ACİL DURUMLARDA İÇ MEKANLARIN TAHLİYESİNE YÖNELİK 3B CBS TABANLI AKILLI MOBİL NAVİGASYON SİSTEMİNİN GELİŞTİRİLMESİ

2017
DOKTORA TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ

Yasin ORTAKCI

ACİL DURUMLARDA İÇ MEKANLARIN TAHLİYESİNE YÖNELİK 3B CBS TABANLI AKILLI MOBİL NAVİGASYON SİSTEMİNİN GELİŞTİRİLMESİ

Yasin ORTAKCI

Karabük Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalında
Doktora Tezi
Olarak Hazırlanmıştır

KARABÜK
Mart 2017

Yasin ORTAKCI tarafından haz lanan "ACİL DURUMLARDA İÇ MEKANLARIN TAHLİYESİNE YÖNELİK 3B CBS TABANLI AKILLI MOBİL NAVİGASYON SİSTEMİNİN GELİŞTİRİLMESİ" başlıklı bu tezin Doktora Tezi olarak uygun olduğunu onaylarım.

Doç. Dr. İsmail Rakıp KARAŞ .....
Tez Danışmanı, Bilgisayar Mühendisliği Anabilim Dalı

Bu çalışma, jürimiz tarafından oy birliği ile Bilgisayar Mühendisliği Anabilim Dalında Doktora tezi olarak kabul edilmiştir. 27/03/2017

Ünvanı, Adı SOYADI (Kurumu)	İmzası
Başkan : Prof. Dr. Bülent BAYRAM (YTÜ)	.....
Üye : Prof. Dr. Abdurrahman GEYMEN (ERÜ)	.....
Üye : Doç. Dr. İsmail Rakıp KARAŞ (KBÜ)	.....
Üye : Yrd. Doç. Dr. Hakan KUTUCU (KBÜ)	.....
Üye : Yrd. Doç. Dr. Ümit ATİLA (KBÜ)	.....

...../...../2017

KBÜ Fen Bilimleri Enstitüsü Yönetim Kurulu, bu tez ile, Doktora derecesini onamıştır.

Prof. Dr. Nevin AYTEMİZ .....
Fen Bilimleri Enstitüsü Müdürü

"Bu tezdeki tüm bilgilerin akademik kurallara ve etik ilkelere uygun olarak elde edildiğini ve sunulduğunu; ayrıca bu kuralların ve ilkelerin gerektirdiği şekilde, bu çalışmadan kaynaklanmayan bütün atıfları yaptığımı beyan ederim."

Yasin ORTAKCI

ÖZET

Doktora Tezi

ACİL DURUMLARDA İÇ MEKANLARIN TAHLİYESİNE YÖNELİK 3B CBS TABANLI AKILLI MOBİL NAVİGASYON SİSTEMİNİN GELİŞTİRİLMESİ

Yasin ORTAKCI

Karabük Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı:
Doç. Dr. İsmail Rakıp KARAŞ
Mart 2017, 98 sayfa

Günümüzde gökdelenler, hava alanları, alışveriş merkezleri gibi karmaşık ve çok katlı binaların sayısı gün geçtikçe artmaktadır. Buna paralel olarak, bu tip binalarda zaman geçiren insanların sayısı da hızla artmaktadır. Birçok insan bu tarz binalarda yönlerini kaybetmekte, aradıkları yerleri ya da çıkışı bulmakta sıkıntı yaşamaktadırlar. Bunun da ötesinde, acil durumlarda, özellikle yangın durumunda, insanların bu binalardan çıkışı çok daha zorlaşmaktadır. Yine yangından kaçış esnasında binada belirli bölgelerde yığılmalar olmakta, bu durum paniğe ve izdihama sebep olabilmektedir. Hatta bazı afetzedeler çıkış güzergâhı bulamamakta, bunun doğal sonucu olarak da hayatlarını kaybedebilmektedirler. Hazırlanan bu çalışmada hem binadaki değişen koşullar hem de kullanıcının kişisel özellikleri dikkate alınarak, gerçek zamanlı çalışan, mobil bir arayüze sahip akıllı bir tahliye sistemi geliştirilmiştir. Bu tahliye sistemi başlangıçta kullanıcıya güvenli bir güzergâh sunmaktadır. Eğer kullanıcıya

sunulan bu güzergahta kullanıcı için risk teşkil edecek bir gelişme olursa, sistem kullanıcıya alternatif ve daha güvenli bir güzergâh sunmaktadır. Bunun yanı sıra tahliye sistemi gerçek zamanlı çalışmakta, kullanıcının konumunu anlık olarak tespit edip bulunduğu konuma göre yönlendirme talimatları oluşturmaktadır. Ayrıca sistemi diğer alternatif tahliye ve yönlendirme sistemlerinden ayıran en önemli özellik, kullanıcıyı kişisel özelliklerine göre yönlendirebilmesidir. Yani tahliye sistemi her kullanıcıya özel çıkış güzergahları sunabilmektedir. Böylece kullanıcı kendisi için en uygun ve güvenli güzergahtan tahliye edilmiş olacaktır. Bu sistemde kullanıcıların konumu RFID (Radio-Frequency Identification) teknolojisi ile takip edilmektedir. Kullanıcının tahliyesi için gerekli olan yönlendirme talimatları merkezi bir sunucu üzerinde yapay sinir ağıyla desteklenmiş bir modül üzerinde hazırlanmaktadır. Bu yapay sinir ağı, sistemi akıllı bir yönlendirme sistemine dönüştürmektedir ve diğer alternatif tahliye sistemlerine üstün kılmaktadır. Sunucu üzerinde üretilen yönlendirme talimatları gerçek zamanlı olarak kullanıcının akıllı telefonuna gönderilmektedir. Sistem kullanıcıyı anlık olarak telefonu üzerinden sesli ve görsel öğeler ile çıkışa yönlendirmektedir. Geliştirilen sistem, yaygın kullanıma sahip bir tahliye sistemi olmaya aday niteliktedir.

Anahtar Sözcükler : Bina içi yönlendirme, mobil, RFID, tahliye, yangın, yapay sinir ağı.

Bilim Kodu **:** 902.1.014

ABSTRACT

Ph. D. Thesis

DEVELOPMENT OF A 3D GIS BASED INTELLIGENT MOBILE NAVIGATION SYSTEM FOR EVACUATION OF INDOOR ENVIRONMENTS IN EMERCENGY CASES

Yasin ORTAKCI

Karabük University
Graduate School of Natural and Applied Sciences
Department of Computer Engineering

Thesis Advisor:
Assoc. Prof. Dr. İsmail Rakıp KARAŞ
March 2017, 98 pages

Today the number of complex and multi-storey buildings, such as skyscrapers, airports and shopping centers, is on the increase with each passing day. In parallel to this, the number of people spending time in such buildings is also on a rapid increase. In these buildings, many people lose their direction, fail to find the places or the exits they are seeking, and thus, they end up with experiencing problems in such cases. Beyond that, it becomes much more difficult for people to exit these buildings under emergency cases, particularly in case of a fire. Congestions occur in certain spots of the building when the people are escaping from the fire, which then leads to panic and confluence. Even, some victims of disaster cannot find an escape route and lose their lives as a foregone conclusion of this situation. Within this context, a dynamic, intelligent, real-time, mobile evacuation system has been developed, which guides each one of people within the building according to their own physical features and varying conditions

within the building in the case of fire. The system produces a secure exit route for each user. If any event occurs that poses a risk for a user on his/her route, system produces an alternative and more secure route for the user. In addition to this, the evacuation system runs as a real-time manner and produces the navigation instructions by positioning the user instantly. The most important property of the system that makes it different than the other evacuation and navigation systems is the capability of navigating the user according to his/her personal features. In short, the evacuation system can produce special exit route for each user. Thus, the user will be evacuated through the most convenient and secure path from the building. In this system, the location of the users is tracked by the RFID Technology. The necessary instructions for guidance to evacuate the users are prepared on an artificial neural network-assisted module on a server. The artificial neural network transforms the system to an intelligent navigation system and makes it superior to the other evacuation systems. The produced navigation instructions are sent to the users' smartphones in a real-time manner. The proposed system guides the users over their smartphone instantaneously via vocally-visual elements and allows them to be transported to the exit. The developed system has the quality of becoming a candidate of being a commonly-used evacuation system.

Key Word : Indoor navigation, mobile, RFID, evacuation, fire, artificial neural network.

Science Code : 902.1.014

TEŞEKKÜR

Rahman ve Rahim olan Allah'ın adıyla… Bu tez çalışmasının her aşamasında beni yönlendiren, cesaretlendiren, teşvik eden, maddi manevi desteğini hiçbir zaman esirgemeyen hem kişiliği ile hem de akademisyenliği ile örnek aldığım ve akademisyenliğe bakış açımı değiştiren Sayın Hocam Doç. Dr. İsmail Rakıp KARAŞ'a en kalbi duygularımla teşekkür ederim.

Ayrıca bu tezin şekillenmesinde değerli yorum ve değerlendirmeleri ile desteklerini esirgemeyen tez izleme komitesindeki hocalarım Sayın Prof. Dr. Bülent BAYRAM'a ve bu çalışmanın ilerlemesinde çok önemli katkıları olan Sayın Yrd. Doç. Dr. Ümit ATİLA'ya bir teşekkürü borç bilirim. Yine bu çalışma süresince gece yarılarına kadar benimle beraber çalışan arkadaşım Öğr. Gör. Emrullah DEMİRAL'a, zaman zaman bana yardımcı olan lisans öğrencilerim Mesul Ünal, Orkun Abacı, Tuğrul Sali'ye ve teknik desteğinden dolayı İbrahim Rakıp KARAŞ'a ayrıca teşekkür ederim.

Üzerimde emeği olan eğitim hayatımdaki bütün hocalarıma sonsuz teşekkür ederim. Beni büyütüp, vatana millete hayırlı olsun diye yetiştiren ve bu günlere gelmemde en büyük pay sahibi olan anne-babama ve bütün aile büyüklerime sonsuz şükranlarımı sunarım. Ayrıca bu zorlu süreçte desteğini hep hissettiğim, fedakârca davranan, beni hep anlayışla karşılayan ve benim yerime de çocuklarımızla ilgilenen kıymetli eşim Ayşe'ye, çoğu zaman ihmal etmek zorunda kaldığım dünyalar güzeli kızım Şevval'e ve yeni doğan biricik oğlum Muhammed Hamza'ya tüm kalbimle teşekkür ederim.

Tez çalışmamı 112Y050 proje numarasıyla destekleyen TÜBİTAK'a ve KBÜ-BAP-14/2-DR-007 BAP Lisansüstü Tez Destek Projesi numarasıyla destekleyen Karabük Üniversitesine teşekkür ederim.

Her şeyin üstünde bana bu çalışmayı tamamlamayı ve bu dereceyi nasip eden, sonsuz ilim ve mutlak hüküm sahibi RABBİME binlerce kere şükürler olsun.

İÇİNDEKİLER

Sayfa

Sayfa

Sayfa

ŞEKİLLER DİZİNİ

Sayfa

Sayfa

ÇİZELGELER DİZİNİ

Sayfa

SİMGELER VE KISALTMALAR DİZİNİ

KISALTMALAR

RFID : Radio-Frequency Identification (Radyo Frekansı Tanımlama)
3B : Üç Boyutlu
RSSI : Received Signal Strength Indication (Alınan Sinyal Gücü Ölçümü)
XML : Extensible Markup Language (Genişletilebilir İşaretleme Dili)
IMU : Inertial Measurement Unit (Atalet Ölçümü Birimi)
GPS : Global Positioning System (Global Konumlandırma Sistemi)
INS : Inertial Navigation System (Atalet Seyrülsefer Sistemi
GIS : Geographical Information System (Coğrafi Bilgi Sistemi)
CBS : Coğrafi Bilgi Sistemi
HTTP : Hyper Text Transfer Protocol (Hiper Metin Transfer Protokolü)
CGI : Common Gateway Interface (Web Sunucuya Genel Çıkış Kapısı)

BÖLÜM 1

GİRİŞ

Günümüzde şehirlerimiz gökdelenler, hava alanları, alışveriş merkezleri gibi çok katlı ve karmaşık binalar ile doludur. Bu binalarda özellikle binaya aşina olmayan insanlar yönlerini kaybetmekte, aradıkları yerleri ya da çıkışı bulmakta problem yaşamaktadırlar. Ayrıca herhangi bir acil durumda (yangın, gaz sızması, su baskını, terörist saldırı vb.) insanların bu binalardan çıkışı çok daha zorlaşmaktadır. Böyle acil durumlarda, bu tarz binalarda bulunan insan sayısının oldukça fazla olması, çıkışlarda ve merdivenlerde yığılmalara, paniğe ve izdihamlara sebep olabilmektedir. Hatta yangın durumunda bazı afetzedeler çıkış güzergâhı bulamamakta, yanarak ya da camdan atlayarak hayatını kaybetmektedirler.

Bu tür olumsuz senaryolara yönelik geliştirilen mevcut tahliye sistemlerinin birçoğu kullanıcının kişisel özelliklerini ve acil durumlarda binadaki değişen şartları göz ardı etmekte ve genel kullanıma yönelik tahliye hizmeti sunmaktadır. Bu tür sistemlerin bazıları yangında binadaki değişen şartları tespit edememekte; bazıları ise sıcaklık, gaz, radyasyon sensörleri ile acil durumu tespit edebilmekte fakat bütün kullanıcılara aynı kaçış güzergâhını sunmaktadır. Hâlbuki bir tahliye sistemi kullanıcının kişisel özelliklerini dikkate almalı ve kullanıcı ile her an etkileşimde olup, bina içindeki değişen koşulları tespit ederek her bir kullanıcıya farklı çıkış güzergâhları sunabilmelidir.

