bağlam duyarlı uygulamalar kullanıcının ihtiyaçlarına, karakteristik bilgisine ve diğer bağlamlara göre farklı hizmetler verirler.

Bağlam, kullanıcı odaklı ve çevresel olarak kategorize edilebilir. Kullanıcı odaklı bağlam, kullanıcının profiline, alışkanlıklarına, fizyolojisine, kısacası kullanıcının doğrudan kendisi ile ilgili olan bilgileri içerebilir. Çevresel bağlam ise kullanıcının bulunduğu yere, hava durumuna, çevredeki insanlara, çevredeki mağazalara vs. gibi bilgilere bağlıdır.

Bu bağlamların bir arada kullanılması ile kullanıcıya sunulacak olan hizmetler daha güçlü ve anlamlı hale getirilmektedir. Böylelikle kullanıcının uygulamayı anlaması değil uygulamanın kullanıcıyı anlaması sağlanmış olur. Bu da uygulamayı kullanan kullanıcının hem vakitten kazanmasını hem de çevresinde olan bitenlerden haberdar olmasını sağlar.

Bu tezin amacı modüler, uygulamadan bağımsız, katmanlı ve servis-tabanlı mimariye uygun bir arakatman yazılımı geliştirmek, böylece bağlam-duyarlı uygulama geliştirmeyi kolaylaştırmaktır.

Bu tez kapsamında kullanıcının profili ve koordinat gibi bağlam bilgilerini kullanabilen, iletişimini RESTful web servisleri ile sağlayan J2ME tabanlı bağlam duyarlı bir arakatman yazılımı geliştirilmiştir. Bu arakatman, bağlam duyarlı

uygulama geliştirmek isteyen geliştiricilere önceden tanımlanmış servisleri sunarak kolaylık sağlamakta, uygulama ile cihaz arasında köprü görevi görmektedir.

Bu tezde işlenen başlıklar şöyledir: İkinci bölümde, bağlam ve bağlam duyarlığın ne olduğu, mobil cihazlarda nasıl bağlam elde edilebileceği ve buna bağlı uygulamalar ile anlatılacak. Mobil uygulama platformları ve ilişkili teknolojiler bölümünde, MoReCon arakatman yazılım mimarisi ve performans testleri işlenecek ve yapılan performans testleri değerlendirilecek. Son olarak da sonuçlar ve gelecek çalışmalar sunulacaktır.

2. MOBİL YAZILIM UYGULAMALARI VE BAĞLAM DUYARLILIK (CONTEXT-AWARENESS)

İlk cep telefonu 1973 yılında Motorola'da çalışan Martin Cooper tarafından üretildi. O günden bugüne mobil cihazlar büyük bir gelişim gösterdi. O tarihte üretilen mobil cihaz 850 gr. iken bugün ki mobil cihazlar 80-90 gr. ağırlığında. Bunun yanında mobil cihazların azalan ağırlığının yanında işlevsellik ve işlem gücü de artmıştır. Örneğin, önceleri cep telefonlarında sadece konuşma yapılabilirken ilerleyen dönemlerde SMS ve internet özellikleri gelmiştir. Buna bağlı olarak bu cihazlarda geliştirilen uygulamaların çeşitliliği ve verimliliği artmıştır. Hem cihazların işlem gücü hem de internet gibi önemli teknolojilerin kullanılabilmesi ile kapsamlı uygulamaların geliştirilmesini sağlamıştır. Artık mobil bir cihaz ile banka işlemleri yapabilir, oyun oynayabilir, uzak masaüstü ile uzaktaki bir bilgisayara bağlanabilirsiniz. Bu tarz uygulamaların gelişmesi ile mobil cihazların kullanımı artmış ve gündelik hayatımızda vazgeçilmez bir yer almıştır.

Etrafımızda, ölçüm yapabileceğimiz birçok durum gözlemleyebiliriz. Bu ölçümler bizlere belirli bir bilgi sağlarlar. Ama bu bilgiler tek başlarına anlamlı olmayabilir. Eğer bu ölçümleri bir araya getirebilirsek daha kullanışlı ve anlamlı bilgiler elde etmiş oluruz.

Bu bölümde bağlam ve bağlam duyarlılık kavramları irdelenecek, bunlara bağlı olarak geliştirilen uygulamalar incelenecek ve mobil cihazlarda bağlam-duyarlı uygulama geliştirmenin zorluklarından bahsedilecektir.

2.1. Bağlam ve Bağlam Duyarlılık Nedir?

Bir varlığın durumunu karakterize eden bilgiye "bağlam" (context) denir. Bu varlık, kullanıcı ve uygulama arasındaki karşılıklı etkileşim ile ilgili olan bir kişi, bir yer veya bir nesne olabilir [1]. Bağlam bilgileri iki şekilde elde edilir:

- Kullanıcının sağladığı bilgilerle (profil, arkadaşlar, ajanda, vs.)

- Algılayıcılar (sensör) ile sağlanan bilgilerle (sıcaklık, koordinat, ses düzeyi ölçer, vs.)

Bağlam bilgileri bulunduğu sisteme göre farklılık gösterebilir. Yani bir sistemde bağlam olarak kullanılan bir bilgi diğer sistemde bağlam olmayabilir.

Bir sistem, kullanıcının işlevi ile ilgili olarak kullanıcıya ilgili bilgi sağlamak için bağlam kullanıyorsa, bu sistem "bağlam duyarlı" (context-aware) olarak tanımlanır [1]. Bu sistem, bir mobil cihaz olabilir. Kullanıcıya sağlayacağı bilgiler ise mobil cihazın sahip olduğu teknolojiye göre değişebilir. Artık günümüz cep telefonlarında genel bağlam bilgileri rahatlıkla elde edebilir. Mesela iPhone [18] ile internete bağlanabilir, kullanıcının anlık koordinatlarını GPS ile saptayabilir ve pusula ile hangi yönde olduğu öğrenilebilir. Örnek olarak, bu teknolojileri kullanarak oluşturulacak bir yapı ile kullanıcının kişisel bilgilerine dayanarak, çevresinde olan sevdiği mağazalarda indirim varsa veya sevdiği türde bir film gösterime girmişse haber veren bir uygulama yapılabilir.

Bağlam duyarlı bir uygulama, güzel tasarlanmış bir arayüz, elde edilen verinin uygulama ile olan ilişkisi, elde edilen verinin verimliliğini arttırmak, bu veriler için servislerin keşfi ve bağlam sağlayıcı teknolojileri bir arada barındırmalıdır.

2.2. Bağlam Duyarlı Uygulamalar

Bağlam duyarlı bir uygulama için öncelikle bağlam elde etmek şarttır. Bağlamlar mobil cihazın sağladığı özellikler ile elde edilebilir. Günümüz mobil cihazlarında en çok kullanılan bağlam olan koordinat bilgisini GPS veya baz istasyonları aracılığı ile elde edebiliyoruz. Bunun yanında, her telefonda olan internet erişimini kullanarak web servisleri aracılığı ile bağlam bilgisi elde etmek mümkün. Geliştirilecek uygulamaya özel olarak dışarıdan bağlam bilgisi sağlayan cihazlarda kullanılabiliyor. Örneğin ses düzeyini ölçen bir cihaz, RFID, medikal bilgi sağlayan sensör gibi [17][18].

Bağlam elde ederken kullanılan farklı teknikler vardır. Bu teknikler donanım tabanlı veya yazılım tabanlıdır. GPS, sıcaklık ölçer gibi teknikler donanım tabanlı olup, kullanıcı tarafından girilen veya bazı gözlemleyici programlar tarafından sağlanan veriler yazılım tabanlıdır. Sonsörler mimarinin alt seviyesinde çalışırlar. Burada bağlam elde ederek bunu sisteme aktarırlar. Eğer cihaza bütünleşik bir sistem yoksa sensörler ile iletişim web servisleri üzerinden sağlanır [18][19].

Bağlam türü ve elde etme yöntemi farklılık gösterebilir. Bu bağlamlar arasında en popüler olanı koordinat bilgisidir [10][11][12]. Diğer yaygın kullanılan bağlamlar ise takvim, aktivite, kullanıcı profili, çevresel faktörler gibidir [14][16][17].

Bağlamların toplanması dışında bu bağlamların sunulması da önemlidir. İnternet ortamında veri iletişimi genellikle web servisleri ile yapılmaktadır [11][12][14]. Bunun yanında, verilerin XML [19], JSON [20] vs. gibi formatlarda sunulması verinin farklı platform veya uygulamalarda kullanılabilmesini sağlamaktadır [16][17][18].

Veri saklama işlemleri, web servisleri üzerinden veya direk olarak veri tabanına erişerek yapılır. Web servisi kullanmayan sistemler veri tabanına direk olarak erişip işlem gerçekleştirirler [17].Veriler, XML ve RDF/OWL gibi dillerle tanımlanıp saklanırlar.

2.3. Mobil Cihazlarda Bağlam Duyarlı Uygulama Geliştirmenin Zorlukları

Bağlam bilgisi mobil cihazların sensör yeteneklerine bağlı olarak çeşitlilik gösterebilir. Bunlara erişim, veri elde etme ve bu verilerin yönetimi kolay değildir. Bu sebeple bağlam duyarlı uygulama geliştirirken her cihazda farklı işlemler yapmak gerekecek uygulamalar taşınabilir olmayacaktır. Bağlam bilgilerinin elde edilmesi ve yönetimi modüler bir yaklaşımla ayrı bir yazılım parçasına devredilirse bağlam-duyarlı uygulamalar bu detaylardan bağımsız olarak geliştirilebilirler.

Bağlam yönetiminde karşılaşılan iki farklı problem vardır. Birincisi, mobil bir cihazın donanım olarak yetersiz olması, ikincisi ise yazılımsal problemlerdir.

Mobil cihazlar, küçük ve belli özelliklerle sınırlandırılmış cihazlardır. Bu özellikler genellikle kullanıcının, telefon etmek, internete girmek, mesaj atmak, ajanda kullanmak gibi temel ihtiyaçlarını karşılayacak şekildedir. Bundan dolayı bir mobil cihazdan pek fazla bir bilgi elde edilememektedir.

Bağlamlar, elde edilen bilgilerden oluşurlar. Ne kadar çok farklı alanda bilgi elde edinilirse o kadar çok çeşitlilik yaratılmış olur. Bu bilgilerin ediniminde sensörlerin yeri çok önemlidir. Sensör çeşitliliği bağlam çeşitliğini artırır. Fakat mobil cihazlardaki sınırlı sensörlerden dolayı yetersiz bağlam bilgileri elde edilmektedir. Bundan dolayı da verimli bir bağlam duyarlı sistem kurulamamaktadır. Örneğin, bir kullanıcı tiyatrodayken telefonuna bir bildirim geldiğinde telefonun ışığı yanar. Burada cihazın o ışığı yakmaması gerektiğini anlamasını gerektirecek sensör yoktur.

Bir diğer konu da, yazılımsal problemlerdir. Bağlamların bir arada uyumlu çalışması için düzgün bir algoritma gerekir. Düzgün yazılmamış bir algoritma ile kullanıcı kendisini bilgi karmaşasının içerisinde bulabilir. Bunun yanında, bağlamı donanımsal yoldan elde etme dışında yazılımsal yoldan da elde etmek gerekir. İnternete ulaşabilen bir cihaz ile web servisleri ile sunulan bazı servisleri kullanarak (Örn. Hava durumu, harita vs.) bağlam duyarlı bir sistem kurulabilir. Bu şekilde desteklenmeyen bir sistem ile verimli bir bağlam duyarlı sistem kurmak zorlaşır.

Geliştirilmiş yapılarda, bağlam-duyarlılık özelliği sisteme bütünleşik bir şekildedir. Bu da daha sonra başka uygulamalarda tekrar kullanımının olmamasına neden olur.

Bir başka zorluk, geliştirilen yapıların modüler olmamasıdır. Bu yüzden, geliştirilen uygulama bağlam-duyarlılık kavramından bağımsız olarak hareket edemeyecektir.

Ayrı bir modül olarak geliştirilmeyen yapılarda daha sonra düzenleme yapılması gereksinimi ortaya çıktığında büyük bir kargaşaya sebep olur. Basit bir düzenleme işleminde bile bütün sistemi etkileyecek bir durum söz konusu olabilir.

2.4. Web Servis Tabanlı Bağlam Duyarlı Mobil Uygulama Çerçeveleri ve Kullandıkları Teknikler

Bu bölümde bağlam-duyarlılığı kullanmış bazı mobil arakatman ve çerçeveler incelenmiştir. İncelemede kullanılan teknikler ele alınmış ve buna göre sınıflandırılmıştır. Bu teknikler, bağlam bilgisi ve sunumu, bağlam sensör teknikleri, bağlam depolama teknikleri ve bağlam yayma teknikleridir. Aşağıda Truong ve diğerlerinin bağlam-duyarlı ve web servis tabanlı sistemler üzerine yaptıkları inceleme sonuçları üzerinden mobil bağlam-duyarlı sistemler seçilerek özetlenmiştir [25]. Referans verilen Çizelgelarda farklı çalışmaların incelendiği görülmektedir. Bunun sebebi referans verilen Çizelgeların bu şekilde incelenmiş olmasından kaynaklanmaktadır.

2.4.1. Bağlam Bilgisi ve Sunumu

Birçok bağlam bilgisi türü sıralamak mümkündür. Bunlardan en popüler olanı "konum" bilgisidir. Bunun yanında "takvim, görev, tercih, ağ durumu" gibi başka türlerde mevcuttur.

Çizelge 1 - Desteklenen bağlam bilgisi türleri [25]

Sistem	Bağlam/Bilgi Türü						
	Takvim	Konum	Aktivite/ Görev	Tercih	Takım	Cihaz	Ağ
Akogrimo		+		+		+	+
Anyserver				+		+	+
ESCAPE		+	+	+	+	+	+
inContext	+	+	+	+	+	+	

Çizelge 1'de bağlam duyarlı web servis kullanan bazı yapıların kullandığı bağlam bilgi türleri gösterilmiştir. Takvim, konum, görev, tercih, ağ gibi bağlamlar bağlam duyarlı yapılarda en çok kullanılan türlerdir. Bu türlerin seçimi servis/görev seçimi [27] ve takım çalışmasını [10] desteklemesinde önemli rol oynar.

Çizelge 1'de verilen bağlam türlerinin açıklaması şu şekildedir:

1. **Konum**, GPS ile veya GSM şebekesi (eğer mevcut ise) ile elde edilen koordinat bilgisidir.
2. **Takvim**, gündelik hayatta devamlı kullandığımız gün, hafta, ay ve yıl bilgilerini sağlayan araçtır.
3. **Takım**, bir iş veya etkinlik üzerinde ortak çalışma ortamı oluşturan araçtır.
4. **Aktivite/Görev,** kullanıcının bulunduğu eylemin bilgisidir.
5. **Tercih**, kullanıcının kişisel bilgilerine bağlı oluşturmuş olduğu isteklerdir.
6. **Cihaz**, gündelik hayatta kullandığımız mobil cihazlardır.
7. **Ağ**, yerel veya internet üzerinden farklı kullanıcılar veya yapılar ile etkileşimin oluşturulmasını sağlar.

Çizelge 2'de kullanılmış olan veri modelleri listelenmiştir. Çizelge 2Bağlam bilgisinin sunumunda farklı modeller ve diller kullanılabilir. Bağlam sunumu da bu diller ile gerçekleştirilir. Bu modellerden en bilineni XML'dir. Bunun yanında RDF ve OWL dilleri kullanan yapılarda vardır. Anyserver [28] ve ESCAPE [16] bağlam bilgilerini XML kullanarak sunar. Akogrimo [26] ise OWL yapısı ile bağlam bilgileri sunar.

Çizelge 2 - Bağlam sunum teknikler [25]

Bağlam/Dil	XML	OWL
Akogrimo		+
Anyserver	+	
ESCAPE	+	

2.4.2. Bağlam Algılama Teknikleri

Bağlam bilgisi, GPS, gözlem (monitoring), kullanıcı tarafından sağlanan bilgiler gibi yazılım tabanlı veya donanım tabanlı sensörler aracılığı ile elde edilir. Bağlam bilgisi elde eden bu sensörlere alt seviye iletişim protokolleri ile erişilir.

Çizelge 3'de bağlan algılama yöntemleri, nasıl elde edildiği ve arayüzleri hakkında bilgi verilmiştir. Bağlamlar, sistem tarafında otomatik olarak veya kullanıcı tarafından harekete geçirilerek elde edilir. Bağlam algılama yöntemi geliştirilen uygulamalardan gelen isteğe bağlı olarak algılayıcılar, otomatik veya dışarıdan harekete geçirilerek bağlam bilgisini elde edebilirler. Algılama yöntemleri, donanımsal veya yazılımsal olabilir. Yazılımsal algılama web servisleri aracılığı ile, fiziksel algılama ise algılayıcı cihazlar ile gerçekleşir. Algılayıcılar tarafından elde edilen bağlamlar, sorgu, abonelik ve sadece itme yani değişen bağlam bilgisinden uygulamayı sürekli haberdar etme yöntemi ile elde edilirler.

Çizelge 3 - Bağlam algılama teknikleri [25]

Sistem	**Mod**		**Algılama Tekniği**		**Algılama Arayüzü**		**Veri Çekme ve Sunma**		
	Otomatik	**Manuel**	**Algılayıcı**	**Sorgulama**	**Web Servis**	**Özel Tasarım**	**Sorgu**	**Abonelik**	**Sadece İtme**
Akogrimo	+			+		+		+	+
Anyserver	+			+		+			
ESCAPE	+	+	+		+		+	+	
inContext	+			+	+	+			+

Web servis tabanlı bağlam duyarlı sistemlerde bağlam bilgisinin elde edilmesi önemlidir. Algı tekniklerinin gelişmesiyle birlikte birçok bağlam bilgisine ulaşmak kolaylaşmıştır. Bugün mobil cihazlara bağlam bilgisi sağlayan bir takım algılayıcılar

bulmak mümkündür. Örneğin ESCAPE sisteminde [16] web servis aracılığı ile algılayıcılara erişilerek bağlam bilgisi edinilmektedir. Ayrıca, bazı web servis tabanlı bağlam duyarlı sistemlerde algılayıcılar ile üçüncü parti sistemler ve bazı protokoller ile haberleştirmek de mümkündür. Sonuç olarak, algılayıcı arayüzler ile bunları destekleyen bileşenler çeşitlilik gösterebilir. Örneğin Akogrimo'da [26] algılayıcılar, bağlam yöneticisi ile SIP (Session Initiation Protocol) ve Web servisleri gibi farklı protokollerin yardımı ile haberleşebilmektedir.

2.4.3. Bağlam Depolama Teknikleri

Web servis tabanlı bağlam duyarlı uygulamalar bağlam bilgilerini saklamak için farklı teknoloji ve yöntemler kullanmıştır. Bazı yapılar depolamayı ayrı bir serviste olarak sunarken bazıları ayrı olarak sunmamışlardır. Örneğin Akogrimo [26], bağlam bilgilerini depolamayı ayrı bir servis olarak sunmamıştır. Bunun yerine depolama bağlam yöneticisinin bir parçası olarak kullanılmaktadır. Bağlam bilgileri, ilişkisel veritabanlı, XML veya OWL olarak saklanabilir. Eğer başka bir teknik geliştirilmemişse, saklanan bu verilere erişim genellikle Web servisleri aracılığı ile sağlanır.

Çizelge 4'de veri saklama modelleri, bu verilerin sorgulamak için kullanılan diller ve kullanılan veritabanları gösterilmiştir. Veri saklama modelleri merkezi ve dağıtık olarak kullanılmıştır. Kullanılan veritabanı türleri ise ilişkisel, XML ve RDF/OWL'dur. Veritabanlarına erişim web servisleri aracılığı ile sağlanır. Saklanan verileri SQL, anlamsal sorgulama dili SPARQL ve XML doküman tarayıcısı olan XPath/XQuery kullanılarak sorgulamalar yapılmıştır.

Çizelge 4 - Bağlam Depolama Teknikleri [25]

Sistem	Saklama Modeli		Veritabanı			Erişim Arayüzü		Sorgu Türü		
	Merkezi	Dağıtık	İlişkisel	XML	RDF/OWL	Web Servis	Diğer	SQL	XPath/ XQuery	SPARQL
Akogrimo	+		+				+	+		
ESCAPE		+		+		+			+	
inContext	+	+			+	+				+

2.4.4. Bağlam Yayma Teknikleri

Web servis tabanlı bağlam duyarlı yapılarda bağlamlar web servisleri aracılığı ile veya özel protokoller ile yayımlanırlar. Örnek olarak, inContext [10], bağlam sunmak için hem dağıtık hem de merkezcil model kullanır. Böylelikle, bağlam bilgilerine bağlam deposundan veya bağlam bilgisi sağlayan farklı servisler aracılığı ile ulaşılabilir. ESCAPE çerçevesinde (framework) [16] ise, bağlam yayımı web servisleri gibi özel protokoller ile sağlanır. Hem Web servis ile algılayıcılara erişerek hem de doğrudan bağlam yöneticisi ile sorgulama yaparak bağlam bilgilerine ulaşılabilir.

Çizelge 5'de bağlam yayma teknikleri gösterilmiştir. Bağlamlara erişim, merkezi olarak saklanan bağlam kaynaklarından veya P2P bağlam sağlayan servisler yoluyla olur. Bağlam yayma işlemi, basit bir kullanım sağlayan SOAP (Simple Object Access Protocol*)* mesaj başlıkları ile sağlanmıştır. Doğrudan bağlam bilgisi ya da sadece referans kaynağı mesaj başlıklarına eklenerek dağıtım sağlanır. Web servisler üstünden bağlam yayma işlemleri abonelikler veya sorgu şeklinde gerçekleştirilmektedir. Bu sorgular doğrudan web servisi üzerinden gerçekleştirilir.

Çizelge 5 - Bağlam Yayma Teknikleri [25]

Sistem	**Ağ Dağılımı**		**Doğrudan Taşıma Dağılımı (Direct Transport Distrubution)**	**Erişim Yolu**			
	Merkezi	**P2P**	**SOAP Uzantısı**	**Web Servisleri Üzerinde Sorgu**	**Web Servislere Abonelik ve Uyarı**	**Web Servis Geri Çağırma ile Abonelik**	**Özel Protokol Üzerinden Abonelik**
Akogrimo	+				+	+	
ESCAPE	+	+		+		+	+
inContext	+		+		+		

3. MOBİL UYGULAMA PLATFORMLARI VE İLİŞKİLİ TEKNOLOJİLER

3.1. Web Servisleri: İletişim Protokolü

Servis Tabanlı Mimari (Service Oriented Architecture)

SOA sistem geliştirilmesinde ve bütünleştirilmesinde kullanılan esnek bir tasarım prensibidir. Nasıl yazılım konusunda nesne tabanlı programlama bir yaklaşımın adı ise SOA da aynen o şekilde düşünülebilir. SOA felsefesine göre, sistemlerin bütünleşmesi servisler aracılığıyla ara katman (middleware) üzerinden olmalı, kolaylıkla yönetilebilmeli, değişiklikler rahatlıkla yapılmalıdır. Bütünleşme denilen şey ise büyük sistemlerin birbirleriyle olan konuşmaları, haberleşmeleridir. Örneğin A sisteminde bir veri değişikliği olduğunda B sisteminin de bundan haberdar olması bir bütünleşme gerektirir.

Servis tabanlı mimari modeli, şu üç ana unsurdan oluşur:

1) Servis Sağlayıcı: Ağ (internet/intranet) üzerindeki bir servisin arayüzüne erişimi sağlamakla görevli düğümdür.

2) Servis Kullanıcı: Kendi iş çözümünü uygulamak üzere, servis sağlayıcıyı kullanan düğümdür. SOA modeli bağlamında kullanıcılar, bir uygulama değil, daha çok yine ağ üzerindeki bir düğüm biçiminde isimlendirilmektedir. Ancak SOA modeli üzerine, web servisleri tasarlandığında, kullanıcılara da -web servisini kullanan istemci uygulamalar- gözü ile bakılabilir. Bu uygulamalar, web servisleri ile SOAP ve HTTP üzerinden iletişim sağlayabiliyor olmaları ile karakterizedir.

3) Servis Arabulucu (Broker) : Global ağ üzerindeki servislere ilişkin tanımları sunan ve servislerin ağ üzerindeki lokasyonlarının bulunmasında kullanılan, bir tür adres defteridir. Kullanıcılar, broker'ı sorgulayarak, kendilerine gereken servis sağlayıcıyı ya da servisi bulurlar. Web servislerine dayalı bir model içerisinde UDDI

(Universal Description Discovery Integration), service broker görevini üstlenmektedir.

Servis Odaklı Mimari modeline ilişkin yukarıda sayılan unsurlar arasında, üç farklı etkileşimden söz etmek mümkündür.

1) Servislerin Yayınlanması: Sağlayıcılar, servislerini bir servis arabulucuda yayınlarlar. Yayınlanan bilgiye, servis arayüz tanımı, servisin lokasyon bilgisi, yanı sıra diğer olası yardımcı bilgi ve dokümantasyonlar dahildir. Bu yayınlama işlemi, servise ilişkin bazı kimlik bilgilerinin arabulucuya tanımlanması hatta broker üzerine kaydedilmesi biçiminde düşünülebilir.

2) Servislerin Bulunması: Kullanıcılar, arabulucuyu kullanarak aradıkları servisleri bulurlar.

3) Servislerin Kullanımı (binding): Gerek servislerin bulunuşu gerekse de kullanımları (binding), uygulamaların kendilerini dinamik olarak yapılandırabilmesine imkan verecek şekilde ve yine dinamik olarak gerçekleştirilebilir. Sözgelimi bir uygulama, servis sağlayıcıya belirli bir süre içerisinde bağlanamazsa, çalışma zamanında başka bir servis sağlayıcıya geçiş kararını otomatik olarak verebilir.

REST (Representational State Transfer) ve RESTful Web Servisi

Rest, World Wide Web gibi dağıtık ortamlar için geliştirilmiş yazılım mimari türüdür. Bu terim ilk defa 2000 yılında Roy Fielding tarafından kullanılmıştır [5]. REST türü mimarilerde sunucu ve istemci olur. İstemci sunucuya bir istek gönderir, sunucu isteği işler ve cevabı istemciye gönderir. İstekler ve cevaplar kaynakların sunumunun transferi etrafında kuruludur.

Bir istemci herhangi bir zamanda uygulama durumu ve dinlenme (at rest) arasında geçiş yapabilir. Dinlenme durumundaki bir istemci, kullanıcısı ile etkileşim halinde olabilir ve bu durum sunucu tarafında herhangi bir yüke sebep olmaz. İstemci yeni bir duruma geçmeye hazır olduğunda istek göndermeye başlar.

REST ilk olarak HTTP'nin bir bağlamı olarak tanımlandı. Ama REST sadece bu protokolle sınırlı değildir.