Bir tahliye sisteminde aranan en temel özelliklerden birisi, sistemin dinamik bir yapıya sahip olmasıdır. Çünkü acil durumlarda bina içerisindeki koşullar sürekli değişebilmektedir. Bu nedenle binadaki değişen fiziksel koşullar gerçek zamanlı olarak takip edilmelidir. Sıcaklık, gaz ve radyasyon gibi bina içindeki fiziksel koşulları etkileyen faktörlerdeki değişim anlık olarak tespit edilmeli ve gerekiyorsa kullanıcıya sunulan çıkış güzergâhında değişiklikler yapılabilmelidir. Örneğin yangının başladığı

bir binada, sistemin kullanıcıya sunduğu güzergâha yangın sıçramış ise, sistem bu tehlikeyi tespit edip, kullanıcıya farklı bir çıkış güzergâhı sunabilmelidir.

Ayrıca tahliye sistemleri yangın olan bir binada, yangının hangi bölgede başladığı ve nerelere yayıldığı bilgisin dışında, karbon monoksit yayılımı ve görüş mesafesi gibi faktörleri de tespit edebilmelidir. Çünkü bu iki faktör de sıcaklık artışı gibi insan sağlığını ciddi derecede tehdit eden etkenlerdendir. Bunun yanı sıra, binadaki bağlantılar üzerinde insan yoğunluğunun tespit edilip, çıkış güzergâhlarında yığılmaların oluşması engellenmelidir. Çünkü özellikle yüksek katlı binalarda çıkış yollarındaki yığılmalar insanlarda panik durumunun oluşmasına ve izdihamlara sebep olmaktadır. Bu sebeple tahliye sisteminin kullanıcıya sunduğu güzergâhta yığılmalar var ise sistemin güzergâhı değiştirerek kullanıcıya daha güvenli bir yol sunabilmesi gerekmektedir. Böylece insanların paniğe kapılması ve bina içinde izdiham oluşumları engellenmiş olur. Bütün bu özellikleri ihtiva eden bir tahliye sistemi, kullanıcıların her an güvendiği ve talimatlarına uymakta tereddüt yaşamadığı, binadaki değişen koşullara anlık olarak tepki verebilen, dinamik bir tahliye sistemi özelliğine sahip olacaktır.

Ayrıca sistemin, genel amaçlı bir navigasyon sisteminden ziyade kullanıcıların fiziksel özelliklerine göre özel yönlendirme yapabilen bir navigasyon sistemi olması gerekmektedir. Çünkü kullanıcının genç ya da ihtiyar olması, erkek ya da bayan olması, fiziksel engelinin, binaya aşinalığının olup olmaması ya da çeşitli hastalıklarının olup olmaması, yönlendirmeyi etkileyen önemli faktörlerdir. Acil bir durumda genç ve sağlıklı bir kişiye sunulan çıkış güzergâhı ile yaşlı ya da engelli bir kişiye sunulan güzergâhın aynı olması çoğu zaman etkili bir yönlendirme hizmeti sağlamamaktadır. İdeal bir tahliye sistemi kullanıcının kişisel özelliklerini dikkate alabilmeli ve buna göre kişiye özel tahliye güzergâhı üretebilme yeteneğine sahip olmalıdır.

Yine mevcut tahliye sistemlerinin birçoğu mobil cihazlar üzerinde çalışabilecek nitelikte olmayıp, simülasyon ortamında çalışacak şekilde tasarlanmıştır. Bu durum birçok uygulamanın gerçek hayatta kullanımını imkânsız hale getirmektedir. Bir tahliye sistemi, kullanıcı ile anlık olarak etkileşim kurabildiği mobil bir platformda

çalışabilmelidir. Günümüzde böyle bir sistemin çalışabileceği ve bütün insanlara hitap edeceği platform artık neredeyse herkesin sahip olduğu akıllı telefonlardır. Kısacası geliştirilecek bir tahliye sisteminin günümüzde akıllı telefonlar üzerinde çalışabilen mobil bir uygulamaya sahip olması gerekmektedir.

Hazırlanan bu tezde, kullanıcının kişisel özellikleri ile yangın durumunda binadaki fiziksel şartları anlık olarak takip eden ve bu veriler doğrultusunda kullanıcıyı çıkışa kadar yönlendiren YSA (Yapay Sinir Ağı) destekli akıllı bir tahliye sistemi tasarlanmıştır. Bu sistem temel olarak üç farklı bileşenden oluşmaktadır. Birinci bileşen kullanıcının tahliye süresince konumunu belirleyen bir konum tespit modülü, ikinci bileşen sunucu üzerinde çalışan, kullanıcıya ve ortama ait faktörleri temel alarak yönlendirme talimatlarını oluşturan bir yönerge modülü ve son bileşen ise kullanıcının akıllı telefonun çalışıp kullanıcıyı yönlendirecek mobil uygulamadır.

Kullanıcının bina içindeki konumu anlık olarak RFID teknolojisi ile tespit edilmektedir. Kullanıcının konumunun anlık takibi, kullanıcı elinde taşıdığı RFID okuyucusu cihaz ile gerçekleştirilmektedir. Bu kapsamda test için seçilen binada belirli noktaların tavanlarına RFID etiketleri yapıştırılmıştır. Kullanıcının bina içindeki hareketi boyunca RFID okuyucu kendisine en yakındaki etiketten gelen sinyali okuyacak ve kullanıcının konumunu tespit edecektir.

Yönerge modülü, sunucu üzerinde kullanıcıya ait konum verilerini, kişisel verileri ve ortam verilerini birer parametre olarak alıp, kullanıcıya özel yönlendirme talimatlarının üretildiği web sunucusu ortamıdır. Tasarlanan yönerge modülüne YSA entegre edilerek akıllı yönlendirme yapan bir navigasyon sistemi özelliği kazandırılmıştır. Bu YSA, kullanıcıya ait on farklı kişisel özelliği ve ortamdaki koşulları temsil eden altı farklı faktörü birer giriş parametresi olarak alıp, bina içindeki her bir bağlantının kullanıcı için kullanılabilirlik derecesini ortaya çıkaran bir model olarak tasarlanmış ve eğitilmiştir. Böylece kullanıcıların kişisel özelliklerine uygun, güvenli bir çıkış güzergâhı üreten, akıllı bir tahliye sistemi elde edilmiştir. Ayrıca kullanıcının bu tahliye sistemi ile etkileşimini sağlayan bir mobil navigasyon uygulaması hazırlanmıştır. Bu mobil uygulama, tahliye esnasında kullanıcının bulunduğu bölgenin fotoğrafını ekrana getirip, kullanıcıyı sesli ve görsel talimatlar ile

binanın en uygun çıkışına yönlendirmektedir. Bu kapsamda bina içindeki bağlantıların iki yönlü fotoğrafları çekilerek mobil uygulamada kullanılacak fotoğraflar hazırlanmış ve sunucuya yüklenmiştir. Sunucudaki yönerge modülünde kullanıcı için hazırlanan görsel ve sesli talimatlar ile ortam fotoğrafları belirli periyotlar ile mobil uygulamaya gönderilmektedir. Tez kapsamında geliştirilen akıllı tahliye sisteminde temel olarak aşağıdaki adımlar izlenmiştir:

- Çalışma alanına ait 3B bina topolojik yol ağının modellenmesi ve 3B konumsal veri tabanının yüklenmesi,
- Bina için fotoğraflarının çekilmesi, optimize edilmesi ve konumsal veri tabanı ile ilişkilendirilmesi,
- Test binasındaki yönlendirme açısından önemli olan bölgelerin RFID etiketleri ile donatılması ve 3B koordinatların ölçümü,
- RFID Tabanlı Konum Belirleme Sisteminin tasarlanması ve sunucu ile entegrasyonu,
- Sunucu üzerinde yönerge modülünün tasarlanması ve YSA ile entegrasyonu,
- Yönerge modülünde görsel, sözlü ve metin formatındaki talimatların oluşturulması,
- Tahliye sisteminin kullanıcı ile buluştuğu mobil uygulamanın tasarlanması ve sunucu-istemci mimarisine göre sunucudaki yönerge modülü ile haberleşmesinin sağlanması,
- Test binasında geliştirilen akıllı tahliye sistemin test edilmesi

Böylece kullanıcının kendi kişisel özelliklerine uygun ve kendisi için tehlike arz etmeyen en kısa yoldan bina dışına tahliyesini gerçekleştiren, gerçek zamanlı, kullanıcı ile direk etkileşim kuran, dinamik, akıllı ve mobil bir tahliye sistemi elde edilmiştir.

1.1. LİTERATÜRDEKİ BENZER ÇALIŞMALAR

Bu bölümde literatürde konu ile ilgili yapılmış çalışmalar üç ana başlığa ayrılıp ayrı ayrı incelenmiştir. İlk kısımda literatürdeki RFID ile konum belirleme konusundaki

çalışmalar, ikinci kısımda bina içi navigasyon sistemleri, son kısımda ise acil durumlarda kullanılabilecek bina tahliye sistemleri incelenmiştir.

1.1.1. RFID Tabanlı Konum Belirleme İle İlgili Çalışmalar

Bina içi konum belirlemeye yönelik olarak önerilen birçok teknoloji mevcuttur. Bu teknolojiler Ultrasound, RFID, Bluetooth, WLAN, Pseudo GPS ve Infrared teknolojileridir [1]. Avantajları ve doğruluğu göz önüne alındığında söz konusu yöntemler içerisinde RFID (Radio Frequency Identification) teknolojisi öne çıkmaktadır [2]. Bu bölümde literatürde yer alan RFID teknolojisi tabanlı iç mekânlara yönelik konum belirleme uygulamaları üzerinde durulmuştur.

Demiral vd. RFID tabanlı konum belirleme teknikleri üzerine yapılmış 28 farklı çalışmayı kapsamlı olarak incelenmiş ve bu çalışmaların avantajları ve dezavantajlarını ortaya koymuşlardır [3]. RFID sisteminde, okuyucu ve taşıyıcıların birbirlerini doğrudan görmeleri gerekmeksizin aralarında kablosuz iletişim kurabilmeleri sistemin bina içerisinde kullanımını kolaylaştırmaktadır. Bir diğer önemli özellik ise birden fazla taşıyıcının aynı anda okunabilmesinin mümkün olmasıdır.

Literatürdeki çalışmalar incelendiğinde RFID ile konum belirleme konusundaki çalışmaları iki ana başlıkta toplamak mümkündür. İlki okuyucunun hareketli etiketlerin sabit olduğu çalışmalar, diğeri ise okuyucunun sabit etiketlerin hareketli olduğu çalışmalardır. Aşağıda öncelikle literatürdeki okuyucunun hareketli etiketlerin sabit olduğu çalışmalardan örnekler verilmiştir.

Byoung vd. RFID etiketlerin yerleşimlerine bağlı olarak iki farklı metot denemiş ve konum tespitindeki hatanın değişimi incelenmişlerdir. Ayrıca bu sistem ultrasonik sensörler ile desteklenerek, sistemin konum belirleme konusundaki doğruluğu artırılmıştır. Kullandıkları pasif etiketleri 30 cm ve 50 cm aralıklarla olmak üzere zemine iki farklı biçimde yerleştirmişler, okuyucu ise hareketli bir robot üzerine sabitlenmiştir. Okuyucunun doğrusal hareketi ile konumu tespit edilmeye çalışılmış ve

50 cm aralıklı etiketler kullanıldığında hata miktarı 2,42 cm iken, 30 cm aralıklı etiketler kullanıldığında ise hata miktarı 1,58 cm'ye düşmüştür [4].

Lee H. ve Lee M. robotların konumunu RFID teknolojisi ile tespit edip, tespitteki doğruluk oranını artırmaya yönelik yaptıkları çalışmada, RFID etiketleri 5 cm aralıklar ile zemine yapıştırmışlardır. Konum tespitindeki hatanın okuyucu ile etiketler arasındaki mesafeden etkilendiklerini tespit etmişlerdir. Hata miktarını azaltmak için Gaussian fonksiyonunu temel alan bir ağırlıklandırma fonksiyonu önermişlerdir. Ayrıca Hough dönüşüm fonksiyonunu kullanarak, robotun hareket yönünü belirlemişlerdir [5]. Yine Hahnel vd. mobil bir robotun sağ ve sol taraflarına 45°'lik açılar ile iki RFID anteni takmışlar ve bina içini de RFID etiketler ile donatmışlardır. RFID ile birlikte FastSLAM algoritmasını kullanan bir lazer tarama teknolojisi kullanarak, robotun bina içindeki konumlarını tespit etmişlerdir [6].

Literatürdeki bir grup çalışmada ise okuyucuların konumlarının sabit olduğu etiketlerin hareketli olduğu RFID konum belirleme yaklaşımı kullanılmıştır. Bu çalışmalarda etiketlerden alınan RSSI değerlerine bağlı olarak en az iki okuyucudan alınan veri üzerinde, trilaterasyon ve triangulasyon yöntemleri kullanılmış ve etiketlerin konumu tahmin edilmeye çalışılmıştır [7-9].

Bechteler ve Yenigün tarafından yapılan çalışmada üç RFID okuyucusu ve bir RFID etiketi kullanılmıştır. RFID okuyucuların hareketli etiketlerden aldığı RSSI değerlerine göre okuyucu ile etiket arasındaki mesafe hesaplanmış, bu mesafe değerleri ile trilaterasyon yapılarak hareketli halde olan etiketin konumu tespit edilmiştir [8].