REST mimarisinde uygulanabilecek bazı kısıtlar vardır. Bunlar:

İstemci-Sunucu (Client-Server): İstemciler ile sunucular farklı arayüze sahiptir. Örneğin, istemciler veri depolamaz. Bu işi sunucular yapar. Bu da istemcilere taşınabilirlik kabiliyeti getirir. Sunucular ise kullanıcı arayüzü ve ya kullanıcı durumları ile ilgilenmezler. Bu da suncuları daha basit ve ölçeklenebilir yapar. Arayüzde bir değişiklik olmadığı sürece İstemci ve Sunucular ayrı ayrı geliştirilir.

Durumsuzluk (Stateless): İstemci, gelen istekleri yanıtlayabilecek kadar bilgiyi kendi tutar. Sunucunun bir durumu olabilir. Bu kısıt sunucu tarafında URL'lerin kaynak olarak saklanmasını gerektirir. Bu olay, sunucunun gözlenebilmesinin dışında kararlılıkta sağlar.

Saklanabilirlik (Cacheable): İstemciler yanıtları saklayabiliyorlar. Yanıtlar, açık ve ya kapalı bir şekilde, saklanabilir olduklarını belirtmelidir. İyi ayarlanmış saklanabilirlik istemci ile sunucu arasında hızlı iletişimi sağlar.

Katmanlı Sistem (Layered System): Bir istemci direk olarak son sunucuya veya aracı sunucuya bağlandığını söyleyemez. Aracı sunucu yük dengesini sağlama özelliği ve cache paylaşmayı sağlayarak sistem ölçeklenebilirliğini arttırabilir.

Uniform Interface (Düzgün Arayüz): REST servisinde arayüz tasarımı temel teşkil eder. Kaynaklar tanımlanması, yönetilmesi ve açıklayıcı mesajlarla bilgilendirmesi gerekir.

REST Arayüz Prensipleri

Kaynakların Tanımlanması: Özgün kaynaklar, web tabanlı REST sistemlerde kullanılan URI'lar gibi, isteklerde tanımlanırlar. İstemciye gönderilen kaynakların sunum şekilleri birbirinden farklıdır. Mesela, sunucu bir istek karşısında veritabanını göndermez. Bunun yerine kayıtları JSON, XML veya HTML şeklinde sunar.

Kaynakların Yönetilmesi: İstemci kaynakları, sahip olduğu izinler dahilinde, düzenleyebilir. Örneğin ekleme, silme, düzenleme yapabilir.

İşlem Mesajları: Yapılan işlemler hakkında istemciye gerekli mesajların iletilmesi gereklidir. Bu istemcinin gerçekleştirmek istediği işlemin durumu hakkında bilgi edinmesi için önemlidir [5].

REST'in Sağladığı Kolaylıklar ve Ana Prensibi

REST'in sağladığı kolaylıklar:

- Bileşenlerin etkileşiminin ölçeklendirir.
- Arayüzün standart oluşturur.
- Bileşenlerin birbirinden bağımsız olmasını sağlar.
- Gecikmenin azalmasını sağlar.
- Güvenliği sağlar.
- Kapsülleme (Encapsulation) yapar.

REST'de diğer önemli bir konuda her bir kaynağın HTTP'deki URI gibi bir tanımlayıcısı olmasıdır. Bu kaynakları işlemek için ağ bileşenleri standart bir arayüz ile (HTTP) iletişim kurar ve kaynakların representationlarını değiştirir.

Bir kaynakla etkileşime geçen uygulamanın iki şey bilmesi yeterlidir: kaynağın tanımı ve bu kaynağa ne yapılması istendiği. Cache, Tunnel, Proxy vs gibi bilgileri bilmesine gerek yoktur. Sadece dönen veririnin türünü bilmesi (XML, RDF vs.) yeterlidir.

RESTful Web Servisi

RESTful web servisi, HTTP ve REST prensibi kullanılarak oluşturulmuştur.

Özellikleri:

- Kaynaklara URI kullanılarak erişilir. Örneğin, http://ornek.com/kaynak/
- Bazı veri türlerini destekler (XML, RDF, JSON gibi).

- HTTP metotlarını kullanarak işlemleri gerçekleştirir (POST, PUT, GET, DELETE).

Çizelge 6'da örnek RESTful sorguları yer almaktadır.

Çizelge 6 - Örnek RESTful Sorguları

GET http://ornek.com/kaynak/
Bu kaynağa ait olan bütün URI'ları ve varsa başka özelliklerini listeler
PUT http://ornek.com/kaynak/?{id}={value}&{key}={value}
Kaynağı başka bir kaynakla değiştirir.
POST http://ornek.com/kaynak/?{key}={value}
Kaynakta yeni bir veri oluşturur.
DELETE http://ornek.com/kaynak/{id}/
Bütün kaynağı siler.

RESTful mimarisi bir protokoldür. RESTful Web Servislerinin, SOAP'tan farklı olarak, resmi bir standardı yoktur. REST bir standart olmamasına rağmen HTTP, URL, XML vs. gibi standartları kullanır [6].

3.2. XML: Veri Protokolü

XML (Extensible Markup Language) makineler ve insanlar tarafından okunabilecek dokümanlar tasarlamaya yarayan bir W3C tarafından tasarlanmış bir standarttır. XML'in tasarım amacı internet üzerinde basitlik, genellik ve kullanılabilirlik sağlamaktır. Text veri formatındadır ve Unicode desteği vardır. Web servislerinde ve diğer programlarda geniş bir kullanım alanı vardır.

XML dosyası gündelik kullandığımız karakterlerden oluşur. <ornek>, </ornek> veya <ornek/> şeklinde etiket yapısı vardır. Örnek bir XML dosyası;

```
<?xml version="1.0" encoding="UTF-8" ?>
<painting>
  <img src="madonna.jpg" alt='Foligno Madonna, by Raphael'/>
  <caption>This is Raphael's "Foligno" Madonna, painted in
    <date>1511</date>–<date>1512</date>.
  </caption>
</painting>
```

3.3. Mobil Platformlar

Symbian

Symbian OS, Nokia'nın mobil cihazlar ve akıllı telefonlar için geliştirdiği, kütüphaneler, kullanıcı arayüzleri, çerçeveler ve birçok yaygın araç ile ilişkili, Symbian Ltd. tarafından geliştirilmiş bir mobil işletim sistemidir. Temeli Psion'un EPOC adlı yazılıma dayanır ve yalnız ARM işlemci ile çalışır.

Symbian, diğer işletim sistemlerinde olduğu gibi çoklu işlem ve korumalı bellek özelliklerine sahiptir. Symbian OS, üç sistem tasarım prensibi ile oluşturulmuştur;

- Bütünlük ve kullanıcı verilerinin güvenliği en üst seviyededir.
- Kullanım kolaylığı sağlanmalıdır.
- Kaynakları verimli kullanmalıdır.

Symbian servislere iste-çağır yaklaşımını kullanan bir mikrokernel kullanır. Bu da kullanıcı arayüzü ve motor'u birbirinden ayırmış olur. Symbian, pil ile çalışan cihazlar için ve ROM tabanlı sistemler için optimize edilmiştir. İşletim sisteminin kendisi ve uygulamalar nesne-tabanlı tasarım ilkesi (Model View Controller) ile tasarlanır. Symbian, kaynakların kullanımında verimliliği ön planda tutar. Örneğin, arka planda çalışan bir uygulama için kullanılan işlemci düşük seviyededir. Bu işlem "Aktif Nesne" adı verilen bir programlama tabiri ile gerçekleştirilir. Ayrıca işletim sistemi iş parçacığı ve "process"'leri kullanır.

Symbian Sistem Modeli aşağıdaki katmanları içerir. En üstten aşağıya doğru;

- UI Çerçeve Katmanı

- Uygulama Servis Katmanı
    - Java ME
- OS Servis Katmanı
    - Genel OS Servisi
    - İletişim Servisi
    - Çoklu Ortam ve Grafik Servisi
- Bağlantı Servisi
- Temel Servis Katmanı
- Kernel Servisleri ve Donanım Arayüz Katmanı

Temel Servis Katmanı kullanıcı taraflı işlemlerde erişilen en alt katmandır. Dosya sunucusu ve kullanıcı kütüphanesini, bütün plug-in'leri yöneten bir plug-in çerçevesi, veri tabanı yönetim sistemi, kriptografik servisi gibi servisler barındırır.

Ayrıca Symbian mikrokernel mimarisine sahiptir. Bu, en az ihtiyaçla dayanıklılık, erişilebilirlik ve hızlı yanıt gibi özelliklerde maksimum verim sağlar. İçeriğinde zamanlayıcı, hafıza yönetimi ve donanım sürücüleri vardır. Network, telefon ve dosya sistem desteği ise OS Servis Katmanına yerleştirilmiştir.

Symbian özelikle bellek kartları gibi diğer cihazlarla uyumluluk gösterecek şekilde tasarlanmıştır. Üç çeşit ağ ve iletişim alt sistemi vardır. ETEL (EPOC telefon iletişimi), ESOCK (EPOC soket) ve C32 (seri iletişimden sorumlu)'dir. Bununla birlikte çok sayıda kullanıcı arayüzü kodu vardır. Alt sistem Bluetooth, IrDA ve USB gibi kısa mesafe iletişim linklerini destekleyen kodları içerir. Sadece temel sınıflar ve altyapı Symbian OS içinde barınır. Diğerleri 3. parti yazılımlardan yapılır.

Bütün yerel Symbian C++ uygulamaları üç yapı üstüne kuruludur: Uygulama mimarisi, uygulama sınıfı, doküman sınıfı ve uygulama kullanıcı arayüzü sınıfı. Bu sınıflar temel uygulama davranışlarını oluştururlar. Geriye kalan gerekli fonksiyonlar, uygulama görünümleri, veri modeli ve veri arayüzü bağımsız oluşturulmuşlardır.

Java ME, SyncML gibi bu sistemde çalışan başka API'lerde vardır. Bu yapılardan beklenen Symbian'a üçüncü parti yazılım olarak destek sağlamalarıdır. Bunun avantajı birçok telefon modelinde aynı fonksiyonelliği sağlayabilmesidir. Ama bunun yanında telefon üreticileri de telefonlarını Symbian OS'e göre bütünleştirmeleri gerekir.

Symbian'da çalışan Java ME uygulamaları, Sun Java Wireless Toolkit gibi standart teknik ve araçlarla geliştirilirler. JAR dosyası şeklinde paketlenirler. CLDC ve CLC uygulamaları NetBeans ile oluşturulur.

Symbian platformu, açık kaynak işletim sistemi ve akıllı telefonlar için yazılım platformudur. Symbian vakfı tarafından desteklenir. Symbian platformu Symbian OS'in yerine geçmiştir ve resmi olarak Şubat 2010'da açık kaynak olmuştur. Symbian platformu Nokia, NTT DoCoMo, Sony Ericsson ve Symbian Ltd.'nin gibi firmaların yazılım alanındaki ortaklığı sonucu ortaya çıkmıştır. Resmi yayın tarihi olan 2009'dan bu yana Symbian platformu topluluk öncüsü olan Symbian vakfı tarafından geliştirilmeye devam etmektedir.

Symbian vakfı ilk olarak 2008'de duyuruldu. Bu vakfın amacı Eclipse Public License (EPL) altında bütün kaynağı açmaktı. Fakat 3. parti yazılımlar yüzünden bu işi daha az açık olan Symbian Foundation License (SFL) altında yayınladı ve sadece bu vakıfa üye kuruluşlar ile paylaştı. Daha sonra 4 Şubat 2010'da 3. parti yazılımlarda dahil bütün kaynağı EPL altında açık kaynak olarak yayınladı [4].

Symbian ilk ortaya çıktığında avkon adında doğal grafik aracını kullandı. Avko'nun gelişim sürecinde dokunmatik ekranlar henüz pratik olarak yoktu. Bu yüzden bütün grafik sisteminin idare etmek klavye üzerinden yapılıyordu. Bu grafik sistem Symbian^3 ile beraber son defa kullanıldı.

Symbian^4 ile Orbit adında dokunmatik arayüzle desteklenmiş Qt tabanlı GUI kütüphanesi kullanılmaya başlandı. Buna ek olarak, bazı saf-Qt uygulamaları hiç değiştirilmeden veya çok az değişiklik ile çalıştırılabiliyordu. Orbit Qt'nin sahip olduğu özelliklerin hepsine sahiptir.

Symbian^3 ve önceki sürümler WebKit tabanlı tarayıcılar kullanıyorlardı. Ayrıca WebKit'i ilk kullanan mobil platform Symbian'dır. Symbian^4'de ise WebKit ve Qt tabanlı bir tarayıcı kullanılmıştır.