Ni vd. bina içinde konum belirleme için RFID teknolojisini kullanan LANDMARC ismini verdikleri bir yaklaşım geliştirmişlerdir. Bu yaklaşımın sunduğu en önemli fayda, hareketli nesnelerin referans etiketleri kullanarak konumlarının belirlenmesinde doğruluğun önemli ölçüde artırılması olmuştur. Bu referans etiketleri bina içine 1 m aralıklar ile yerleştirilmiş aktif RFID etiketleridir. Hareketli bir etiketin konumunu belirlemede, okuyucunun bu etiketten aldığı RSSI değeri ile aynı ortamda bulunan referans etiketinden aldığı değer beraber kullanılmıştır. Referans etiketin konumu sabit

olduğu ve mutlak olarak bilindiği için, k-N komşuluk algoritması kullanılarak hareketli durumdaki etiketin konumu hesaplanmıştır [10].

Jiang vd. RFID teknolojisi ile bina içi konum tespitinde doğruluğun artırılması konusunda yaptıkları çalışmada hareketli RFID etiketlerinin yanı sıra sabit referans etiketlerini kullanmışlardır. Bu çalışmada etiketlerden okunan RSSI değeri, geleneksel yöntemlerde olduğu gibi geometrik mesafe değerine dönüştürülmemiştir. Yazarlar bu dönüşümün karmaşık bina ortamlarında yeterli doğruluğu vermediğini iddia etmektedirler. Yeni bir yaklaşım olarak hareketli etiket ile bulunduğu bölgedeki diğer referans etiketler bir dizi içine atılmıştır. Anlık olarak bu dizi içindeki her bir etiketin konumu hesaplanmaktadır. Referans etiketlerin gerçek konumları bilindiği için elde edilen değerler ile gerçek değerler karşılaştırılarak konum tespitindeki hata miktarı bulunmaktadır. Gerekli kalibrasyonlar tekrar tekrar yapılarak, hata miktarı en düşük seviyeye indirilmeye çalışılmıştır [11].

Hazırlanan bu tezde konum tespiti için birinci yaklaşım kullanılmıştır. Yani etiketlerin sabit, okuyucunun hareketli olduğu ve kullanıcının taşıdığı RFID okuyucusu ile binaya yerleştirilmiş pasif RFID etiketleri aracılığıyla konumunun tespit edildiği bir yöntem tercih edilmiştir. Bu tercihte uygulamanın geliştirme maliyetinin düşük olması ve yakın gelecekte akıllı telefonlarda RFID teknolojisine sahip olacağının düşünülmesi etkili olmuştur.

1.1.2. Bina İçi Navigasyon İle İlgili Çalışmalar

Bu bölümde bina içinde yönlendirme ile ilgili yapılmış çalışmalar incelenmiş ve bu çalışmaların avantaj ve dezavantajları ortaya konulmuştur. Jain vd. yaptıkları çalışmada akıllı telefonlar kullanılarak geliştirilen çeşitli bina içi navigasyon uygulamalarını maliyet, doğruluk, kullanım kolaylığı, sürdürülebilirlik, mahremiyet, hata oranı, hesaplama karmaşıklığı ve veri transferleri gibi kriterler üzerinden karşılaştırarak değerlendirmelerde bulunmuşlardır [12].

Chung vd. bina içinde insanları yönlendirmek için artırılmış gerçeklik nesneleri ile desteklenmiş bir prototip navigasyon uygulaması geliştirmişlerdir. Bu uygulamada

yönlendirme talimatları binanın zeminine yansıtılmaktadır. Böylece yönlendirme için ekran değil mobil cihaza takılan bir yansı aparatını kullanılmıştır. Ayrıca yazarlar önerdikleri sistem ile ekran tabanlı navigasyon sistemlerini kullanıcı görüşlerini de göz önünde bulundurarak karşılaştırmışlardır. Önerilen sistem kullanıcının konumunu takip etmediği için tamamlanmış bir navigasyon uygulamasından ziyade bir yönlendirme tekniğidir [13].

Khalifa vd. hipermarketlerde engellileri yönlendirecek bir bina içi navigasyon uygulaması geliştirmişlerdir. Uygulamada öncelikle kullanıcı alış-veriş listesindeki malzemeleri sisteme tanımlaması gerekmektedir. Bu işlemden sonra sistem kullanıcının konumunu RFID ile tespit etmekte ve kullanıcının alış-veriş listesindeki malzemelerin konumunu veri tabanında sorgulamaktadır. Hem kullanıcının hem de malzemelerin tamamının konumları tespit edildikten sonra sistem kullanıcının en kısa sürede alış-veriş işlemini tamamlaması için en kısa yolu bulmaktadır. Ancak uygulamanın mobil bir uygulama olmaması kullanım açısından dezavantaj oluşturmaktadır [14].

Blattner vd. hareket kabiliyeti sınırlı kullanıcıların bina içinde yönlendirilmesi için Bluetooth sinyal yayıcıları kullanarak bir navigasyon sistemi geliştirmişlerdir. Bu çalışmada bina planı XML formatında bir dosyaya dönüştürülmüştür. Binanın bütün bileşenleri XML etiketleri ile temsil edilmiştir. Bu sistem kullanıcının bina içinde gitmek istediği noktaya olan en kısa ve kendisi için herhangi bir engel içermeyen yolu bulan ve bir Android mobil cihaz üzerinde kullanıcıyı yönlendiren bir bina içi navigasyon sistemidir [15].

Sriharee, OWL (Web Ontology Language) tabanlı semantik yönlendirme yapan bir bina içi navigasyon uygulaması geliştirmiştir. Bu navigasyon sisteminde yol bulma ve kullanıcıyı yönlendirme işlemi sembolik bilgiler ile gerçekleştirilmektedir. Uygulamanın istemci tarafını Android mobil cihazlar temsil ederken, sunucu tarafını ise semantik web teknolojilerini kullanan bir bina içi navigasyon servisi temsil etmektedir. Geliştirilen uygulama kullanıcının konumunu otomatik olarak tespit edememektedir. Bundan dolayı kullanıcı her konum değişikliğinde mobil uygulama üzerinden konumunu güncelleyerek sistemden navigasyon hizmeti almaktadır [16].

Czogalla şehirlerde yoğun bir insan trafiğine sahip, kısmi olarak kapalı alan statüsündeki transfer merkezlerinde insanların yönlendirilmesi için bir navigasyon sistemi tasarlamıştır. Tasarlanan bu sistem Android akıllı telefonlara yönelik olup, yönlendirme işleminde telefonun WIFI, GPS, Bluetooth, kamera ve atalet ölçüm sensörlerini (IMU-Inertial Measurement Unit) kullanılmıştır. Telefonlarda genel olarak atalet ölçüm sensörü olarak ivme ölçer, mıknatıs ölçer, jirometre ve basınç ölçer mevcuttur. Kapalı alanlarda bu atalet ölçüm sensörleri ile kullanıcının bina içinde konum tespiti ve yönlendirilmesi yapılmıştır. Kullanıcının hangi katta olduğu basınç sensörü ile tespit edilmiştir. Bu navigasyon sisteminde bina kat planı OpenStreetMap programından elde edilmiş ve açı koruyan Web Mercator koordinat sisteminde görselleştirilmiştir. Başlangıç noktası ile varış noktası arasında Enine Arama (Breadth First Search-BFS) algoritması kullanılarak muhtemel güzergâh belirlenmiştir [17].

Koac vd. bina içindeki çıkış levhası, yangın işaretçi levhaları ve metinsel levhaları kullanarak kullanıcıları bina içinde yönlendiren bir navigasyon sistemi geliştirmişleridir. Bu sistem bina içinde herhangi bir yapay işaretçiye ihtiyaç duymadan, binada zaten mevcut olan doğal işaretçiler ile yönlendirme yapmaktadır. Yapılan çalışmada iPad bir cihaz kullanılmış ve bu cihazın kamerasından elde edilen görüntülerde bu doğal yönlendirme levhaları tespit edilip, kullanıcının levhaya olan uzaklığı ve açısı tespit edilebilmektedir. Böylece kullanıcıyı bina içinde mobil cihaz ile 3B ok işaretleri ile yönlendirmektedir. Sunulan sistemde, bir ortamda aynı levhadan birden fazla olması ya da iki farklı levha arası mesafenin 10 m'den daha fazla olması gibi durumlarda sistem kullanıcının konumunu tespit edememektedir [18]. Bina içinde konum tespiti ve yönlendirme konusundaki bazı çalışmalarda doğal işaretçilerin yerine yapay işaretçiler de kullanılmıştır [19].

Neges vd. canlı video görüntülerini ile IMU tabanlı sensörlerden elde edilen verileri birleştirerek bir bina için navigasyon sistemi sunmuşlardır. Sunulan sistemde binanın içi, artırılmış gerçeklik işaretçileri ile donatılmıştır. Bu işaretçiler kullanıcının taşıdığı mobil cihazın kamerasından elde edilen canlı görüntülerde aranmaktadır. Eğer görüntü içerisinde işaretçilerden biri tespit edilmiş ise, kullanıcının bina içindeki konumu bu işaretçiye göre belirlenmektedir. İşaretçinin görülemediği bölgelerde ise IMU

sensörleri kullanılarak adım ve yön tespiti yapılmakta ve kullanıcının konum takip edilmektedir [20].

Serrao vd. bina içindeki yönlendirme açısından önem ifade eden noktalarda görsel işaretçiler (visual landmarks) kullanarak özellikle görme engelli kullanıcılara yönelik bir bina içi navigasyon uygulaması tasarlamışlardır. Bu görsel işaretçiler kullanıcının elinde taşıdığı bir kamera ile tespit edilmektedir. Bu kamera kullanıcı tarafından yatay olarak 180º, dikey olarak ise 45º döndürülebilmektedir. Sistem kameradan gelen görüntüleri işleyerek, işaretçileri tespit etmekte ve buna göre kullanıcının konumunu belirlemektedir. Bununla birlikte kullanıcının bulunduğu konumdan gitmek istediği noktaya bir güzergâh oluşturulmakta ve bu güzergahta kullanıcı sesli olarak yönlendirilmektedir. Kullanıcının bu sistemden hizmet alabilmesi için elinde dizüstü bir bilgisayar taşıması gerekmektedir. Önerilen sistemin mobil cihazlara uyarlanabilmesi durumunda kullanılabilirliği çok daha artacağı bir gerçektir [21].

Bellini vd. acil durumlarda sağlık personelinin yönlendirilmesi için mobil bir rehber uygulaması sunmuşlardır. Bu sistem sağlık personeline hem dış ortamlarda hem de bina içi ortamlarda yönlendirme hizmeti vermektedir. Dış ortamda konum tespiti ve yönlendirme GPS yardımı ile yapılırken, bina içinde QR kodu ve mobil cihazdaki sensörleri kullanılarak gerçekleştirilmiştir. Sensör verileri adaptif Kalman filtreleri kullanılarak konum tespiti açısından uygun hale getirilmiştir. Ayrıca bu navigasyon sisteminde sağlık personelini yönlendirecek merkezi bir kontrol birimi de bulunmaktadır [22].

Yaşlı ve görme engelli insanların aşina olmadıkları bir binada yönlerini bulmaları oldukça zordur. Tsirmpas vd. bu insanlara yönelik bir bina içi navigasyon sistemi tasarlamışlardır. Tasarlanan sistem kullanıcıların konumları RFID teknolojisi ile belirlemekte ve sesli olarak kullanıcıyı yönlendirmektedir. Ayrıca kullanıcının önündeki engeller ultra-sonic mesafe ölçücü ile tespit edilmekte ve yönlendirme işleminde göz önünde bulundurulmaktadır. Uygulamanın sunucu tarafında Dijkstra algoritması ile kullanıcıya sunulacak güzergâh hesaplanmaktadır [23].

Cheng vd. yayaları hem bina içi hem de bina dışında yönlendirecek bir navigasyon sistemi geliştirmişlerdir. Bu sistemde dış ortamlarda konum tespitinde GPS kullanılırken, bina içinde WIFI tabanlı konumlandırma yapan RSSI parmak izi izleme tekniği kullanılmıştır. Ayrıca bina içindeki konum tespitinde hatayı en aza indirmek için INS (Inertial Navigation System) sensörlerinden de faydalanılmıştır. Bu sensörlerden elde edilen veriler, Kalman Filtresi [24,25] ile filtrelenerek adım tespitinde kullanılmıştır. Tasarlanan navigasyon sisteminin mobil cihazlara yönelik olmaması sistemin gerçek anlamda kullanılabilirliğini olumsuz yönde etkilemektedir. Yazarlar çalışmanın gelecek planı ile ilgili olarak, hazırlanan navigasyon sisteminde mobil cihazlara yönelik uyarlamalar yapılacağına da değinmişlerdir. Böylece kullanıcının sistemden hizmet alırken üzerinde taşımak zorunda olduğu birçok sensöre ihtiyacı kalmayacaktır. Çünkü günümüz mobil cihazları INS sensörlerini ve WIFI alıcılarını zaten bünyelerinde barındırmaktadırlar [26].

Boysen vd. IFC formatındaki ham haldeki sayısal bina verilerinden bina içi ortamlar ile ilgili anlamlı veriler çıkaran bir prototip tasarlamışlardır. Bu çıkarım işleminde PostGIS araçları kullanılmış ve elde edilen bina verileri PostgreSQL'e aktarılmıştır. PostgreSQL konumsal verileri saklayan, işleyen ve bu veriler üzerinde analizler yapabilen bir veri tabanı yönetim sistemidir. Ayrıca tasarlanan bu prototip ürün akıllı telefonlarda çalışan ve iki boyutlu harita üzerinde yönlendirme yapan bir navigasyon modülüne de sahiptir [27].