Android

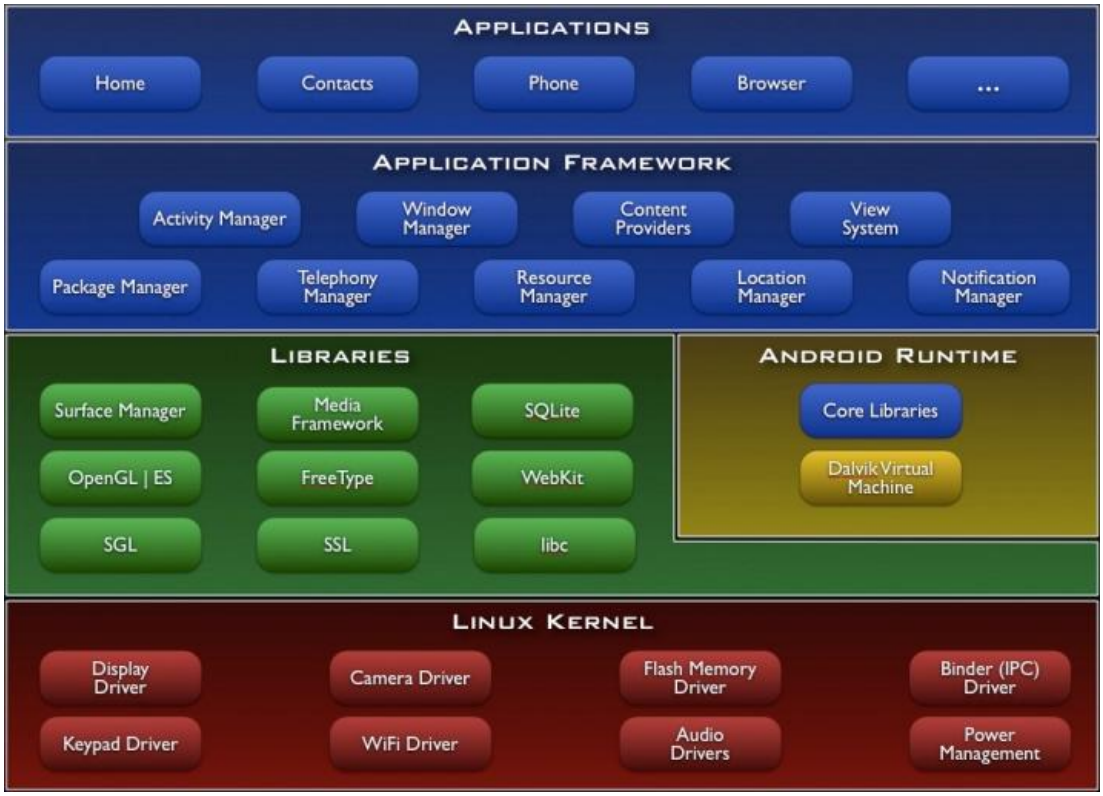
Android, mobil cihazlar için geliştirilmiş, işletim sistemi, arakatman ve anahtar uygulamaları barındıran bir yazılım kümesidir [8]. Android SDK, yazılım geliştirmek için gerekli araç ve API'leri sağlar. Yazılım geliştirme Android platformu üzerinde Java dilinde gerçekleştirilir.

Uygulama Framework, tekrar kullanım ve bileşenlerin değiştirilebilmesini sağlar.

Dalvik sanal makinesi mobil cihazlar için optimize edilmiştir. Bütünleşik tarayıcı, açık kaynak WebKit motoru kullanılarak geliştirilmiştir. İyileştirilmiş grafikler, 2B grafik kütüphanesi, OpenGL ES 1.0 tabanlı 3B grafikler ile desteklenmiştir (Donanım desteği isteğe bağlıdır). SQLite ile yapısal veri depolama kullanılmıştır. Birçok ses, görüntü ve resim formatları desteklenmektedir (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF). GSM desteği vardır (Donanım bağımlıdır). Kamera, GPS, pusula, ve ivmelendirici desteği vardır (Donanım Bağımlıdır).

Şekil 1'de Android işletim sistemin başlıca bileşenlerini göstermektedir.

Uygulamalar: Android, e-posta istemcisi, SMS programı, takvim, harita, tarayıcı, kişiler vs. gibi ana uygulamaları içinde barındırır. Bütün uygulamalar Java ile yazılmıştır.



Şekil 1 - Android Bileşenleri [8]

Uygulama Çerçevesi (Application Framework): Açık kaynak geliştirme platformunun sağlanmasıyla, Android geliştiricilerinin, çok zengin ve yenilikçi uygulamaları geliştirmelerinin önünü açar. Geliştiriciler, cihaz donanımı, koordinat bilgilerine erişimi, arka plan servislerini, alarm ayarlamalarını, durum çubuğuna bildiri eklemelerini vs. gibi birçok özelliği özgürce kullanmalarını sağlar.

Geliştiriciler, temel uygulamalar tarafından kullanılan API'lere tam yetki ile erişebilmektedirler. Uygulama mimarisinin tasarımı bileşenleri tekrar kullanabilmek için basitleştirilmiştir. Her uygulama kendi becerisini diğer uygulamalar ile belli güvenlik önlemleri altında paylaşabilir, başka uygulamaların özelliklerine erişebilir. Aynı mekanizma bileşenlerin kullanıcılar tarafından kolaylıkla yer değiştirebilmesini sağlar. Bütün uygulamalar bir takım servis ve sistemlerden oluşur;

Zengin ve geniş "View"'ler uygulama geliştirmede kullanılabilir. Örneğin, List, grid, text box, buton, web tarayıcı gibi. İçerik Sağlayıcılar, uygulamanın başka uygulamalardan veri alabilmesini ve paylaşabilmesini sağlar. Kaynak Yöneticisi,

resimler ve arayüz tasarım dosyaları kod gerektirmeyen kaynakları yönetir. Uyarı Yöneticisi, uygulamaların durum çubuğunda isteğe uyarlanmış bildirileri yayınlayabilmesini sağlar. Faaliyet Yöneticisi, uygulamaların yaşam döngüsünü ve geri plan yönetimi ile ilgilenir. Android'in bazı bileşenleri C/C++ kütüphanelerini kullanır. Android'in bu özelliği geliştiricilerin karşılarına bazı framework'lerde çıkar. Bazı temel kütüphaneler şunlardır;

C Sistem Kütüphanesi: Gömülü Linux tabanlı cihazlar için özel ayarlanmış BSD tabanlı C sistem kütüphanesi.

Ortam Kütüphaneleri: PacketVideo'nun OpenCORE yazılımı tabanlı, popüler ses ve video biçimlerini kayıt edebilen ve oynatabilen yazılımları içerir. Bazı ortam türleri, MPEG4, H.264, MP3, AAC, AMR, JPG ve PNG'dir. Yüzey (Surface) Yöneticisi: Görüntü alt sistemine ve bileşik 2B ve 3B grafik katmanlarına erişimi kontrol ediyor. LibWebCore: Android web tarayıcısını destekleyen modern web tarayıcı motoru. 3B Kütüphaneleri: OpenGL ES 1.0 API tabanlı gerçekleştirimidir. Donanım desteği (eğer varsa) ile veya yüksek optimizasyonlu 3B görüntü yazılım kullanır. FreeType: Bitmap ve vektör fontu oluşturur. SQLite: Uygulamalar için güçlü ve az yük getiren ilişkisel veritabanı motorudur.

Android Yürütümü (Anroid Runtime): Android, Java programlama dilindeki temel kütüphanelerin çoğu fonksiyonelliğini geliştiricilere sağlar.

Her Android uygulaması Dalvik sanal makinesinin bir örneği olarak kendi süreçinde (process) çalışır. Dalvik bir cihazda birçok sanal makinenin aynı anda verimli çalışacak şekilde yazılmıştır. Dalvik sanal makinesi hafızada minimum yer kaplayacak şekilde optimize edilmiş olan Dalvik Executable (.dex) formatındaki dosyaları yürütür. Sanal makine kayıt-tabanlıdır. Java derleyicisi ile derlenmiş ve dx araçları ile ".dex" formatına dönüştürülmüştür.

Dalvik sanal makinesi threading ve alt-seviye hafıza yönetiminin olduğu Linux kernelinde bulunur.

Linux Kernel: Android, güvenlik, hafıza yönetimi, işlem yönetimi, ağ yığıtı, ve sürücü modeli gibi temel sistem servislerinin olduğu Linux v2.6'da bulunur. Kernel, yazılım ve donanım arasında soyut bir katman gibi davranır.

3.4. Java ME: Mobil Uygulama Platformu

Java ME Platformuna Genel Bakış

Java ME teknolojisi ilk olarak küçük cihazlarda, belli kısıtlamalar getirilmiş şekilde, uygulama geliştirmek üzere tasarlanmıştır. Bu amaçla, Sun kısıtlı ortamlar için Java ME teknolojisinde temelleri belirlemiş ve sınırlı hafıza, görüntü ve pil kapasitesi olan küçük cihazlarda Java uygulamalarını çalıştırmayı mümkün kılmıştır [15].

Java ME Platformu, özel cihaz veya piyasa isteklerini karşılamak için bütün bir Java çalışma-zamanı ortamını teknolojilerin bir araya getirip özelleştirilmesiyle oluşturulmuştur. Bundan dolayı, platform son kullanıcıya esneklik ve uyumluluk sağlar.

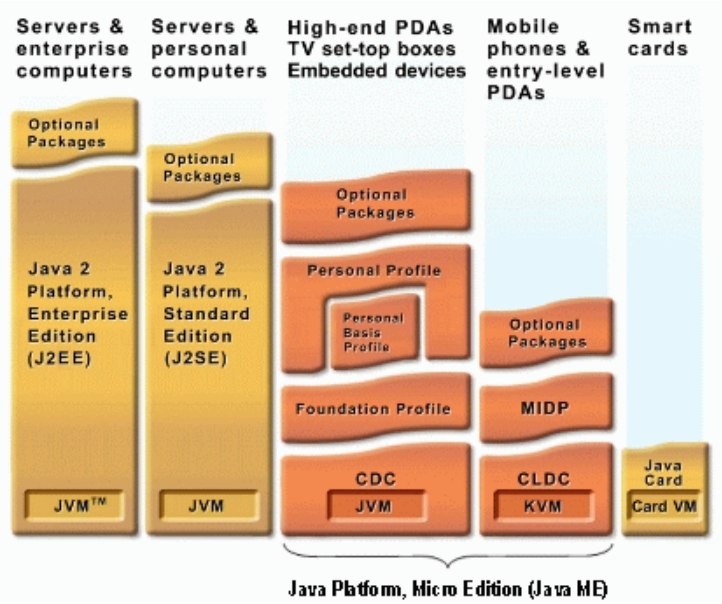
Java ME teknolojisi üç temel özelliğe dayanır:

- Birçok cihaz için en temel kütüphaneleri ve sanal makine özelliğini sağlayacak ayarlar içerir,
- Az sayıda cihazı destekleyen bir takım API'lerden oluşan profili vardır.
- Teknolojiye özel API'lerden oluşan bir seçeneksel paketi vardır.

Zaman içinde Java ME platformu iki temel yapılandırmaya ayrılmıştır. Birincisi küçük cihazlara uygunluk, ikincisi akıllı telefonlar gibi donanım özellikleri daha iyi olan cihazlar için daha geniş özelliklerdir.

Küçük cihazlar için olan yapılandırma Bağlanmış Sınırlı Cihaz Yapılandırması (Connected Limited Device Configuration – CLDC), akıllı telefonlar gibi daha donanım özellikleri daha güçlü olanlar için ise Bağlanmış Cihaz Yapılandırması'dır (Connected Device Configuration – CDC).

Şekil 2'de Java ME teknolojisinin bileşenlerini ve diğer Java teknolojileriyle olan ilişkileri gösterilmektedir.

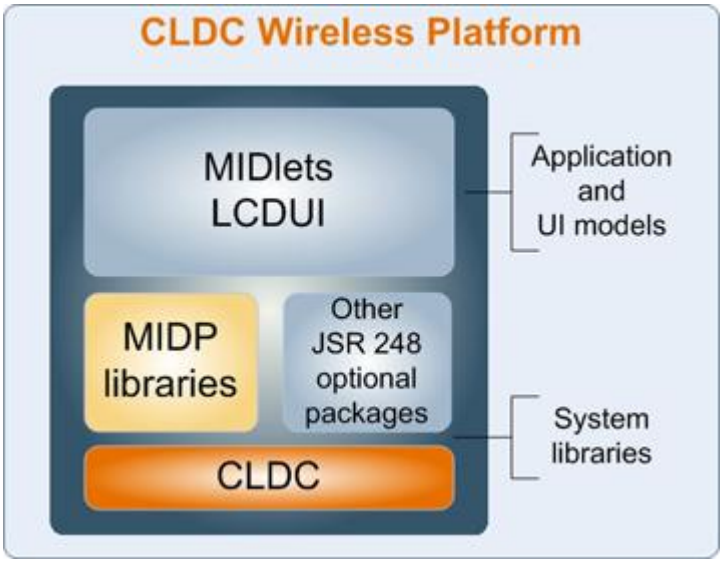


Şekil 2 - Java ME Bileşenleri [13]

Küçük Cihazlar İçin Yapılandırma (CLDC)

CLDC yapılandırması sınırlı kaynaklı olan cihazlar için tasarlanmıştır. Bu yapılandırma özellikle küçük hafızalı mobil cihazlarda güç ve grafiksellik gerektiren işlemlerdeki ihtiyaçları karşılamak için tasarlanmıştır. Ayrıca Java ME platformu uygulama tanımı için üst seviye farklı API'leri tanımlayan birtakım profilleri belirler. Şekil 3'te de gösterildiği üzere, mobil cihazlar ve sınırlı kapasitedeki diğer cihazlar için CLDC'nin MIDP (Mobile Information Device Profile) ile birlikte Java uygulama ortamı hazırlanmasını sağlamasıdır.

Bugünkü cihazların çoğunda CLDC ve MIDP ortamının implementasyonu gerçekleştirilmiş ve MIDlet'ler oluşturulmuştur. MIDlet, Java ME ile oyun, iş, vs gibi alanlar için geliştirilmiş uygulamadır. MIDlet, bir kere yazılıp derlendikten sonra Java ME teknolojisini barındıran bütün ortamlarda çalışabilmektedir.