1.1.3. Acil Tahliye Sistemleri İle İlgili Çalışmalar

Park vd. 2009 yılında acil durumlarda binalardaki insanları tahliye etmekle görevli olan personele yönelik bir navigasyon sistemi tasarlamışlardır. Tasarlanan navigasyon sisteminden uygun yolu bulmak için zaman bağımlı Dijkstra algoritması kullanılmıştır. Bina içerisindeki bağlantılar mesafeye göre değil, kat etmek için gerekli zamana göre ağırlıklandırıldığı için zaman bağımlı olarak tanımlanmıştır. Sistem önce kullanıcının mevcut konumundan en kısa sürede ulaşabileceği çıkışa olan yolu bulmakta, daha sonra ise simülasyonda üretilen ve bina içindeki çeşitli faktörleri temel alarak yönlendirme iterasyonlarını gerçekleştirmektedir [28]. Geliştirilen sistem, bir

simülasyon üzerinde çalışmakta ve kullanıcı ile etkileşimini sağlayacak mobil bir uygulamaya ihtiyaç duymaktadır.

Lee ve Zlatanova, 3B bina içi Coğrafi Bilgi Sistemleri (CBS) veri modelleri üzerinde acil durumlarda tahliye amaçlı kullanılabilecek metodlar tanımlamışlardır [29]. Oracle Veri Tabanı Yönetim Sistemi 10g sürümünde konumsal verileri saklamak, bu veriler üzerinde analizler yapmak için Ağ Veri Modeli (Network Veri Model) isimli bir model geliştirmişlerdir. Bu model verileri hem topolojik hem anlamsal hem de geometrik açıdan ilişkilendirip ağ modeli üzerinde çeşitli analizler yapmaya olanak sağlamaktadır. Bunun yanı sıra verilerin görselleştirilmesi konusunda da yardımcı araçlara sahiptir [30].

Kwan ve Lee, 11 Eylül'de New York Dünya Ticaret Merkezine gerçekleştirilen saldırıdan etkilenerek, gökdelenler gibi çok katlı binalarda muhtemel terörist saldırılara karşı acil müdahale için GIERS (GIS-Based Intelligent Emergency Response Systems) ismini verdikleri 3B CBS tabanlı akıllı bir müdahale sistemi geliştirmişlerdir. Geliştirilen bu gerçek zamanlı sistemin terörist saldırılara karşı yapılan müdahalelerde hız açısından katkılarını sorgulamışlardır. Ayrıca teröristlerin sisteme sızıp kurtarma personeli ile ilgili bilgilere ulaşabileceği konusundaki sistem açısından çekincelerini de belirtmişlerdir [31].

Inoue vd. acil durumlarda mobil cihaz üzerinde kullanıcıların tahliyesi için kullanılan bir sistemi tasarlamışlardır. Bu sistemde bina içi konum tespiti Bluetooth işaretçilerinden (beacon) alınan sinyallerle sağlanmaktadır. Bu çalışma kapsamında bina yangını fark edebilen bir sensör ağına sahiptir. Önerilen sistem yangını tespit ettiğinde güvenli bir çıkış güzergâhı üretmektedir ve kullanıcı bu güzergâh üzerinde yönlendirilmektedir. Bu sistemin dinamik bir yönlendirme yapamaması yani oluşturulan kaçış güzergâhının yangının durumuna göre değiştirilememesi sistemin dezavantajlarındandır [32].

Chu ve Wu yangında tahliye sistemi olarak kullanılacak bulut bilişim tabanlı bir tahliye sistemi tasarlamışlardır. Tasarlanan bu sistem, yangın durumunda kullanıcıya sıcaklık, mesafe ve izdiham oluşumu gibi faktörlere bağlı olarak en uygun kaçış

güzergâhını sunmaktadır. Kullanıcının konumu RFID teknolojisi ile belirlenmekte ve sunucu üzerinde bulut mantığı ile çıkış güzergâhı hesaplanmaktadır. Bu sistemde RFID sinyallerini işlemek için Viewpoint Calculator, çıkış güzergâhını belirlemek için bulutta çalışan Path Planner yazılımı ve hesaplanan güzergâhı telefonlar üzerinde görselleştirmek için MobiX3D programı kullanılmıştır [33].

Pang vd. söz konusu bu doktora tezi kapsamında yapılan çalışmaya benzer şekilde, araç rotalama sisteminde yönlendirme aracı olarak YSA'yı kullanmışlardır. Bu sistemde araç sürücülerinin güzergâh seçimindeki davranışları modellenmiş ve kullanıcılara özel en kısa güzergâhı sunan bir sistem tasarlamışlardır. Sürücünün daha önceki yol tercihleri ve bulunan yoldan sapması YSA'nın eğitimi için kullanılmış ve böylece güzergâh seçme işlemi sürücünün tercihlerine uygun şekilde yapılmıştır [34].

Literatürde yer alan çalışmalara genel olarak bakıldığında, geliştirilen tahliye sistemleri dinamik yapıda olmayıp, binada değişen şartlar karşısında çoğu zaman kayıtsız kalmaktadır. Yine birçok bina içi navigasyon sisteminin, akıllı bir yönlendirme yeteneğine ve dinamik bir kontrol mekanizmasına sahip olmadığı görülmektedir. Bu durum acil vakalarda bina içi trafik yoğunluğuna, izdihama, çıkışların tıkanmasına ve hatta insanların zarar görmesine sebep olabilmektedirler. Bu konuda yapılan çalışmalar, genel olarak kullanıcıların kişisel özelliklerini göz ardı eden ve binadaki değişen şartları dikkate almayan, genel amaçlı tahliye sistemleridir. Bazıları ise simülasyon ortamlarında çalışan, mobil arayüzü olmadığı için ya da gerçek zamanlı çalışmadığı için kullanıcı ile etkileşim sağlayamayan ve gerçek hayatta kullanılabilirliği olmayan sistemlerdir.

Hazırlanan bu çalışmada hem binadaki değişen koşullar hem de kullanıcının kişisel özellikleri dikkate alınarak, gerçek zamanlı çalışan, mobil bir arayüze sahip akıllı bir tahliye sistemi geliştirilmiştir. Bu tahliye sistemi başlangıçta kullanıcıya güvenli bir güzergâh sunmaktadır. Eğer kullanıcının tahliyesi esnasında sunulan bu güzergahta kullanıcı için risk oluşturacak bir gelişme olduysa, sistem hemen kullanıcı için alternatif ve daha güvenli bir güzergâh oluşturacak kabiliyete sahiptir. Yani sistem dinamik bir yapıda çalışmaktadır. Bunun yanı sıra tahliye sistemi gerçek zamanlı çalışmakta, kullanıcının konumunu anlık olarak tespit edip bulunduğu konuma göre

yönlendirme talimatları oluşturmaktadır. Bu talimatlar kullanıcının akıllı telefonunda görsel, işitsel ve metin içerikli öğeler ile görüntülenecek ve kullanıcının bu şekilde anlık olarak yönlendirilmesi sağlanacaktır. Ayrıca sistemi diğer alternatif tahliye ve yönlendirme sistemlerinden ayıran en önemli özellik, kullanıcıyı kişisel özelliklerine göre yönlendirebilmesidir. Yani tahliye sistemi her kullanıcıya özel çıkış güzergahları sunabilmektedir. Böylece kullanıcı kendisi için en uygun ve güvenli güzergahtan tahliye edilmiş olacaktır. Bu kullanıcıya özel güzergahlar sistemin bünyesinde bulunan özel bir YSA desteği ile hazırlanmaktadır. Bu özelliği ile tahliye sistemi aynı zamanda akıllı bir tahliye sistemi kimliğine sahiptir. Bu sistemin kullanıcı ile etkileşimi, önceden çekilmiş fotoğraflar ile kullanıcının bulunduğu alanı görselleştiren ve kullanıcıyı üç boyutlu olarak bina içinde yönlendirme hizmeti veren bir mobil uygulama ile sağlanmıştır.

1.2. AKILLI TAHLİYE SİSTEMİNDEKİ KISITLAMALAR VE KABULLER

Geliştirilen akıllı tahliye sisteminin gerçek hayatta kullanılabilmesi için bazı ön kabuller yapılması gerekmektedir. Sistemin kullanılabilirliği ile ilgili bu ön kabuller, bina ile ilgili kısıtlamalar ve kullanıcı ile ilgili kısıtlamalar olmak üzere iki başlık altında ele alınmıştır. Bu tez kapsamında hem bu sistemi kullanacak bireylerin hem de sisteminin uygulanacağı binanın bu kısıtlamalara uyduğu kabul edilmiştir.

1.2.1. Bina İle İlgili Kısıtlamalar Ve Kabuller

- Bu tez kapsamında geliştirilen akıllı tahliye sistemi, Bayındırlık ve İskân Bakanlığınca hazırlanan ve 2007 yılında resmi gazetede yayınlanan "Binaların Yangından Korunması" hakkında yönetmeliğe uygun binalar için tasarlanmıştır [35]. Bu yönetmeliğe göre:

    - Yüksekliği 21,50 m'den fazla olan binalarda, binanın dış yüzeyine kaçış merdiveni yapımına izin verilmediği için, hazırlanan akıllı tahliye sisteminde binanın dışında bulunan yangın merdivenleri yönlendirme hizmetinde kullanılmamıştır.

    - Binada normal merdivenlerin yanı sıra, tekerlekli sandalyeli fiziksel engellilerin kullanabileceği düz merdiven rampalarının olduğu kabul edilmiştir.
    - Binanın elektrik tesisatı ve sistemleri yangına dayanıklı kablolar ve cihazlar kullanılarak tesis edilmelidir. Buna bağlı olarak, kaçış yolları aydınlatmasının, yangın algılama ve uyarı sistemlerinin, yangın hâlinde çalışır durumda olması gerekmektedir.
- Bina yangını algılama, kontrol ve uyarı için yazılım ve donanım bileşenlerden oluşan bir kontrol sistemine sahip olmalıdır. Yani bina akıllı bina niteliğinde bir bina olmalı ve içerisindeki çeşitli faktörlerin (sıcaklık, karbon monoksit, görüş mesafesi, insan yoğunluğu vb.) değişimini tespit edebilen bir sensör ağına ve bu verileri anlamlandıran yazılımlara sahip bir kontrol sistemine sahip olması gerekmektedir. Ayrıca bina sensörlerden elde edilen verileri anlık olarak merkezi bir sunucuya aktarabilecek bir alt yapıya sahip olmaldır.
- Binada kullanıcıların konumlarını anlık olarak takip eden RFID Tabanlı Konum Belirleme sistemi için kullanılan etiketlerin yanmaz özellikte olması gerekmektedir.

1.2.2. Kullanıcı İle İlgili Kısıtlamalar Ve Kabuller

Bu çalışma boyunca kullanıcı olarak tanımlanan kişi, akıllı telefonu ya da tableti vasıtasıyla bu tahliye sisteminden hizmet alan son kullanıcıdır.

- Öncelikle kullanıcının akıllı bir mobil cihaza sahip olması gerekmektedir. Bu kapsamda geliştirilen mobil tahliye uygulaması Android işletim sistemi yüklü akıllı telefon ya da tablet gibi mobil cihaza sahip kullanıcılara yöneliktir.
- Geliştirilen mobil tahliye sistemi, kullanıcının bireysel özelliklerini temel alarak akıllı bir yönlendirme yaptığı için, bütün yönlendirme talimatları akıllı telefona sahibine yöneliktir. Yani bir başka kullanıcı için ya da bir grup insan için ortak kullanılabilecek bir uygulama değildir. Her bir kullanıcı kendi kişisel özelliklerine göre kendi akıllı telefonundan farklı bir yönlendirme hizmeti almalıdır. Eğer kullanıcı yanındaki çocuğu, annesi, babası ya da eşiyle birlikte

bu sistemden hizmet almak zorunda ise, bu grup içinde sistem açısından yönlendirilmesi en zor kişi baz alınmalı ve mobil uygulamaya bu baz alınan kullanıcının özellikleri tanımlanmalıdır. Örneğin 40-60 yaş aralığında olan, herhangi bir hastalığı olmayan, atletik bir yapıya sahip erkek bir kullanıcı yanında yine kendisi ile aynı yaş grubunda olan ama şişman ve çeşitli hastalıklara sahip eşiyle beraber sistemden tahliye talibinde bulunacak ise mobil cihazına eşinin özelliklerini tanımlamalıdır. Böylece tahliye sistemi bu çifti bayana göre yönlendirecek ve bayanın yangından zarar görmeden tahliye edilmesi sağlanacak. Bayan için sunulan yol erkek kullanıcı için bir sorun teşkil etmeyeceği için erkek kullanıcı da aynı güzergahtan zarar görmeden tahliye edilebilecektir.

- Ayrıca konumunu takip edebilmek açısından, kullanıcının mobil cihazının RFID okuyucu ile bütünleşik bir yapıda olması gerekmektedir.

BÖLÜM 2

TAHLİYE SİSTEMİ TASARIMINDA KULLANILAN YAZILIM TABANLI TEKNOLOJİLER

2.1. SERVLET

Geliştirilen akıllı tahliye sisteminin sunucu üzerindeki bütün işlemleri Java'nın Servlet sınıfları kullanılarak geliştirilmiştir. Akıllı telefondan gelen kullanıcıya ait kişisel bilgilerinin ele alınması, oturum bilgilerinin tutulması, YSA ile kullanıcı için en uygun güzergâhın bulunması ve bu güzergâh üzerinde kullanıcıyı yönlendirecek talimatların oluşturulup mobil uygulamaya gönderilmesi gibi işlemlerin hepsi servlet sınıflarında gerçekleştirilmektedir. Bu bölümde tahliye sisteminin kontrol merkezi görevini yürüten servlet sınıfları hakkında genel bilgiler verilmiştir.