Şekil 3 - CLDC Kablosuz Platformu [13]

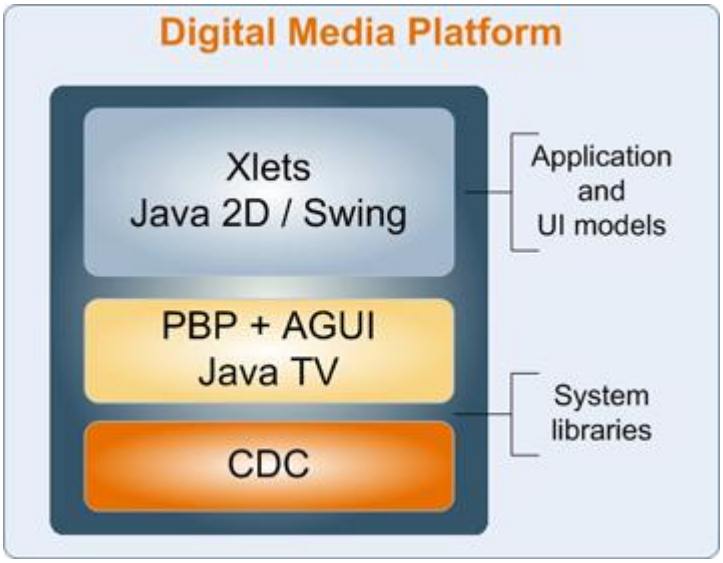
Akıllı Telefon ve Benzeri Cihazlar İçin Yapılandırma (CDC)

CDC yapılandırması, ağ bağlantısına sahip akıllı telefonlar ve set-üstü cihazlar için tasarlanmıştır. CDC yapılanmasının amacı, Java SE temelli geliştirme araçları ile teknolojik yetenekleri geliştirmek ve geniş kapsamlı bağlı aygıtların kaynak kısıtlarına uygun davranarak onların özelliklerini desteklemektir. Şekil 4'te CDC'nin uygulama ve diğer kütüphanelerle olan ilişkisi gösterilmiştir. CDC yapılandırmasının yararlarına bakacak olursak;

- Mobil olarak müşterilere, çalışanlara ve ortaklara ulaşabilmek için kullanılan ağ tabanlı uygulamaların kurumsal yararları,
- Java teknolojisinin bütünlüğü ve güvenilirliği,
- Geliştiriciler için Java platformunun zengin API'si ile güvenli ve üretkenlik sağlayan Java programlama dili ile geliştirilen uygulamalardır.

CDC yapılandırmasında tanımlanmış üç farklı profil vardır;

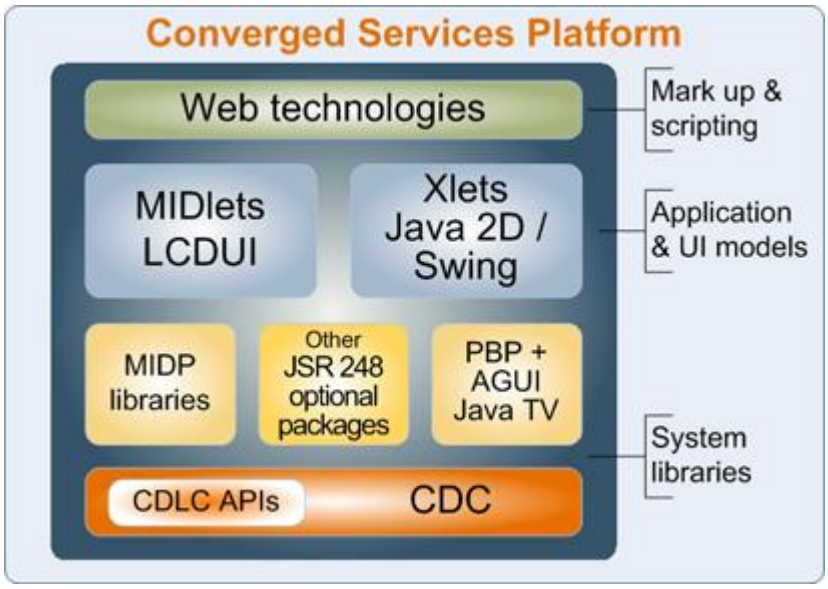
- Vakıf (Kurum) Profili (JSR 219)
- Kişisel Temel Profil (JSR 217) ve
- Kişisel Profil (JSR 216)



Şekil 4 - Dijital Ortam Platformu [13]

Birleşik Servisler İçin Java Platformu

Java ME platformu sınırlı internet bağlantısı olan mobil cihazlardan yetenekli çevrimiçi mobil cihazlara kadar her şeyi kapsar. Platformun tasarımı servislerin ihtiyacı olan esnekliği ve verimliliği sağlar. Servisler kolaylıkla başka yapılandırmaya veya profile aktarılabilir. Şekil 5'te CLDC ve CDC kütüphanelerinin birleşimini ve bunların diğer katmanlarla ilişkisini göstermektedir.



Şekil 5 - Servis Platformu [15]

4. MoReCon: MOBİL UYGULAMALAR İÇİN BAĞLAM DUYARLI ARAKATMAN YAZILIMI

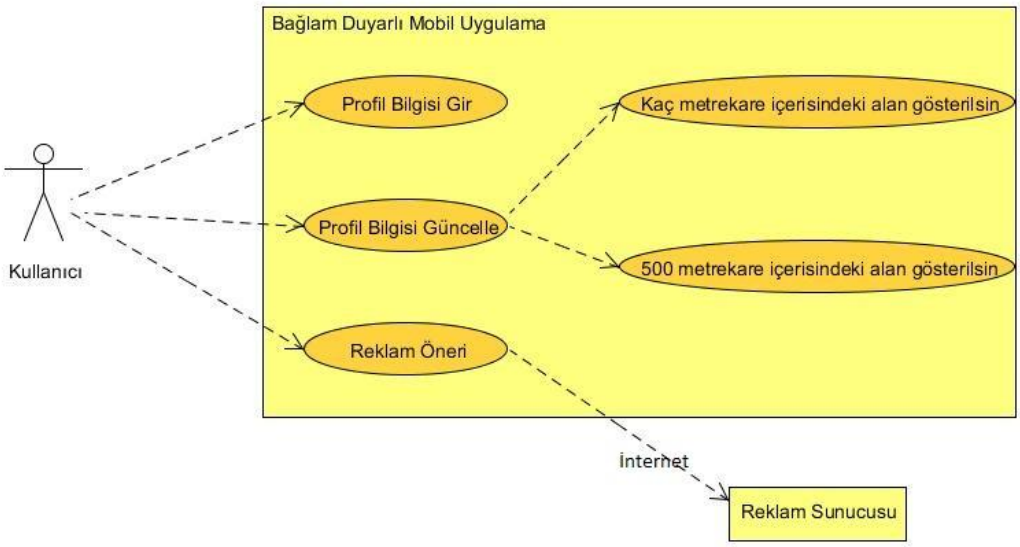
MoReCon (**Mo**bile **Re**stful **Con**text-Aware Middleware), bağlam-duyarlı mobil uygulamaların daha kolay geliştirilmesi ve bağlam (context) bilgilerinin daha kolay yönetimi için geliştirilen servis-tabanlı bir arakatman yazılımdır. Bu arakatman J2ME'de uygulama geliştirmek isteyecek olan geliştiricilere RESTful arayüzü ile bağlam türleri oluşturmak ve bunlar üzerinde ekleme, çıkarma, güncelleme ve çağırma gibi işlemler kullanarak bağlam bilgisi oluşturmalarına olanak tanır.

4.1. Motive Edici Bir Uygulama

Bir kullanıcı kişisel tercihlerine göre reklam önermeleri sunan mobil bir uygulama kullanmaktadır.

Kullanıcı bir gün canı sıkılıp arabasıyla dolaşmak istemiştir. Arabasına binip dolaşırken mobil cihazından kendisine 500m. uzaklıkta ki bir benzinlikte kampanya olduğunu öğrenir. Zaten, benzini azalmış olan aracına benzin almayı düşünen kullanıcı böylelikle hem benzinini almış olacak hem de hediyesini alacaktır. Kullanıcı o benzinliğe doğru yola koyulur ve benzinlikten hem benzini hem de hediyesini almıştır. Bu olay kullanıcıyı mutlu etmiştir. Mutlu olan kullanıcı daha sonra kıyafet almak istemektedir. Bunun için en uygun yerin alış veriş merkezi olduğunu düşünür ve hemen yola koyulur. Alışveriş merkezine varan kullanıcı mağazaları gezerken ne alacağına karar verememiştir. Tam o sırada mobil cihazından kullanıcının sevdiği bir mağazanın tüm ürünlerde %40 indirimde olduğunu öğrenir ve hemen o mağazaya giderek kendisine kıyafet alır. Fakat bu kadar hareket kullanıcıyı hem yormuş hem acıktırmıştır. Alışveriş merkezinin yemek katına doğru çıkan kullanıcı etrafına bakarak ne yiyeceğine karar vermeye çalışır. Kullanıcı en hesaplı yiyeceğin hangisi olduğunu öğrenmek istemektedir ama o kadar dolaşacak hali yoktur. Mobil uygulamasından etrafında kampanyalı yiyecek satan restoranlara bakar. En sonunda "Genç Turkcell" kampanyası yapan bir restorandan yemek

almaya karar verir. Kullanıcı yemeğini yerken mobil cihazından bu hafta vizyona yeni girmiş olan bir bilim-kurgu filminin ve bu filme akşam 19:00'a kadar %50 fiyatına girebileceği uyarısı gelir. Kullanıcımız bilim-kurgu filmlerinin fanatiği olduğundan ve daha saatin 17:00 olmasından dolayı bu teklifi geri çevirmez. Kullanıcı, hemen gider biletini alır, filmi izler ve evine döner. Kullanıcı günü kendi ilgi alanına göre önermeler yapan bir uygulama sayesinde gerek fiyat olarak indirimden gerekse kampanyalardan yararlanmıştır.



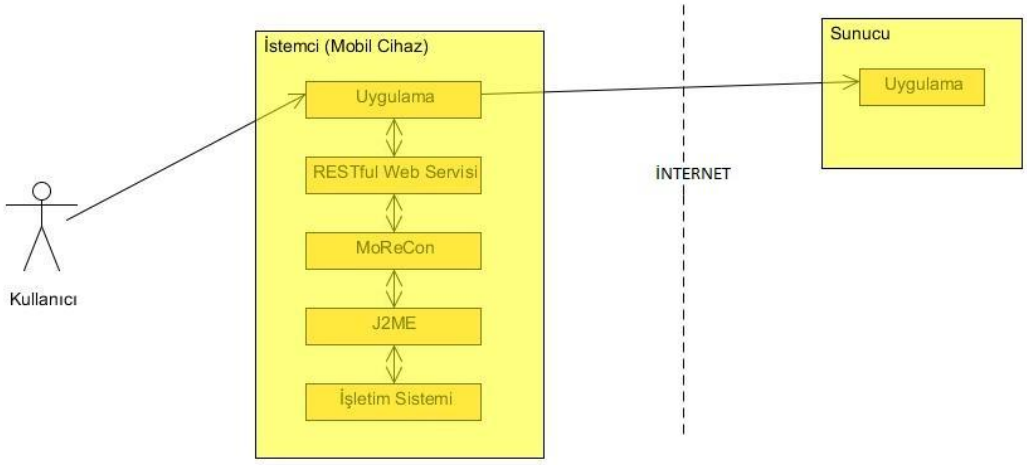
Şekil 6 - Kullanım-Şekli (Use-Case) Şeması

Bu tez kapsamında geliştirilen arakatman kullanılarak bir bağlam duyarlı reklam uygulaması oluşturulmuştur. Bu uygulama istemci ve sunucu olarak iki kısımdan oluşturulmuştur (Şekil 6).

Sunucu kısmında belirli firmalara ait kampanya, adres gibi bilgiler yer alır. Bu bilgiler MySQL veritabanında [22] saklanır. Bu bilgilerin sunumu ise RESTful web servisleri aracılığı ile XML verisi olarak sunulur. Bu bilgilere ulaşmak için kullanılan örnek URL şu şekildedir:

```
http://MoReConRESTfulServer/resources/generic?latlon=39.9207700,32.8541100&interest=Alisveris
```

Bu URL ile kullanıcının enlem, boylam bilgileri ile ilgi alanına ait bilgiler gönderiliyor. Daha sonra, sunucu tarafında bu bilgelere bakılarak elde veriler filtre ediliyor ve cevap XML olarak mobil cihaza gönderiliyor.



Şekil 7 – MoReCon Katmanlı Mimarisi

İstemci tarafı J2ME ile geliştirilmiş bir uygulamadır (Şekil 7). Bu uygulama ile:

- Profil bilgisi oluşturulup düzenlenebilir,
- Kullanıcının koordinat ve profilindeki "ilgi alanı" bilgilerine göre sunucuya bağlanıp ilgili reklamları alır ve kullanıcıya gösterir.

Mobil cihazdaki veri ekleme çıkarma işlemleri yerel web servisi üzerinden gerçekleştirilir. Örnek bir URL verecek olursak:

```
/profile/?name=Onur&surname=Soyer&age=27&mail=osoyer@etu.edu.t
r&location=Ankara&interest=Kitap,Sinema
```

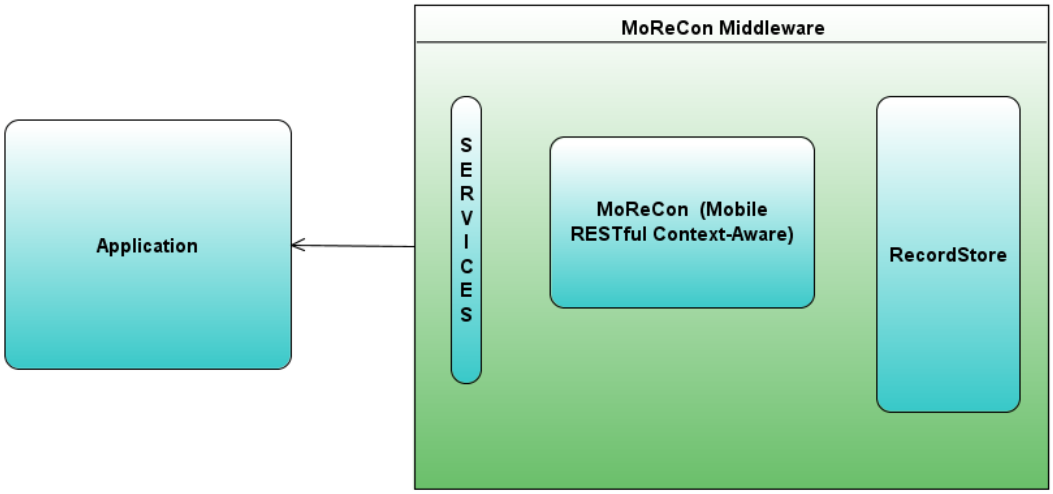
Bu URL'yi kullanarak "profile" adında yeni bir "Bağlam" türü oluşturulmuş oldu. Oluşturma işlemi sırasında bu türün adını taşıyan yeni bir kayıt açılır ve dosya

sistemine kaydedilir. “adi, soyadi, yas, cinsiyet” bu türün özellikleridir. Daha sonra yeni veri eklenirken bu kısıtlara bakılarak kullanıcının başka anahtar kullanması, eksik veya fazla anahtar girilmesi önlenecektir.

Örnekte de görüldüğü gibi arakatmana karmaşık yollardan değil sadece URL ile bağlanılmıştır.

Uzak sunucu ile iletişim HTTP GET komutu ile sağlanır. Yukarıda da bahsettiğimiz gibi sunucunun gönderdiği XML dokümanındaki etiketler işlenir. İşlenen veri daha sonra kullanıcıya gösterilir.

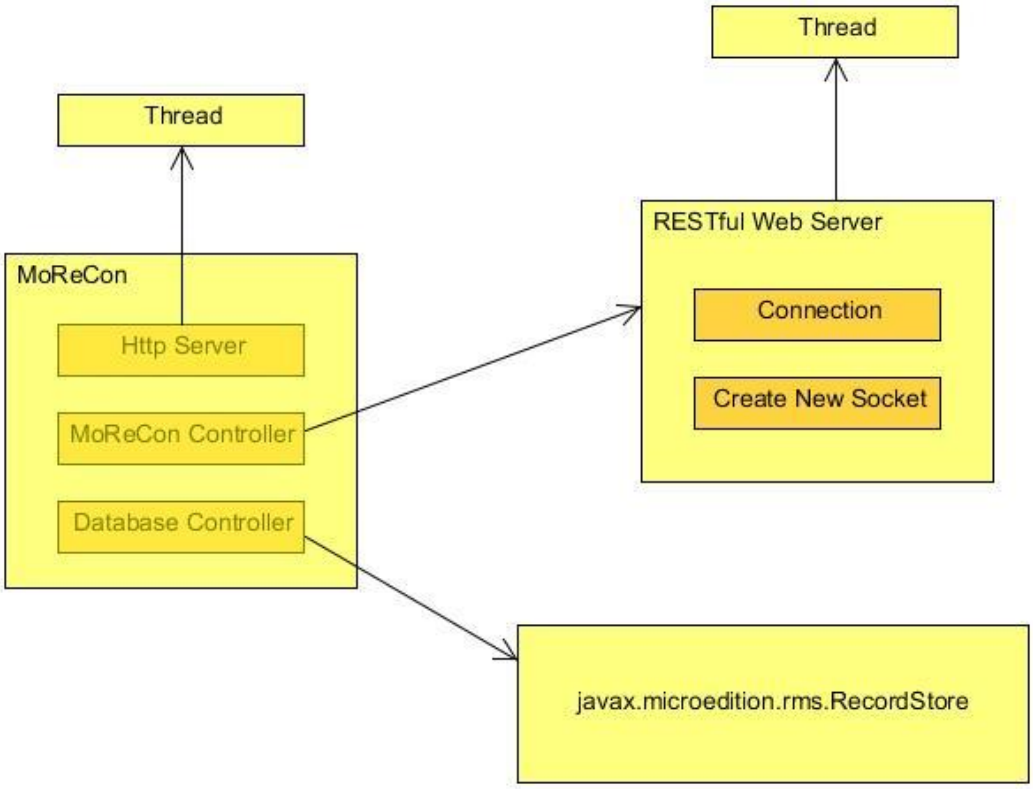
4.2. Bağlam Duyarlı Mobil Uygulama Mimarisi



Şekil 8 - Uygulama Mimarisi

Mimari, bu tez kapsamında oluşturulan ara katman ve reklam öneri uygulamasından oluşmaktadır (Şekil 8). Ara katman, uygulama ile cihaz arasındaki iletişimi sağlama görevi görür. İçerisinde RESTful web servisi, veri kayıt sistemi, tanımlanmış diğer web servislerinden veri çeken bir sistem ve verileri işleyebilmek için bazı araçlardan oluşmuştur. Uygulamada ise, oluşturulmuş olan bu servislerden doğrudan faydalanılmış, ek olarak herhangi bir araç yazılmamıştır.

4.3. MoReCon Arakatman Mimarisi



Şekil 9 - MoReCon Arakatman Mimarisi

MoReCon, RESTful web servisi, HTTP istemci, RecordStore yöneticisi'den meydana gelmektedir (Şekil 9).

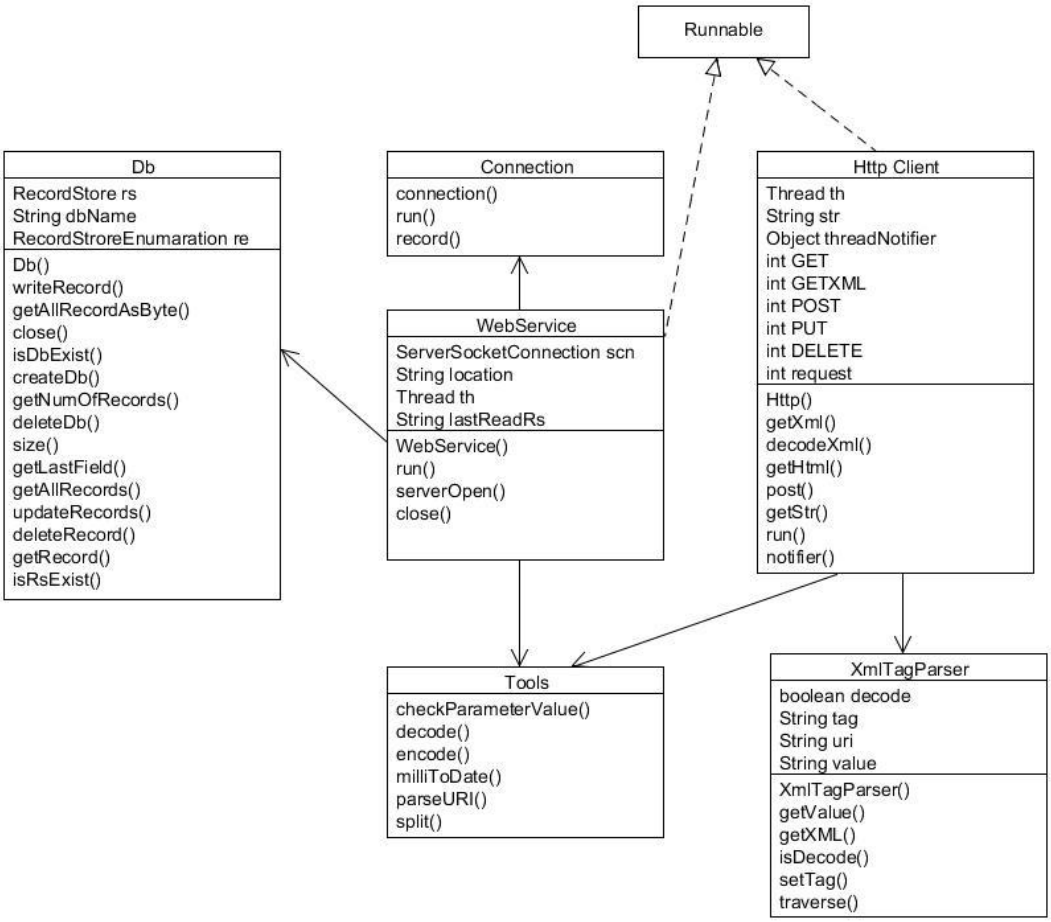
Öncelikle, kayıtlar RecordStore adlı J2ME kayıt sisteminde saklanır. RecordStore saklama işlemlerini <anahtar, değer> ikilileri şeklinde gerçekleştirir. "Anahtar", tamsayı bir değerdir. Yeni ekleme yapıldığında en son kayıtın anahtarına bakarak sonraki anahtarı oluşturur. Silme işlemi de anahtar girilerek gerçekleştirilir. Bu kayıt sistemi Kayıt Yöneticisi (Database Controller) ile yönetilir. Burada, yeni kayıt dosyası oluşturma, veri ekleme, veri silme, veri güncelleştirme, kayıtları listeleme ve bir takım kontrol metotları bulunur. Bu şekilde RecordStore'a erişim ve işlemler kolaylaştırılmıştır.

MoReCon'un temel amacı bağlam oluşturmak ve bu bağlamlar üzerinde işlemler gerçekleştirmektir. Kullanıcı istediği kadar bağlam oluşturabilir, düzenleyebilir ve silebilir. Her bir bağlam türü ayrı bir RecordStore elemanında saklanır. Saklı tutulan kayıtın ismi bağlam türünün ismi ile aynıdır. Yeni bir tür oluşturulurken karmaşıklıkları önlemek ve kontrolleri kolaylaştırmak için anahtar-değer (key-value) ilişkisine benzer bir ilişki kullanılmıştır. Kayıtın 1. satırında o bağlama ait özellikler saklanır. Geri kalan diğer satırlarda da bağlama ait değerler saklanır. İlk satırdaki anahtar bilgisi bir kere oluşturulduktan sonra güncellenemez veya silinemez. Diğer oluşturulan veriler güncellenip silinebilirler. Her bir türdeki bağlama erişmek RecordStore'ın özgün olarak atadığı kayıt numarası ile sağlanır.

MoReCon ile iletişim RESTful Web Servisleri ile sağlanır. HTTP başlıklarını ve tanımlanmış URL'leri kullanarak gerekli istekleri alır ve işler. Gelen HTTP isteğinin önce başlığı ayrıştırılır. Buna göre isteğin türü (PUT, POST, GET, DELETE) anlaşılır. Daha sonra URL'ye bakılarak isteğin ne olduğu öğrenilir ve gerekli alt sınıflara işlem yapılmak üzere yönlendirilir. Bu tanımlamalar, PUT, GET, POST, DELETE metotları altındadır. Gelen başlık ve URL okunarak bu metotlara yönlendirilir ve burada kayıt ekleme, silme, çağırma vs. gibi işlemleri gerçekleştirerek istemciye cevap gönderilir.

MoReCon, RESTful ile haberleşmeyi kolaylaştırmak veya dışarıdaki bir web servis ile haberleşme sağlamak için özel oluşturulmuş bir HTTP istemci sınıfı içermektedir. Bunun sebebi J2ME'de tanımlanmış olan sadece iki tane HTTP komutu (POST,GET) olmasıdır. Burada ek olarak tanımlanmış PUT ve DELETE metotları ile desteklenerek RESTful Web Servisleri ile haberleşmesi sağlanır.

Web Servislerinden alınan XML veriyi işleme görevi XMLTagParser adlı bir sınıf tarafından yürütülür. Bu sınıf J2ME için geliştirilmiş olan kXML adlı kütüphane ile desteklenmiştir. Gelen XML dokümanından istenilen etiket içerisindeki değeri döndürür.



Şekil 10 - UML Şeması

MoReCon sınıf diyagramı Şekil 10'da verilmiştir.

Arakatman birçok farklı sınıftan oluşmuştur. Bu sınıflar ve özellikleri şunlardır:

- WebService: Bu sınıf uygulama ile arakatman arasındaki köprüyü kurar. Oluşturulmuş olan web servisi ile POST, PUT, GET, DELETE metotları ile istekleri gerçekleştirir. Bu servis aracılığı ile yeni bağlamlar oluşturularak bu bağlamlar üzerine veri eklenir, çağırılır, düzeltilir ve silinir.

- DB: J2ME kayıt sistemi olan RecordStore ile ilgili işlemleri basite indirgeyerek veri saklama, güncelleme, çağırma ve silme işlemlerini gerçekleştirir.

- Tools: Veri işlerken kullanılan bazı araçları barındırır.

- XmlTagParser: Xml tabanlı dosyalarda belli bir etikete ulaşmak için kullanılır.

- Http: POST, PUT, GET ve DELETE metotlarını kullanarak bir web sayfası için istekler gönderir.