Web sayfaları kullanılmaya başladıkları ilk dönemlerde statik bir yapıya sahipti ve kullanıcı ile etkileşimli işlemlere izin vermemekteydi. Zamanla kullanıcıya göre değişen ve kullanıcıya dinamik hizmetler verebilen web sayfalarına ihtiyaç duyulmaya başlanmıştır. Günümüzde bir web sayfasından beklenen en temel özelliklerden biri kullanıcılardan veri alarak bu verileri işlemesi ve işlemlerin sonuçlarının kullanıcıya göstermesidir. Java'da dinamik web uygulamaları geliştirmek için farklı teknolojiler bulunmaktadır. Java Servlet bunlardan bir tanesidir [36].

Java Servlet, web sunucusu ya da bir uygulama sunucusu üzerinde ara katman yazılımı olarak çalışır. HTTP istemcisinden ya da web tarayıcısından gelen talebi sunucu üzerinde icra eden Java tabanlı sınıftır. Servlet kullanımı ile veri tabanından, web formlarından ve diğer kaynaklardan elde edilen giriş verileri ile dinamik web sayfaları oluşturulabilir. Java Servlet, Java Enterprise Edition'da Java Servlet API'siyle uyumlu bir Java sınıfı olup HTTP isteklerine cevap vermek için kullanılır. Belirli bir istemci-sunucu protokolüne bağlı olmamasına rağmen genellikle HTTP protokolü ile

kullanılır. Üretilen kod genelde HTML olsa da bazen XML ya da başka bir formatta da olabilir [37]. Servletler, CGI (Comman Gateway Interface) ya da ASP.NET gibi Java dışındaki web içerik teknolojilerinin Java'daki karşıt ürünüdür. Servlet CGI programları ile kıyaslandığında şu tip avantajlara sahiptir [38]:

- Hissedilir derecede performans artışı sağlar.
- Servletler aynı adres uzayında farklı iş parçacıkları (thread) olarak çalışabilirler. Böylece her bir istemci için ayrı proses tanımlanmasına gerek yoktur. İşlemler iş parçacıkları üzerinden gerçekleştirilebilir.
- Java tabanlı oldukları için platformdan bağımsız olarak çalışabilirler. Bu özelliği ile geliştiriciye istediği web sunucuyu, işletim sistemini ve aracı seçme imkânını tanır.
- Sunucu üzerindeki Java güvenlik yöneticisinin sağladığı önlemler sayesinde daha güvenilirdir.
- Java kütüphanelerine tam erişim yetkisine sahiptirler. Veri tabanına erişimde kullanılan JDBC API'si dâhil olmak üzere bütün Java API'lerini kullanabilir.

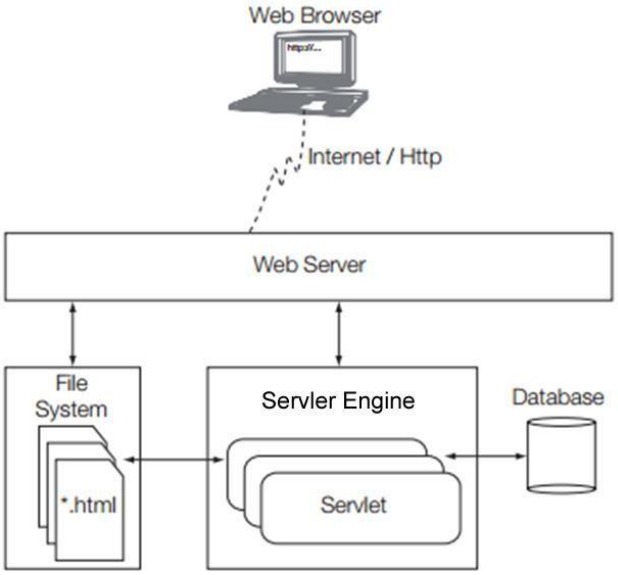
2.1.1. Servletin Görevleri

Servlet temel olarak şu görevleri icra eder [39]:

- İstemciden gelen istekleri okur. Bu istek, bir HTML formdan ya da bir appletten veya bir HTTP istemcisi olarak mobil bir cihazdan gelebilir.
- Okunan veriyi işler ve sonuç üretir. Bu işlemler veri tabanı ile bağlantı kurma, veri tabanı üzerinde sorgular yapma, web servislerini çağırma ya da diğer Java API'lerini kullanma gibi işlemlerdir.
- Üretilen sonucu HTTP cevabı (response) olarak istemciye gönderir. Üretilen sonuç HTML, XML, JSON formatında olabilir.

2.1.2. Servletlerin Mimarisi

Java servletlerinin web sunucu gibi bir servlet motoru (Servlet Engine) üzerinde çalıştırılması gerekir. Servlet motoru, servletleri yönetir ve istemci ile servletler arasındaki bağı kurar. Şekil 2.1'de bir web uygulaması mimarisinde servletin konumu gösterilmiştir.

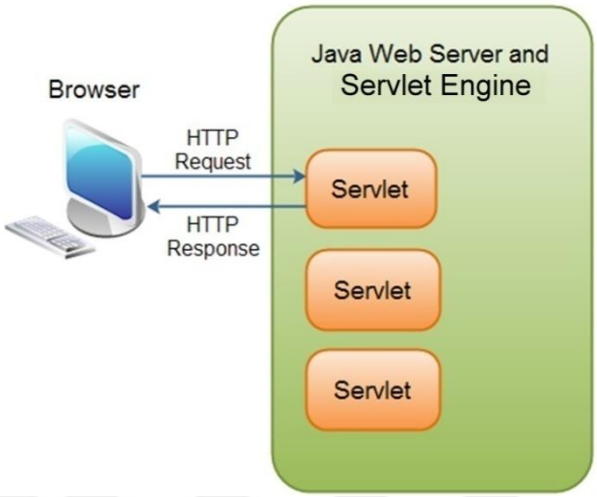


Şekil 2.1. Servletin web sunucudaki konumu [40].

2.1.3. Servlet Nasıl Çalışır?

- Kullanıcı web tarayıcısından ulaşmak istediği sayfayı belirtir ve bu bilgiyi sunucuya gönderir.
- Web sunucusu kullanıcı tarafından gelen bu HTTP isteği (request) ya da başka formatta gelen istekleri alır. Bu isteğe uygun olan servleti belirler. Bu servlete ait bir kopyanın bellekteki varlığını kontrol eder. Eğer var ise istek ve tüm bilgi bu servlete gönderilir. Eğer istenilen servletin kopyası bellekte yok ise yeni bir servlet nesnesi oluşturulur. İstek ve bilgi daha sonra oluşturulan servlete gönderilir. Servlet kendisine gelen isteğe göre bir sonuç üretir. Servletin oluşturacağı cevap (response) farklı biçimlerde olabilir. Genel olarak oluşturulan sonuç HTML sayfası şeklindedir.

- Servlet oluşturduğu sonucu web sunucusuna gönderir.
- Web sunucusu servletten gelen sonucu isteği yapan kullanıcıya gönderir.

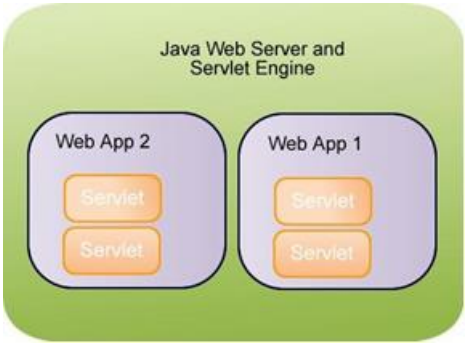


Şekil 2.2. Servletin çalışma şekli [41].

Servlet sınıfları kullanım amaçlarına göre farklı tiplere sahiptirler. HTML isteklerini anlayarak HTML sonucu üreten ve yaygın olarak kullanılan temel servlet sınıfı HttpServlet'dir.

Bir web sunucusunda servletlerin çalıştırabilmesi için, bünyesinde Servlet Motoru (Servlet Engine) bulundurması gerekmektedir. Bu motor, servletleri Java Sanal Makinesi ile çalıştırabilir, oluşan sonuçları kullanıcı ile ilişkilendirip, kullanıcıya geri döndürebilir.

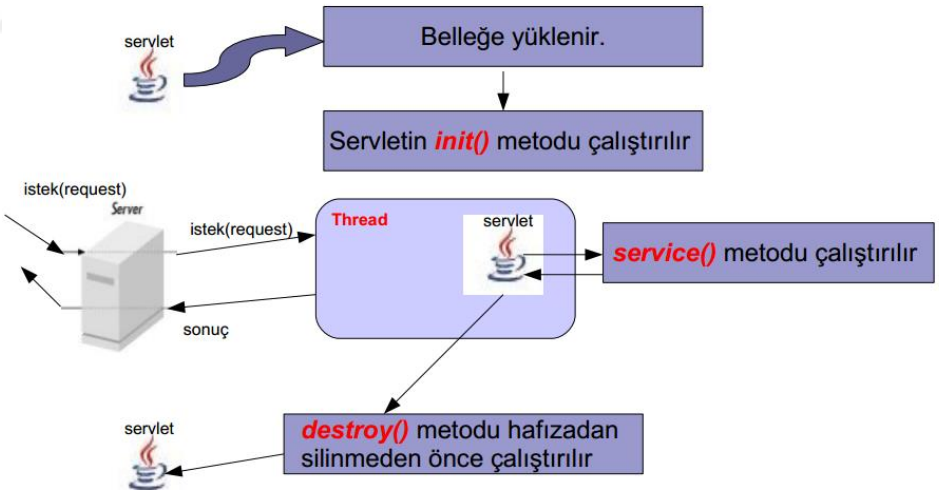
Web sunucusu üzerindeki bir servlete birden fazla istek aynı anda gelebilir. Sunucu bu servlet sınıfından tek bir nesne oluşturur. Bu nesne, çok iş parçacıklı (multi-threading) yapıda tüm istekler için ortak kullanılır. Yani belirli bir servlete erişmeye çalışan istemcilerin her birinin talebi ayrı bir iş parçacığı olarak ele alınır. Sunucuda her bir talep için ayrı bir proses oluşturulmaz. Servletlere ait bu özellik, sunucu üzerinde ciddi bir performans kazancı sağlamaktadır. Ayrıca bir servlet motoru birden fazla web uygulamasını aynı anda çalıştırabilir (Şekil 2.3). Her bir web uygulaması sunucu üzerinde kendi servletini çalıştırır [36]. Bir web uygulaması servlet ile birlikte Java ServerPages (JSP), Java Server Faces (JSF) ve web servisi içerebilir.



Şekil 2.3. Web uygulamalarında servlet kullanımı [41].

2.1.4. Servlet Yaşam Döngüsü

- Servlet init metodu ile oluşturulur.
- İstemciden gelen her çağrı service metodu tarafından değerlendirilir.
- Servlet görevini tamamladıktan sonra destroy metodu ile bellekten kaldırılır.



Şekil 2.4. Servletin yaşam döngüsü [36].

2.1.5. Servlet Metodları

2.1.5.1. Init Metodu

Init metodu servlet ilk oluşturulurken bir kez çağrılacak şekilde tasarlanmıştır ve her bir kullanıcı isteğinde tekrar tekrar çağrılmaz. Kullanıcı ilk servleti çağırdığında, bu servletin tek bir örneği oluşturulur ve belleğe yüklenir. Bu aşamadan sonra servlete gelen her talep, yeni bir iş parçacığı ile servletin doGet ve doPost metoduna erişim sağlar. Init metodu servletin yaşam döngüsü boyunca ihtiyaç duyacağı verilerin oluşturulmasını sağlar.

init metodu tanımlaması şu şekildedir:

```
public void init() throws ServletException {
// Initialization code...
}
```

2.1.5.2. Service Metodu

Service metodu servletin asıl görevini yerine getiren ana metottur. Servlet motoru istemciden gelen isteği değerlendirmek ve cevaplamak için service metodunu çağrılır. Servlete gelen her bir talepte sunucu yeni bir iş parçacığı görevlendirir ve Service metodunu çağırır. Service metodu, HTTP isteğinin tipini (GET, POST, PUT, DELETE) belirler buna göre doGet, doPost, doPut, doDelete metotlarını çağırır.

Aşağıda Service metodunun genel yapısı gösterilmiştir:

```
public void service(ServletRequest request,ServletResponse response)
throws ServletException, IOException{…}
```

GET çağrısını ele almak için kullanılan doGet() metodunun yapısı aşağıdaki kod parçacığında gösterilmiştir:

```
public void doGet(HttpServletRequest request,HttpServletResponse
response) throws ServletException, IOException {
//Servlet Code
}
```

POST çağrısını ele almak için kullanılan doPost() metodunun yapısı ise aşağıdaki kod parçacığında gösterilmiştir:

```
public void doPost(HttpServletRequest request,HttpServletResponse
response) throws ServletException, IOException {
//Servlet Code
}
```

2.1.5.3. Destroy Metodu

Destroy metodu ise servletin yaşam döngüsünün sonunda bir kere çağrılır. Bu metod servlete, veri tabanı bağlantısını ve arka planda çalışan iş parçacıklarını sonlandırma, çerezler listesine yazma vb. sonlandırma işlemleri gerçekleştirmesi imkânını sağlar. Bu metodun çağrılmasından sonra servlet nesnesi atık-toplama işlemi ile bellekten kaldırılarak sonlandırılır.

Aşağıda bir destroy() metodunun genel yapısı gösterilmiştir:

```
public void destroy() {
// Finalization code...
}
```

İstemci tarafından GET metodu ile gönderilen isteğin servlet tarafından ele alınışı genel olarak aşağıdaki kod bloğundaki gibidir. Ayrıca istemci her hangi bir metod belirtmediğinde de yine varsayılan olarak bu yapı kullanılır [41].

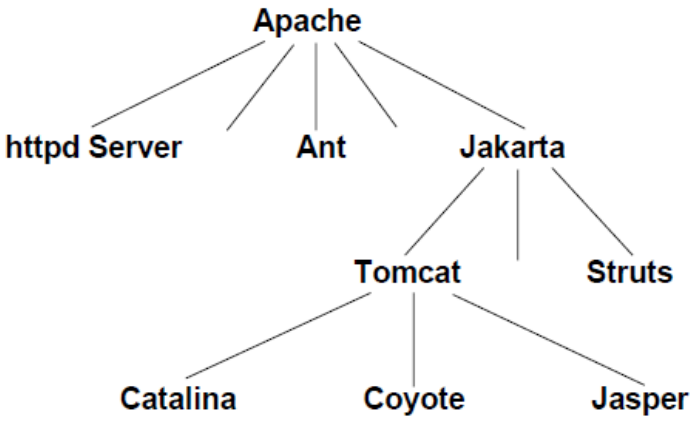
```
import java.io.*;
importjavax.servlet.*;
import javax.servlet.http.*;
public class ServletTemplate extends HttpServlet {
public void doGet(HttpServletRequest request, HttpServletResponse
response)
throws ServletException, IOException {

   // Use "request" to read incoming HTTP headers
   // (e.g., cookies) and query data from HTML forms.

   // Use "response" to specify the HTTP response status
   // code and headers (e.g., the content type, cookies).

   PrintWriter out = response.getWriter();
   // Use "out" to send content to browser.
   }
}
```

2.2. APACHE TOMCAT SUNUCUSU

Web sunucuları, HTTP gibi protokolleri kullanarak istemcilerden gelen istekleri dinleyen ve bünyesinde barındırdığı web uygulamalarını yönelten altyapı yazılımlarıdır. Web uygulamaları istemciden gelen isteği işleyip, çıktıyı üretir. Elde edilen çıktı web sunucu tarafından istemciye gönderilir. Kısaca hem statik hem de web programlama dillerinde hazırlanmış dinamik web sayfalarını istemcilere ulaştırabilmek için bir web sunucusuna ihtiyaç duyulmaktadır. Java teknolojileri birden çok web sunucusunu (Apache Tomcat, Jboss, Glass Fish, Web Sphere vb.) desteklemektedir. Hazırlanan tahliye sistemi öncelikle web sunucusu olarak Apache Tomcat üzerinde denenmiştir. Bu kısımda Apache Tomcat web sunucusu hakkında bilgiler verilmiştir.

Apache web sunucusu direkt olarak servlet sınıflarını ve JSP (Java Server Page) sayfalarını çalıştıramaz. Bu nedenle, Apache'nin bir alt bileşeni olan ve Jakarta olarak adlandırılan Tomcat sunucusu kullanılır. Tomcat Java tabanlı, açık kaynak kodlu ve tamamen ücretsiz olan bir web uygulama motorudur. Apache Software Foundation tarafından geliştirilmiştir.

Tomcat hem Java kodlarını çalıştıran Java Sanal Makinası hem servletleri hem de JSP sayfalarını çalıştıran motordur [42]. Tomcat kendi bünyesinde sunucu ile ile ilgili yönetim ve ayarlama araçları bulundurur. Bu yönetim ve ayarlama işlemini XML formatında dosyalar üzerinden de yapmak mümkündür. Şekil 2.5'de bir Apache sunucunun bileşenleri ve bu yapı üzerinde Tomcat'in konumu gösterilmiştir.

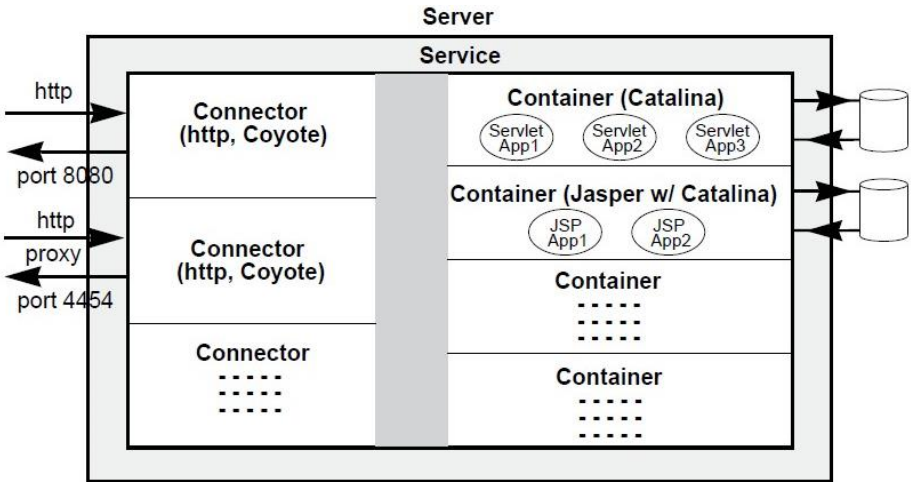


Şekil 2.5. Apache'nin ve Tomcat'in bileşenleri [43].

Catalina, Tomcat'in servlet motoru görevini yöneten kısmıdır. Coyote web bağlantısını sağlayan kısım, Jasper ise Tomcat 4.1'den sonra kullanılmaya başlanan ve JSP motoru görevini yürüten kısımdır.

2.2.1. Tomcat'in Yapısı

Tomcat, varsayılan olarak 8080 portunu dinler. Windows'ta bir servis, Linux ve Unix'te bir arka plan programı olarak çalışır. Tek bir Tomcat nesnesi birden fazla servis hizmeti sunabilir.



Şekil 2.6. Tomcat'in yapısı [43].

Her bir Tomcat servisi en az bir web bağlantı aracı (connector) ve en az bir tanede Catalina motoruna sahiptir. Server, service, connector, container ve engine nesnelerinin hepsi de esnek bir şekilde ayarlanabilir özelliklere sahiptir.

Varsayılan olarak Tomcat servisi tek bir işletim sistemi prosesi olarak çalışır. Bu proses Java sanal makinesini çalıştırır. Tarayıcıdan gelen her bir HTTP isteği ayrı birer iş parçacığı olarak bu proses içinde çalıştırılır.

2.3. ORACLE GLASSFISH SUNUCUSU

Geliştirilen tahliye sistemi kodları web sunucu olarak öncelikle Apache Tomcat üzerinde çalıştırılmış daha sonra ise Oracle Glassfish Uygulama Sunucusu'na taşınmıştır. Glassfish tercihinde Java platformunu satın alan Oracle firmasının resmi sunucu olarak Glassfish'i desteklemesi etkili olmuştur. Bu kısımda Glassfish sunucusu hakkında genel bilgiler verilmiştir.

Glassfish, Java uygulamalarının ve web servislerinin kullanıcı isteklerine cevap verebilmesini sağlayan bir uygulama sunucusudur. Assembly, diğer bir adıyla paketleme, ayrık uygulama bileşenlerinin ya da modüllerin sunucu üzerinde tek bir paket olarak yüklenmesi işlemidir. Glassfish sunucusu, uygulamaların ve modüllerin Java Enterprise Edition'ın (Java EE) özelliklerine uygun bir şekilde paketlenmesini sağlar. Ek olarak Glassfish sunucusu kendi tanımlayıcılarını da tutarak daha etkin bir sunucu hizmeti vermeye çalışır. Bu tanımlayıcılar, XML uzantı bir dosyada Java EE uygulamalarının ve modüllerinin sunucuya nasıl kurulacağı ile ilgili bilgileri tutar [44].

Modüller, bir veya daha çok Java EE bileşeninden oluşan ve aynı web konteyner üzerinde çalışan uygulama birimleridir. Glassfish sunucusunun desteklediği modüller ikisi şunlardır [45]:

- Web Modül: Web modül web uygulaması olarak da bilinir ve Java EE uygulama sunucusuna yüklenebilecek servletlerin, EJB'lerin, HTML sayfaların, sınıfların bir koleksiyonu olarak tanımlanabilir. Web Application Archive (WAR) dosyası web uygulamalarının paketlenmesinde kullanılan

standart dosya formatıdır. Bir WAR dosyası servletleri, JSP dosyalarını, statik sayfaları, istemci taraflı appletleri, enterprise bean sınıflarını, annotation ve web tanımlayıcılarını içerir.

- EJB Modül: Bir EJB modülü bir veya daha fazla Enterprise Bean ve EJB yükleme tanımlayıcısını içeren bir yazılım birimidir. Enterprise Bean'leri birleştirmek için kullanılan standart format ise Java Archive (JAR) dosyalarıdır. Bir EJB JAR dosyası bean sınıflarını içerir.

2.4. NETBEANS

Geliştirilen uygulamanın sunucu tarafını oluşturan servlet sınıfları NetBeans'de geliştirilmiş ve test edilmiştir. NetBeans, masaüstü, mobil ve web uygulamaları geliştirmeye olanak sağlayan açık kaynak kodlu, ücretsiz bir yazılım geliştirme ortamıdır. NetBeans Java, HTML5, PHP ve C++ gibi programlama dillerinin de içinde bulunduğu birçok dilde program geliştirmeyi destekler. Bir projenin geliştirilme sürecindeki; projenin oluşturulması, hata ayıklaması, test edilmesi ve yüklenmesi adımlarının hepsine olanak sağlar. Windows, Linux, Mac OS X ve diğer UNIX tabanlı sistemlerde çalışır.

NetBeans Java'daki en son yeniliklere ve JDK 8 teknolojilerine detaylı bir destek sunar. Aynı zamanda JDK 8, Java EE 7 ve JavaFX 2'ye destek sunan ilk yazılım geliştirme ortamıdır. Java, XML, Web servisleri, SQL'in en güncel standartlarını kullanarak Java EE'ye tam destek sunar.

NetBeans'de geliştirilen uygulamalar, uygulama sunucularından Glassfish'de her konuda tam olarak desteklenirken, diğer uygulama sunucularından JBoss Application, Oracle WebLogic, IBM WebSphere ve Tomcat sunucuları üzerinde de çalışabilir [46].

2.5. JDBC BAĞLANTI HAVUZU (CONNECTION POOL)

Geliştirilen uygulamada kullanıcının telefonundan sunucuya sık sık istek gönderilmekte ve bu isteklerin birçoğu veri tabanına bağlanmayı ve çeşitli sorgulamalar yapmayı gerektirmektedir. Her bir istemci tarafından gönderilen istek için sunucu üzerinde yeni bir veri tabanı bağlantı nesnesi oluşturmak, sunucu performansını olumsuz yönde etkileyen bir durumdur. Bu durum hem bağlantı için harcanan süreyi hem de bellekteki yük miktarını artırmaktadır. Geliştirilen uygulamayı, bir binanın tahliyesi sırasında yüzlerce belki de binlerce kişi kullanacağı için her bir bağlantıda yeni bir veri tabanı bağlantı nesnesi oluşturmak sisteme ağır bir yük yükleyecektir. Bu nedenle uygulamamızda veri tabanı bağlantıları için Java'nın JDBC API'lerince desteklenen JDBC Bağlantı Havuzu tekniği kullanılmıştır.

Bağlantı havuzu belirtilen bir veri tabanına bağlanmak için tekrar tekrar kullanılabilen bir grup bağlantı nesnesinden oluşur. Bağlantı havuzunda veri tabanı bağlantıları Veri Kaynağı (DataSource) nesneleri kullanılarak gerçekleştirilir. Veri Kaynağı adeta bir bağlantı fabrikası gibi çalışır ve belirtilen sayıda veri tabanı bağlantı nesnesi oluşturur. Sunucu üzerinde bir veri tabanı bağlantısı gerektiğinde Veri Kaynağı havuzundaki bağlantı nesnelerinden biri kullanılır. Veri Kaynağının bağlantı yapacağı veri tabanı sunucusu, veri tabanı ismi, ağ protokolü, kullanıcı adı ve şifre gibi bazı önceden ayarlanması gereken parametreleri vardır. Ayrıca Veri Kaynağı oluşturulurken başlangıçta en az kaç tane bağlantı nesnesi oluşturulacağı, en fazla kaç tane bağlantı nesnesine izin verileceği, havuzda kullanılmadan bekleyen bağlantı nesnelerinin ne kadar süre sonra yok edileceği gibi bazı ayarlamalar başlangıçta yapılır. Sunucu üzerinde bu bağlantı nesneleri Veri Kaynağı bağlantı havuzunda kullanılmaya hazır bekletilir. getConnection() metodu ile havuzdaki kullanılmayı bekleyen bağlantı nesnelerinden biri çağrılabilir.

```
Connection connection = datasource.getConnection();
```

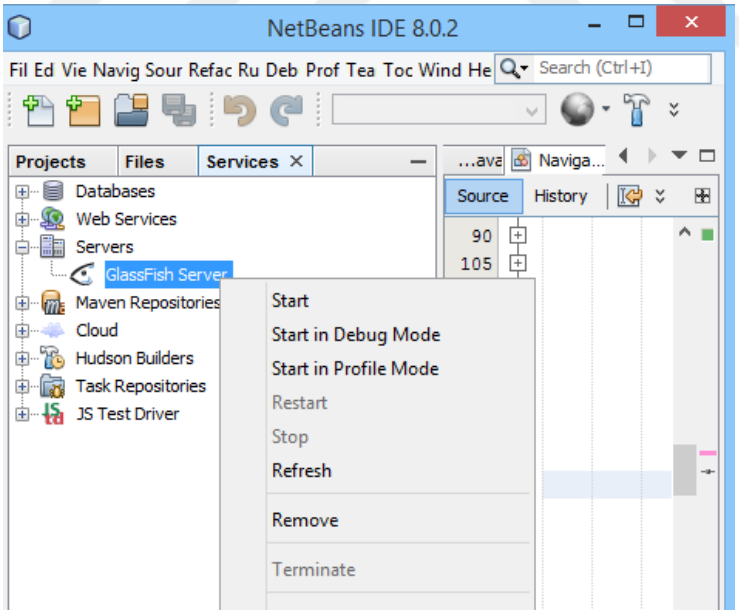
Böylece yeni bir bağlantı yapmanın sisteme getireceği yükten kurtulmuş olunur. Ayrıca Veri Kaynağı, uygulamada kapatılan her bir bağlantı nesnesini tekrar havuza

geri döndürerek atık toplama (gabrage collecting) işlemini de gerçekleştirmiş olur [47].