4.4. MoReCon RESTful Web Servis Protokolü

MoReCon web servisi, farklı bağlamları kolayca oluşturabilmeyi sağlayan bir yapıya sahiptir. Bu yapı sayesinde URL kullanarak yeni bir bağlam oluşturulabilir, oluşturulmuş bağlama veri eklenebilir, çıkarılabilir, düzenlenebilir ve silinebilir. Bu yapı sayesinde arakatman ile aradaki bağlantı esnek bir şekilde sağlanmıştır.Çizelge 7'de MoReCon RESTful web servisine yapılabilecek olan sorgu çeşitleri gösterilmektedir. Sorgular "POST", "PUT", "GET" ve "DELETE" metotları ile gerçekleşmektedir. "POST" metodu ile bir bağlam türüne ait yeni kayıt eklenmektedir. Eğer bir bağlam için ilk defa "POST" metodu çalıştırılıyorsa burada, yeni bağlam türü ve bu bağlama ait kısıtlar oluşturulmaktadır. Bağlam türü Çizelge 7'de ilk satırda görüleceği üzere {bağlam} etiketi olarak belirlenir. Daha sonra sorgunun {anahtar}'ları bu bağlamın kısıtlarını belirler. Bu kısıtlar daha sonra yeni bir kayıt ekleneceği zaman bu kayıtın doğru anahtarları kullanıp kullanmadığını kontrol amaçlı kullanılır. "PUT" metodu ise daha önceden eklenmiş olan bir kayıtı günceller. Güncelleme sorgusu, güncellenmek istenen kayıtın "id"'si ve o bağlama ait kısıtları kullanılarak gerçekleştirilir. "DELETE" metodu, "id"'si verilen bir kayıtı siler. "GET" metodunda birkaç farklı sorgu vardır. Bu sorgular ve cevaplar Çizelge 7'de gösterilmiştir. "GET" metodu ile, sistemde kayıtlı olan bağlam türlerinin listesi, bir bağlama ait bütün kayıtlar, bir bağlama ait tek bir kayıtın sonuçları ve bir bağlama ait tek bir kayıtın tek bir etiketinin sonucu elde edile bilmektedir.

Çizelge 7 - URL Yapısı

POST /{bağlam}/?{anahtar1}={değer1}&{anahtar2}={değer2}&… Sorgu ilk defa çalıştırıldığında yeni bir bağlam oluşturulur. Bağlamın kısıtları {anahtar}'lar ile belirlenir. Sonraki sorgularda bu {anahtar}'lara bakılarak sorgunun doğruluğu kontrol edilir ve doğru sütun'a veri eklenir.
PUT /{bağlam}/?id={id}&{anahtar1}={değer1}&… Önceden kayıtlı bir veriye ait bütün courier'ların değerlerini günceller.
PUT /{bağlam}/?id={id}&{anahtar1}={değer1} Önceden kayıtlı bir verinin sadece belirtilmiş anahtarının değerini günceller.
GET / Oluşturulmuş olan bütün bağlamların listesini ekrana yazdırır.
GET /{bağlam}/ Bu bağlama ait bütün kayıtları eklenme sırasına göre ekrana yazdırır.
GET /{bağlam}/{id}/ Bu bağlama ait id'si verilmiş olan kayıtı ekrana XML formatında yazdırır.
GET /{bağlam}/{id}/{anahtar}/ Bu bağlama ait id'si verilmiş olan kayıtın, belirtilmiş olan anahtarının değerini ekrana yazdırır.
DELETE GET /{bağlam}/{id}/ Bu bağlama ait id'si belirtilmiş olan kayıtı siler.

Çizelge 8'da ise HTTP metotları kullanılarak yapılmış örnek sorgular ve cevaplar yer almaktadır.

Çizelge 8 - Örnek Sorgular ve Dönen Cevaplar

POST /tv/?program=Mac&kanal=TRT&saat=12:00

HTTP 1.1 200 OK

PUT /tv/?id=12&program=Film&kanal=trt&saat=15:00

HTTP 1.1 200 OK

PUT /tv/?id=12&program=Mac

HTTP 1.1 200 OK

PUT /tv/?id=12&kanal=Star

HTTP 1.1 200 OK

GET /

```
<xml>
      <context>profil</context>
      <context>takvim</context>
</xml>
```

GET /profil/

```
<context>
      <profile id=2>
            <location>Ankara</location>
            <mail>osoyer@etu.edu.tr </mail>
            <age>27</age>
            <name>Onur</name>
            <surname>Soyer</surname>
            <interest>Sinema,müzik,tez</interest>
      </profile>
      <profile id=3>
            <location>İstanbul</location>
            <mail>test@etu.edu.tr</mail>
            <age>10</age>
            <name>Ali</name>
            <surname>Veli</surname>
            <interest>tv,yüzme</interest>
      </profile>
```

```
</context>
```

Get /profil/2/

```
<profile>
      <location>ankara</location>
      <mail>osoyer@hotmail.com</mail>
      <age>27</age>
      <name>Onur</name>
      <surname>Soyer</surname>
      <interest>Alisveris,Konser,Vizyon,Yemek</interest>
</profile>
```

Get /profile/2/name/

```
<name>Onur</name>
```

Get /profile/2/surname/

```
<surname>Soyer</surname>
```

Get /profile/2/age/

```
<age>16</age>
```

4.5. Kullanıcı Arayüzleri

Sun Emülator çalıştığında, ekrana yansıyan görüntü emülatörde yüklü olan uygulamaları gösterir. Bu uygulamalardan çalıştırmak istenen uygulamanın imleç üstüne getirilir ve "launch" butonuna basılır (Şekil 11). Burada, "MoreconApp" adlı uygulama tez kapsamında geliştirilmiş olan arakatman üstüne yazılmış bir uygulamadır. Bundan sonraki resimlerde bu uygulamanın içerdiği özellikler anlatılacak.

Şekil 11 - Uygulama Seçme Menüsü

Bu görüntü uygulamanın ana ekranıdır. Sağ alt köşede "Menü" butonundan oluşur. Bu buton ile uygulamanın alt bölümlerine ulaşılmaktadır (Şekil 12).

Şekil 12 - MoReCon Uygulaması Ana Menü

"Profile" ve "Suggest", "Menü" butona ait seçeneklerdir. "Profil", seçeneğinin üzerine gelip emülatör'ün "ok" tuşuna basarak alt menüye ulaşılır (Şekil 13).

Şekil 13 - Uygulama Ana Menü Komut Listesi

Bu ekran görüntüsü "Profil" bölümüne aittir. Bu bölümde kullanıcı ile ilgili bilgi girişi yapılabilen kutulardan oluşur. Burada girilen "interest" bilgisine göre sistem daha sonra önermelerde bulunacaktır (Şekil 14).

Şekil 14 - Profil Ana Menü

Kutulara bilgiler girildikten sonra kayıt etmek için, “Menü” butonuna tıklayarak “Save” butonuna basılır. Bu buton ilk kayıt işleminde çalışır. Bunun sebebi geliştirilen arakatmanın HTTP metotlarının kullanıldığını göstermek içindir (Şekil 15).

Şekil 15 - Profil Bilgi Kaydetme

Bu ekran görüntüsü başarılı bir kayıt sonrası gösterilir (Şekil 16).

Şekil 16 - Profil Bilgi Kaydetme - Olumlu Uyarı

Bu ekran görüntüsü daha önce bir kayıt oluşturulduğunu ve bu yüzden yeniden oluşturulamayacağını göstermektedir (Şekil 17).

Şekil 17 - Profil Bilgi Kaydetme - Olumsuz Uyarı

Bu ekran daha önce oluşturulmuş kullanıcı profil bilgisini güncellemek için kullanılan "Update" butonunu göstermektedir. Güncellemek için bu buton seçilerek mobil cihazın "ok" tuşuna basılır (Şekil 18).

Şekil 18 - Profil Güncelleme

Başarılı bir şekilde gerçekleştirilmiş güncellemenin ekran görüntüsü aşağıdaki şekildedir (Şekil 19).

Şekil 19 - Profil Update - Olumlu Uyarı

Başarısız bir güncelleme işleminin ardından gösterilen ekran görüntüsü aşağıdaki şekildedir (Şekil 20).

Şekil 20 - Profil Update - Olumsuz Uyarı

“Suggest” adlı öneri penceresine ulaşmak için uygulamanın ana menüsünde “menü” butonunun altında “Suggest” butonuna tıklanır (Şekil 21).

Şekil 21 - Suggest'e Erişim Butonu

“Suggest” penceresinin bir görüntüsü. Bu bölümde kullanıcı kaç metrelik bir alanda kendi ilgi alanına uygun önermeleri görmek istediğini belirtiyor (Şekil 22).

Şekil 22 - Öneri - Uzaklık Bilgisi Kutusu

Kullanıcının ilgi alanına bağlı olarak dönen sonuç ekranı aşağıda verilmiştir (Şekil 23).

Şekil 23 - Öneri - Kullanıcı İlgi Alanına Uygun Yanıt

4.6. Performans Analizi

Çizelge 9 – Test Bilgisayarı ve Geliştirme Ortamı Özellikleri

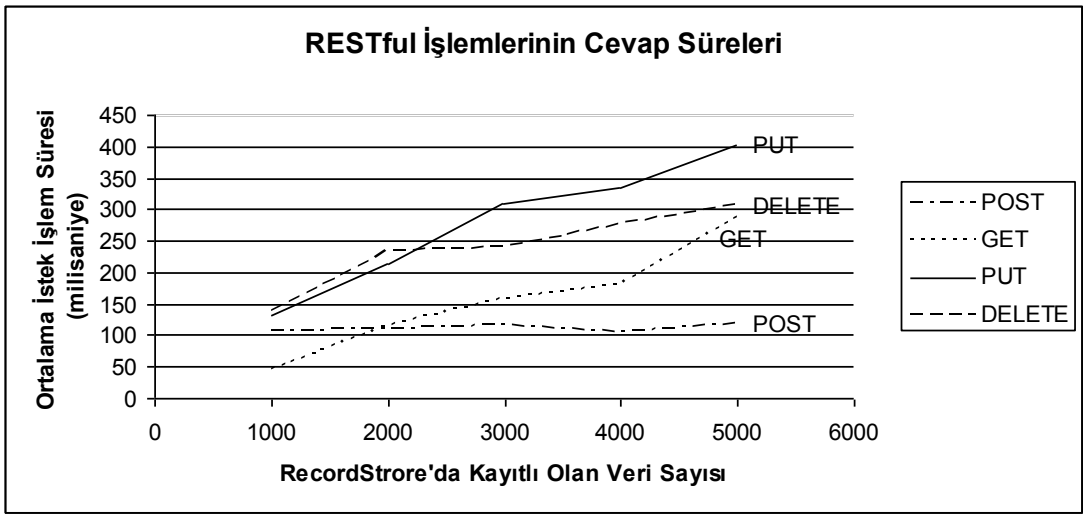
Tür	Dizüstü
İşlemci	Intel T7200 2 GHz.
Bellek	2 GB
J2ME Versiyonu	Sun Java Wireless Toolkit 2.5.2_01 for CLDC (SDK 2.5.2_01)
Geliştirme Ortamı	NetBeans 6.8

Arakatman performans başarımını test etmek için bir takım sorgular gönderilmiş ve bu sorgular yanıtlanıncaya kadar geçen süreler performans hesabında kullanılmıştır. Test için kullanılan bilgisayar ve geliştirme ortamı bilgileri Çizelge 9'daki gibidir. Test esnasında kullanılan HTTP parametreleri de Çizelge 10'da gösterilmiştir.

Çizelge 10 - Test Parametreleri

Test çalıştırma sayısı	50
Test Komutları	
GET	`/profil/1/`
POST	`/profil/?name=Onur&surname=Soyer&age=27&location=Ankara&interest=Sinema`
PUT	`/profil/?id=5&name=Onur&surname=Soyer&age=27&location=Ankara&interest=Sinema`
DELETE	`/profil/1/`

Test işlemleri için önce Şekil 24'te de gösterildiği gibi "POST" metodu ile 1000, 2000, 3000, 4000 ve 5000 verilik kayıtlar oluşturulmuştur. Eklenen her bir veri için sorgu gönderildiği andan cevabın geldiği ana kadar olan süre hesaplanmıştır. Daha sonra, hesaplanan bu sürelerin toplamının toplam veri sayısına bölünerek ortalama işlem süresi hesaplanmıştır. Veriler eklendikten sonra "GET", "PUT", "DELETE" metotları için aynı işlem tekrarlanmıştır. "GET" metodu için, oluşturulmuş verilerden rastgele "id" atayarak 50 adet sorgu yapılmış ve bu sorguların ortalama işlem süreleri hesaplanmıştır. "PUT" metodu için, rastgele "id" atayarak seçilen 50 verinin güncellemesi yapılmış ve ortalama süre bulunmuştur. Aynı şekilde "DELETE" metodu kullanılarak rastgele 50 seçilen adet "id"'nin kayıtları silinmiş ve bu işlem içinde ortalama süre bulunmuştur. Bu işlemlerin sonuçları Şekil 24'te belirtilmiştir.