2.5.1. Glassfish'de Bağlantı Havuzu Kullanımı

Glassfish web sunucusu üzerinden JNDI (Java Naming and Directory Interface) Veri Kaynağı kullanılarak Oracle veri tabanına ulaşmak için öncelikle ojdbc6.jar ya da üst sürümlerine ait JDBC sürücüleri temin edilmelidir. Bu sürücü dosyası Glassfish sunucusu üzerindeki kütüphane klasörüne (*<Glassfish- 4.1>/ Glassfish/ domains/ domain1/ lib/ext)* kopyalanarak uygulamada erişilebilir bir konuma yerleştirilmiş olur. Bu ön işlemlerden sonra aşağıdaki adımlar izlenerek bağlantı havuzu tanımlanır ve uygulamada veri tabanı bağlantıları bu nesne üzerinden yapılır [48]:

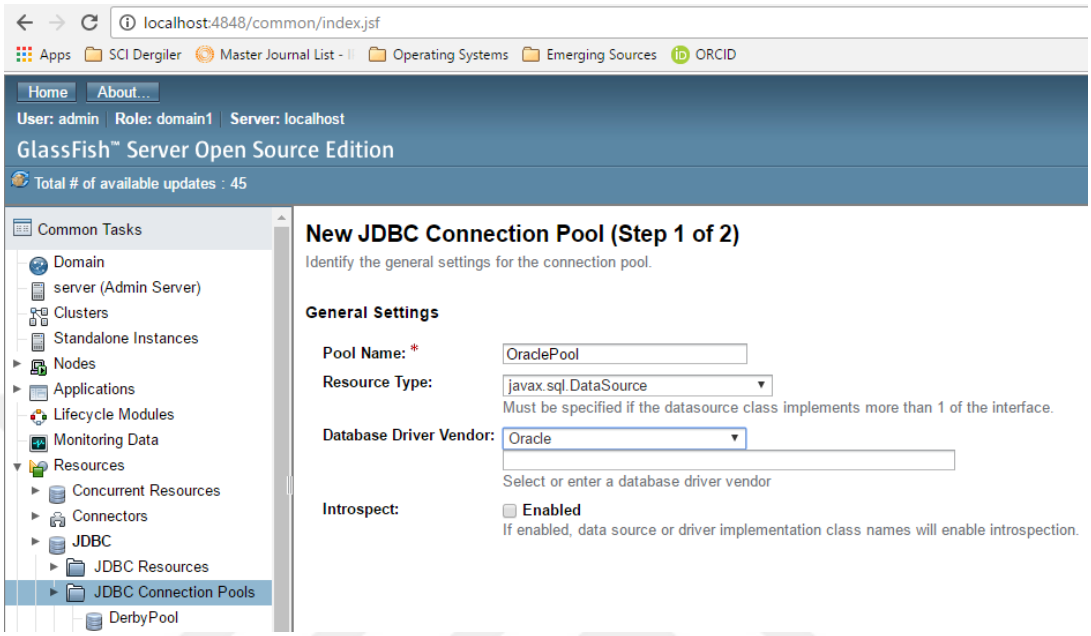
NetBeans IDE'si de Servisler sekmesinden Glassfish sunucusu başlatılır. Glassfish varsayılan olarak 8080 portu üzerinde çalışır (Şekil 2.7).



Şekil 2.7. Glassfish sunucusunu başlatılması.

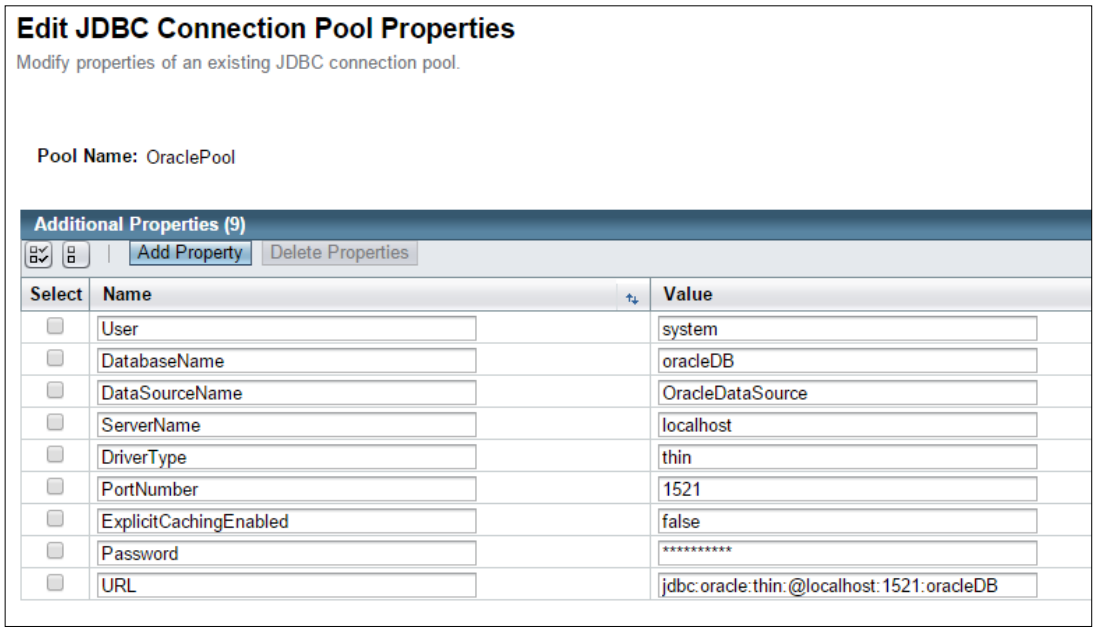
Daha sonra web tarayıcısına *http://localhost:4848/* adresi yazılarak yönetici konsoluna geçilir. Açılan pencereden Resources→JDBC→JDBC→ Connection Pools menüsü

izlenerek yeni bir bağlantı havuzu nesnesi oluşturulur. Oracle veri tabanı için seçilmesi gereken değerler Şekil 2.8'de gösterilmiştir.



Şekil 2.8. Bağlantı havuzu nesnesi oluşturulması.

Bir sonraki adımda yeni bağlantı havuzu nesnesine ait bağlantı parametreleri tanımlanıp, kaydedilerek işlem tamamlanır (Şekil 2.9).



Şekil 2.9. Bağlantı havuzu parametreleri.

Bağlantı havuzu nesnesini tanımladıktan sonra JNDI veri kaynağı tanımlamak için Resources → JDBC→JDBC Resources menusüne girip yeni bir kaynak tanımlanır. JNDI kaynağına bir isim verilir ve Pool Name olarak yukarıda oluşturduğumuz OraclePool bağlantı nesnesi seçilerek kaydedilir. Böylece uygulamada kullanılabilecek bir bağlantı havuzu elde edilmiş olur.

2.5.2. Bağlantı Havuzunun Uygulamada Kullanılması

Hazırlanan bağlantı havuzundaki nesnelerin uygulamada kullanılabilmesi için bir ön işlem olarak Servlet Container'da tanımlanması gerekmektedir. Bu işlem için Servlet Container'da bir servis hazırlanmıştır. Arka planda çalışan bu servis ServletContextListener arayüzünü (interface) kullanmaktadır. ServletContextListener, ServletContext'deki değişiklikleri (events) dinleyen ve bu değişikliklere tepki veren bir Java arayüzüdür. ServletContext ise sunucu üzerindeki servletleri yönetmek ve sunucu ile servletler arasındaki iletişimi sağlamak için bazı metotlar sunan bir Java arayüzüdür. Her bir web uygulaması için bir Context tanımlanır [49].

ServletContextListener'in uygulamadaki değişiklikleri ve bildirimleri dinleyebilmesi için geliştirilen web uygulamasının tanımlayıcı dosyası olan web.xml'de listener olarak tanımlanması gerekir. ServletContextListener'in iki soyut metodu vardır [50].

- contextInitialized(ServletContextEvent) : Web uygulaması sunucu üzerinde çalışmaya başlarken gelen bildirimleri işler.
- contextDestroyed(ServletContextEvent): ServletContext'in sonlandırılması sırasındaki son işlemlerin yapıldığı metottur.

Hazırlanan bu servisin contextInitialized metoduna şu kodlar eklenerek bağlantı nesnesi tanımlanmış olur:

```
initContext = new InitialContext();
dataSource = (DataSource) initContext.lookup("jdbc/Oracle");
s_context.setAttribute(“PoolListener”, dataSource);
```

Veritabanı bağlantısı için bu bağlantı havuzundan bir nesneyi kullanmak isteyen her bir Servlet sınıfının init() metodunda aşağıdaki kodlar çalıştırılarak datasource nesnesi hazır hale getirilmiş olur [47].

```
@Override
public void init(ServletConfig config) throws ServletException {
   super.init(config);
   datasource=(DataSource)getServletContext().getAttribute("Pool");
}
```

Datasource da tanımlandıktan sonra bir bağlantı nesnesi oluşturmak için aşağıdaki kodların çalıştırılması yeterlidir.

```
Connection connection = datasource.getConnection();
```

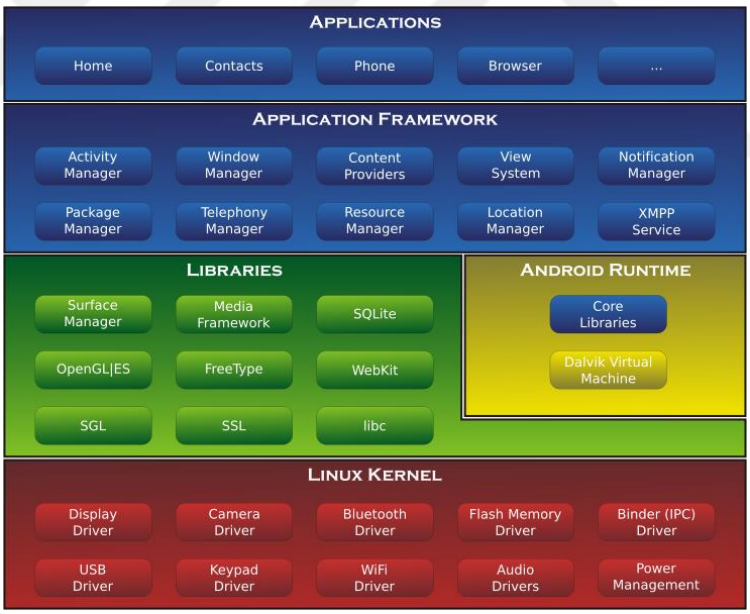
2.6. ANDROID

Android, Google, Open Handset Alliance ve özgür yazılım topluluğu tarafından geliştirilen, Linux tabanlı, mobil cihazlar ve akıllı telefonlar için geliştirilmiş, açık kaynak kodlu bir mobil işletim sistemidir [51]. Ayrıca günümüzde Android, akıllı gözlük, televizyon ve otomobillerde de kullanılan bir işletim sistemi haline gelmiştir. Android, geniş bir geliştirici kitlesine sahiptir. Android işletim sistemine sahip mobil cihazlar dünyada 190'dan fazla ülkede milyonlarca kişi tarafından kullanılmaktadır. Aylık olarak Google Play'den indirilen Android uygulama sayısı 1,5 milyarı geçmiş durumdadır. Google Play Android uygulamalarının indirilebildiği, Google tarafından organize edilen kurumsal uygulama mağazasıdır [52]. Android son yıllardaki bu hızlı gelişimi ile en hızlı gelişen mobil işletim sistemi olmuştur.

Android, kullanıcıların yanı sıra uygulama geliştiricileri için de birçok avantajlar sunmaktadır. Geliştiriciler bilgisayarlarına Android SDK (Software Development Kit) kurulumu yaparak Android uygulamaları geliştirebilmekte ve geliştirdikleri uygulamaları Google Play sanal mağazası üzerinden dünyadaki milyonlarca kişiye ulaştırabilmektedir. Android SDK, Java programlama dili ile Android platformu üzerinde uygulama geliştirmek için araçlar ve API (Application Programming Interface)'ler sunan bir geliştirme aracıdır [53]. Uygulamanın kullanıcı arayüzleri XML ortamında tasarlanabilmektedir. Android esnek yapısı ile geliştiricilere, arayüz

tasarımından cihazlarla ilgili en ince detaylara ulaşmaya kadar birçok geliştirme imkânı sunar.

Android, Linux 2.6 çekirdeği üzerine inşa edilmiş bir mobil işletim sistemidir [54]. Android işletim sisteminin çekirdeği proses, bellek ve giriş-çıkış birimi yönetimi, sürücü kontrolü ve güvenlik gibi temel konularda Linux çekirdeğini referans almıştır [55]. Ayrıca Android sadece bir işletim sistemi değil, içinde ara katman yazılımları, kütüphaneler, çeşitli servisler ve kullanıcı uygulamalarını barındıran bir mobil platformdur (Şekil 2.10). Android platformunun ara katman yazılımları, kütüphaneleri ve API'leri C dilinde kodlanmıştır. Android, derlenmiş Java kodunu çalıştırmak için Java sanal makinesinin yerine, daha basit bir formu olan Dalvik Sanal Makinesi'ni (DSM) kullanır. DSM kısıtlı bellek kapasitesine sahip ve düşük hızlardaki işlemciler için tasarlanmış bir sanal makinedir [56].



Şekil 2.10. Android platformunun mimarisi [57].