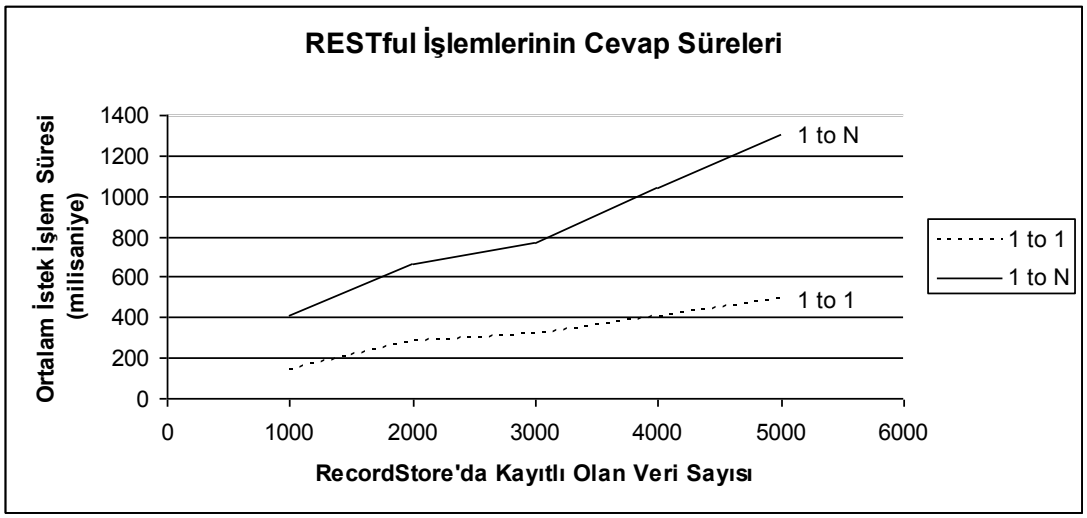


Şekil 24 - RESTful İşlemlerinin Cevap Süreleri

Şekil 24'te görüldüğü gibi sorgular arasındaki işlem sürelerinde farklılıklar gözlenmektedir. "POST" metodunun işlem süresinin sabit bir doğru çizmesinin sebebi her yeni kayıtın en sona eklenmesinden dolayıdır. Diğer "PUT", "GET" ve"DELETE" metotları için ise her bir sorgu için "id" gerekmektedir. Bu "id"'ye göre kayıt sistemi "id"'yi bulmak için kayıtları tarar. Bundan dolayı da kayıt sayısı arttıkça "id"'yi bulup işlem yapma süresi de orantılı olarak artmaktadır. Örneğin, "PUT" işlemi için sistem güncellenecek olan verinin "id"'sini RecordStore'da en baştan tarayarak kıyaslama yapar. Kıyaslama sonunda eşleşen "id"'yi günceller. Aynı şekilde, "DELETE" metodunda silinecek olan kayıtın "id"'si RecordStore'da en baştan taranır ve eşleşme olduğunda silinir. "GET" metodunda da bütün RecordStore taranır ve eşleşme olduğunda kayıt getirilir. Bunun sebebi RecordStore'un kayıtlar için index tutmamasından kaynaklanır. Grafikte tepe noktaları için aralarındaki işlem sürelerindeki farkına bakacak olursak "PUT" işlemi 400 milisaniye, "DELETE" 300 milisaniye, "GET" 280 milisaniye ve "POST" 130 milisaniye sürmüştür. "PUT", "DELETE" ve "GET" işlemlerinin hepsi kayıt sisteminde "id" taraması yapmaktadır. Fakat aralarındaki performans farkının sebebi, "PUT" işleminde veri güncelleme işlemi olduğundan dolayı "DELETE" ve "GET" işlemlerine göre daha fazla sürmekte, "DELETE" işleminde ise kayıt silme işlemi gerçekleştiği için "GET" işlemine göre daha fazla sürmektedir. Grafikte artışların

doğrusal olmamasının sebebi, geliştirme yapılan bilgisayarın o anki performansına ve emülatörün performansına bağlı olarak değişiklik göstermesidir.

Şekil 25'te yapılan bu testin sonuçları yer almaktadır.



Şekil 25 - Join Yapılmış 1-1 ve 1-N İlişkili İki RecordStore'dan HTTP GET ile Veri Çekmenin Ortalama Performans Sonuçları

JOIN için test işlemi 1000, 2000, 3000, 4000 ve 5000 verilik iki farklı bağlam oluşturularak tekrarlanmıştır. Bu bağlamlar birbirlerine "foreign key" kullanarak bağlanacak şekilde oluşturulmuştur. Oluşturulan bu veriler ile 1-1 ve 1-N ilişkili sorgular yapılmıştır. Daha sonra, oluşturulan bu veriler üzerinde rastgele 50 "id" seçilerek sorguları yapılmıştır. Sorgu işlemi için önce bir bağlama ait veri çağırılmış bu veriden diğer bağlama ait verinin "id" bilgisi elde edilmiştir. Elde edilen "id" bilgileri diğer bağlamda GET metodu ile sorgulanmıştır. Her bir sorgu için ilk yapılan sorgudan, en son gelen cevaba kadar geçen sürelerin ortalaması alınarak istek işlem süreleri hesaplanmıştır. Örnek olarak, kullanıcının arkadaş listesi ve kullanıcının ajanda bilgilerini tutacak şekilde iki ayrı bağlam olsun. Sorgu işlemi, önce kullanıcının arkadaş bilgilerini çekiyor ve daha sonra bu bilgileri kullanarak kullanıcının ajandasında bu isimlerin ilişkili olduğu etkinlikleri getiriyor.

Şekil 25'teki sonuçlara göre 1-N gerçekleştirilen ortalama işlem süresi 1300 milisaniye iken 1-1 için gerçekleştirilen işlem süresin 500 milisaniye sürmüştür. Bunun sebebi 1-N için bir bağlam verisinin ilişkili olduğu diğer bağlamın veri

sayısının 1-1'e göre fazla olmasıdır. Buna göre, yapılan sorgu sayısı ve işlem süresi daha fazla sürmüştür. Ayrıca, RecordStore'un index tutmadığını Şekil 24'ü açıklarken bahsetmiştik. Bu da, sorguların işlem sürelerindeki farkı etkilemektedir.

4.7. MoReCon Özellikleri

- **Küçük bir paket**
  MoReCon paketi 114 KB boyutundadır. J2ME tabanlı mobil cihazlarda çalışabilir.
- **Modüler yaklaşım**
  Uygulamalar, MoReCon'un sağladığı bağlam duyarlı veri yönetimi hizmetleri karmaşıklığından bağımsız olarak ayrı bir katman olarak geliştirilebilir.
- **Taşınabilirlik**
  MoReCon, J2ME özelliği sebebiyle farklı platformlara taşınabilir ve kullanılabilir.
- **Dil bağımsız uygulama geliştirmeye izin vermesi**
  MoReCon RESTful web servisleri ile erişildiği için servis tabanlı mobil mimariye izin verir. Uygulama istenilen herhangi bir dilde geliştirilebilir. Örneğin MoReCon J2ME ile geliştirilmesine rağmen, uygulama programı Windows Mobile SDK ile geliştirilebilir.
- **Uzaktan erişilebilir**
  MoReCon, RESTful web servisi ile sahip olduğu verilere uzak bilgisayar ve uygulamaların internet üzerinden erişimine izin verir, böylece dağıtık ve etkileşimli uygulamalar geliştirmek mümkündür.

5. SONUÇ VE GELECEK ÇALIŞMALAR

Bu tezde, RESTful web servisi ile haberleşen bağlam duyarlı bir arakatman geliştirilmiştir. Bu arakatman ile kolay bir şekilde bağlam oluşturulabilir ve bu bağlamlar üzerinde işlemler gerçekleştirilebilir. Bu arakatman ile geliştirilecek olan uygulamalarda kullanılacak olan bağlamlar sayesinde verilerin anlam kazanması sağlanacaktır. Ayrıca RESTful arayüzü ile oluşturulan bağlamlar cihazların haberleşmesi ile paylaşılabilir ve bu şekilde bağlam sayısı arttırılabilir.

Mobil emülator kullanarak, MoRecon arakatman ile RESTful web servisi üzerinden bağlam oluşturma, güncelleme, silme ve çağırma üzerine performans testleri yapılmıştır. Bu sonuçlara göre güncelleme ve silme işlemlerinde veri yükü arttıkça hız yavaşlamaktadır. Bunun yanında veri ekleme ve çekme işlemlerinde herhangi bir yük artışı gözlenmemiş, cihaz kararlılığını korumuştur.

Bu çalışmanın bir sonraki safhasında elde edilen bağlamların saklama ve sunma yöntemleri üzerine gerçekleştirilebilir. Verilerin makineler tarafından anlaşılıp kullanılabilmesi ve bağlamların bir otomasyon şeklinde haberleşebilmesini sağlamak amacı ile verilerin anlamsal olarak saklanması ve sorgulanması sağlanabilir. Bu şekilde, arakatman, bağlamlar için otomatik olarak ontoloji oluşturabilecek ve sorgulayabilecektir.

KAYNAKLAR

[1] A. K. Dey, G. D. Abowd, “Towards a better understanding of context and context-awareness”, Technical Report GITGVU-99-22, Georgia Institute of Technology, College of Computing, 1999.

[2] Edwin J. Y. Wei, Alvin T. S. Chan, "Semantic Approach to Middleware-Driven Run-Time Context-Aware Adaptation Decision", Proc. of the 2008 IEEE International Conference on Semantic Computing (ICSC 2008), pp.440-447, 2008.

[3] Symbian açık kaynak olarak yayınlandı. Bkz: http://www.symbian.org/news-and-media/2010/02/04/symbian-completes-biggest-open-source-migration-project-ever Son erişim: 01.07.2010

[4] Roy Thomas Fielding, Architectural Styles and the Design of Network-based Software Architectures, Ph.D. Thesis, University of California, Irvine, Irvine, California, 2000.

[5] Richardson, Leonard; Ruby, Sam, “RESTful Web Services”, O'Reilly, 2007.

[6] Maedche, A. and Staab, S., “Ontology Learning for the Semantic Web”. IEEE Intelligent Systems, 16 (2): 72–79, 2001.

[7] Berners-Lee, Tim; Fischetti, Mark, “Weaving the Web”. HarperSanFransisco, 1999.

[8] What is Android?, Bkz: http://developer.android.com/guide/basics/what-is-android.html Son erişim 02.07.2010 Son erişim 14.07.10

[9] M. van Setten, M. Pokraev, S. Koolwaaij J., “Context-Aware Recommendations in the Mobile Tourist Application COMPASS”, In Nejdl, W. & De Bra, P. (Eds.). AH 2004, LNCS 3137, Springer-Verlag, 2004.

[10] Hong-Linh Truong, Schahram Dustdar, Dino Baggio, Stephane Corlosquet, Christoph Dorn, Giovanni Giuliani, Robert Gombotz, Yi Hong, Pete Kendal, Christian Melchiorre, Sarit Moretzky, Sebastien Peray, Axel Polleres, Stephan Reiff-Marganiec, Daniel Schall, Simona Stringa, Marcel Tilly, HongQing Yu, "inContext: A Pervasive and Collaborative Working Environment for Emerging Team Forms," saint, pp.118-125, 2008 International Symposium on Applications and the Internet, 2008

[11] Lakshmish Ramaswamy, Deepak P., Ramana Polavarapu, Kutila Gunasekera, Dinesh Garg, Karthik Visweswariah, Shivkumar Kalyanaraman, "CAESAR: A Context-Aware, Social Recommender System for Low-End Mobile Devices," mdm, pp.338-347, 2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware, 2009

[12] Damiao R. De Almeida, Claudio De Souza Baptista, Elvis R. Da Silva, Claudio E. C. Campelo, Hugo F. De Figueiredo, Yuri A. Lacerda, "A Context-Aware System Based on Service-Oriented Architecture," aina, vol. 1, pp.205-210, 20th International Conference on Advanced Information Networking and Applications - Volume 1 (AINA'06), 2006

[13] JavaME Technology, bkz: http://java.sun.com/javame/technology/index.jsp#cldc Son erişim 14.07.10

[14] João Paulo Sousa, Eurico Carrpatoso, Benjamin Fonseca, "A Service-Oriented Middleware for Composing Context Aware Mobile Services," iciw, pp.357-362, 2009 Fourth International Conference on Internet and Web Applications and Services, 2009

[15] Wegdam, M. (2005) "AWARENESS: a project on context AWARE mobile NEtworks and ServiceS", 14th Mobile and Wireless Communication Summit, Dresden, Germany.

[16] Hong-Linh Truong, L. Juszczyk, A. Manzoor, and S. Dustdar, "ESCAPE – An Adaptive Framework for Managing andProviding Context Information in Emergency Situations" Smart Sensing and Context, Second European Conference, EuroSSC 2007, Kendal, England, October 23-25, Proceedings 2007

[17] Solsvik, F. D4.2.3: Final Integrated Services Design and Implementation Report, 2006

[18] iPhone bkz. http://www.apple.com/tr/iphone/ Son erişim 20.08.2010

[19] XML bkz. http://www.w3.org/XML/ Son erişim 20.08.2010

[20] JSON bkz. http://www.json.org/ Son erişim 20.08.2010

[21] Context-Awareness bkz. http://en.wikipedia.org/wiki/Context_awareness Son erişim 20.08.2010

[22] MySQL bkz. http://www.mysql.com/ Son erişim 20.08.2010

[23] NetBeans bkz. http://netbeans.org/ Son erişim 20.08.2010

[24] Sun Java (TM) Wireless Toolkit 2.5.2_01 for CLDC bkz. http://java.sun.com/products/sjwtoolkit/overview.html Son erişim 20.08.2010

[25] Hong-Linh T. , S. Dustdar, "A Survey on Context-Aware Web Service Systems", International Journal of Web Information Systems, Vol. 5, No. 1. (2009), pp. 5-31, 2009

[26] Solsvik, F. D4.2.3: Final Integrated Services Design and Implementation Report. Akogrimo, 2006

[27] Prezerakos, G. N., Tselikas, N. D., & Cortese, G., Model-driven Composition of Context-aware Web Services Using ContextUML and Aspects. 2007 IEEE International Conference on Web Services (ICWS 2007) (pp. 320-329). IEEE Computer Society, 2007

[28] Han, B., Jia, W., Shen, J., & Yuen, M.-C. (2008). Context-Awareness in Mobile Web Services. Parallel and Distributed Processing and Applications (pp. 519-528). Springer-Verlag, 2008

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı : SOYER, Onur

Uyruğu : T.C.

Doğum tarihi ve yeri : 30.08.1983 Ankara

Medeni hali : Bekar

e-mail : osoyer@etu.edu.tr

Eğitim

Derece	Eğitim Birimi	Mezuniyet tarihi
Lisans	Gaziantep Üniversitesi/Fizik Mühendisliği	2007

Yabancı Dil

İngilizce