Geçmişten günümüze Android sürümlerinin gelişimi Çizelge 2.1'de gösterilmektedir:

Çizelge 2.1. Android sürümleri.

Sürüm No	Sürüm İsmi	API No	Android Desteği
1.0	-	1	Sonlandırıldı
1.1	-	2	
1.5	Cupcake	3	
1.6	Donut	4	
2.0	Eclair	5	
2.0.1		6	
2.1		7	
2.2	Froyo	8	
2.3.3	Gıngerbread	9	
2.3		10	
3.0	Honeycomb	11	
3.1		12	
3.2		13	
4.0	Ice Cream Sandwich	14	
4.0.3		15	
4.1	Jelly Bean	16	
4.2		17	
4.3		18	
4.4	Kitkat	19	Sadece Güvenlik Güncellemeleri
4.4W		20	
5.0	Lollipop	21	Devam Ediyor
5.1		22	
6.0	Marshmallow	23	
7.0	Nougat	24	
7.1		25	

Ayrıca Google Play'a kayıtlı Android cihazlar referans alınarak Android sürümlerinin 8 Mart 2017 tarihi itibari kullanım oranları Çizelge 2.2'de gösterilmektedir [58]:

Çizelge 2.2. Android sürümlerinin kullanım oranları.

Sürüm	API No	Kullanım Oranı
Gingerbread	10	% 1
Ice Cream Sandwich	15	% 1
Jelly Bean	16	% 3,7
	17	% 5,4
	18	% 1,5
KitKat	19	% 20,8
Lollipop	21	% 9,4
	22	% 23,1
Marsmallow	23	% 31,3
Nouget	24	%2,4
	25	%0,4

Android'in desteklediği resmi yazılım geliştirme ortamı IntelliJ IDEA tabanlı Android Studio'dur. Bunun yanı sıra Eclipse ve NetBeans yazılım geliştirme ortamlarında da

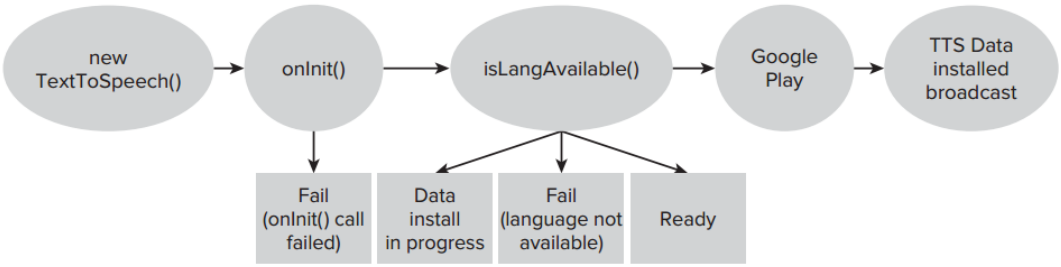
Android uygulamaları geliştirmek mümkündür. Ayrıca Microsoft .NET çatısında da Android uygulamalar geliştirilebilmektedir. Android kodları genel olarak Java programlama dili tabanlı olmakla beraber Android Studio 2.2 sürümünden itibaren Android Studio'da C++ programlama dilini de desteklenmektedir.

2.6.1. Android Text To Speech (TTS) Kütüphanesi

Bu kısımda geliştirilen tahliye sisteminin mobil uygulamasındaki sesli yönlendirme talimatlarının oluşturulduğu TTS kütüphanesi hakkında bilgiler verilmiştir.

Android'deki TTS kütüphanesi mobil cihazlarda metinlerin seslendirmesi için kullanılır ve birçok farklı dilde seslendirme imkânı sunar. Mobil uygulamada seslendirme yapabilmek için öncelikle TTS tipinde bir nesne oluşturulması gerekir. Bu işlem asenkron olarak bir iş parçacığı tarafından gerçekleştirilir. Bu iş parçacığı TTS nesnesini oluşturduğunda, TextToSpeech.OnInitListener arayüzünü (interface) çağırır. Asenkron olarak devam eden TTS nesnesinin oluşturulması sırasında seslendirme işlemi yapılamaz. TTS sınıfının kullanımı özetle şu adımlardan oluşur:

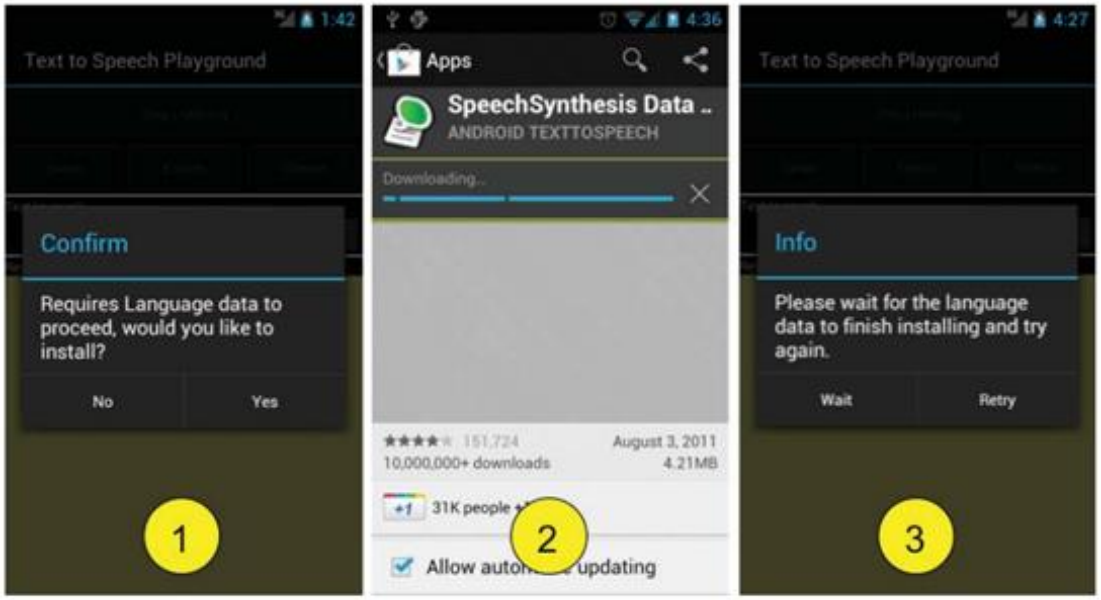
1. TTS nesnesi oluşturulur.
2. Cihazın seslendirme yapacağı dile desteğinin olup olmadığı kontrol edilir. Eğer ilgili dil desteği yok ise internet üzerinden indirilmesi sağlanır.
3. Seslendirme yapmadan önce asenkron olarak çalışan TTS motorunun TTS nesnesinin çalışmaya hazır hale getirmesi beklenir.
4. Nesne hazır olduğunda setLanguage() ile seslendirme dili belirlenir ve speak() metodu ile seslendirme yapılır.
5. Seslendirme işlemi tamamlandığında ise shutdown() metodunu çalıştırarak TTS motorunun kullandığı kaynakları serbest bırakması sağlanır.



Şekil 2.11. TTS nesnesinin kullanıma hazırlanma süreci.

TTS nesnesinin çalışmasını daha detaylandıracak olursak, öncelikle bir TTS nesnesi oluşturulur ve bu nesne ile OnInitListener arayüzü çalıştırılır. Bu durumda Android hemen asenkron çalışan TTS motorunu aktif hale getirir. TTS motoru TTS objesini hazırlayıncaya kadar TTS sınıfına yapılan bütün çağrılara cevap verilmez yani TTS sınıfının bütün metotları pasif haldedir. TTS motoru TTS nesnesini hazır hale getirdiğinde OnInitListener.onInit() metodunu çağırır. onInit() metodunda TTS nesnesinin başarılı bir şekilde başlatılıp başlatılamadığı kontrol edilir. Eğer TextToSpeech.SUCCESS gibi bir sonuç alınırsa nesnenin kullanıma hazır olduğu anlaşılır. Bir sonraki aşamaya geçilerek, mobil cihazda seslendirme yapılacak dil ile ilgili kontroller yapılır. Eğer cihazın ilgili dile desteği yok ise uygulama çalıştırılamaz. Eğer cihazın ilgili dile desteği var fakat dile ait veri dosyası eksik ise ilgili dil dosyası Google Play üzerinden cihazın dahili belleğine indilir. İndirme işlemi bittiğinde kurulum dosyası android.speech.tts.engine.TTS_DATA_INSTALLED bilgisini yayınlar.

Şekil 2.12'de kullanıcının Google Play üzerinden bir dil dosyası indirmesi gerektiğinde karşılaşacağı ekran görüntüleri yer almaktadır. Şekil 2.12.a'da uygulama, dil kontrolünde hata vermiş ve kullanıcıdan kurulum dosyalarını indirmesi istenmektedir. Şekil 2.12.b'de ise Google Play üzerinden dil dosyasının indirilmesi gösterilmektedir. Eğer kullanıcı bu indirme işlemi tamamlanmadan tekrar uygulamaya dönmeye çalışırsa Şekil 2.12.c'deki ekran ile karşı karşıya kalacaktır.



Şekil 2.12. TTS için dil dosyasının indirilme süreci.

TTS ile seslendirme işleminin kod örneği aşağıda verilmiştir:

```
private void TextToSpeechMethod(){

   tts = new TextToSpeech(context, new OnInitListener(){
   @Override
   public void onInit(int status)  {
      if (status == TextToSpeech.SUCCESS){
         int result = tts.setLanguage(Locale.US);
         if (result == TextToSpeech.LANG_MISSING_DATA ||
            result == TextToSpeech.LANG_NOT_SUPPORTED)
         Log.e("TTS", "This Language is not supported");
         else
            tts.speak("Test", TextToSpeech.QUEUE_FLUSH, null);
      }else
         Log.e(TAG, "Error creating text to speech");
      }
   });

}
```

Uygulamada TTS nesnesi kullanıma hazır hale geldiğinde speak() metodu ile seslendirme yapılır. speak() metodunun ilk parametresi seslendirilecek metin, ikincisi ise kuyruk işlemleri ile ilgili parametredir. Eğer bu parametre QUEUE_ADD ise TTS gelen seslendirme isteğini kuyrukta sıranın sonuna ekler. Eğer bu parametre QUEUE_FLUSH ise kuyruk boşaltılıp ilk sıraya gelen seslendirme isteği yerleştirilir. Yani mevcut seslendirme hemen sonlandırılıp yeni gelen seslendirme işlemine

başlanır. speak() metodunun üçüncü ve son parametresi ise playback ayarları ile ilgili parametrelerden oluşur [59].

Tez kapsamında geliştirilen mobil uygulamada kullanıcı ikişer saniyelik periyodik aralıklarla sunucudan yönlendirme talebinde bulunmaktadır. Alttaki kısımda bu periyodik işlemlerin Android'de gerçekleştirilmesi için kullanılan handler ve zamanlayıcı nesnelerinden bahsedilmiştir.

2.6.2. Handler

Handler, Android'de iş parçacıklarını yönetmek için kullanılan bir Android bileşenidir. İş parçacıklarına ait iş kuyruklarındaki Runnable nesnelerinin çalıştırılması için kullanılır. Runnable, iş parçacığı üzerinde çalıştırılacak olan kod bloğunu temsil eder. Her bir handler nesnesine bir iş parçacığına bağlanır ve handler nesnesinin yapacağı işlem bu iş parçacığı üzerinde icra edilir. Handler nesnesi tarafından yeni bir görev oluşturulduğunda ilgili iş parçacığının iş kuyruğuna eklenir. handler.post(Runnable) komutu ile Runnable nesnesi üzerindeki kodlar iş parçacığına atanmış olur. Ayrıca Android'de arka planda çalışan iş parçacıkları uygulamanın kullanıcı arayüzündeki nesnelere erişemezler. Kullanıcı arayüzü üzerindeki nesnelere erişim ve kontrol hakkına sahip tek iş parçacığı UI (User Interface) isimli ana iş parçacığıdır. Arka planda çalışan bir iş parçacığı kullanıcı arayüzü üzerinde işlem yapmak istediğinde yine Handler nesnesini kullanır [60].

2.6.3. Zamanlayıcı (Timer)

Zamanlayıcılar birçok programlama dilinde olduğu gibi Android'de de bir defa çalışacak ya da tekrarlanacak görevlerin icrasında kullanılan yazılım nesnesidir. Zamanlayıcı tarafından icra edilecek görevler Android'de TimerTask olarak adlandırılır. Bir kez çalıştırılacak görevler için belirli bir zaman tayin edilir ve o zaman geldiğinde görev icra edilir. Tekrar eden periyodik görevler ise belirli zaman aralıkları ile ardışık bir şekilde çalıştırılır. Tekrar eden bu görevler bir iş parçacığı üzerinde icra edilir. Yani aynı iş parçacığı tekrar tekrar kullanılır.

İcra edilecek görev çoklu görev içeriyorsa, yani işlemler tek bir iş parçacı üzerinden yürütülmüyorsa, Android'de Timer yerine ScheduledThreadPoolExecutor sınıfının kullanımı tavsiye edilir. ScheduledThreadPoolExecutor sınıfında da görevler belirli zaman aralıkları ile icra edilir. Yalnız her bir göreve iş parçacığı havuzundan boşta olan bir iş parçacığı atanacağı için görevlerin çalışma sıraları değişebilir. Yani ikinci sırada başlayan görev birinci sırada başlayan görevden önce tamamlanabilir [61].

Uygulamada iki saniyelik periyodlar ile yürütülen işlemin kodları aşağıda verilmiştir:

```
final Handler handler = new Handler();
Timer timer = new Timer();
TimerTask doAsynchronousTask = new TimerTask() {
  @Override
  public void run() {
     handler.post(new Runnable() {
        public void run() {
           try {
              new getNavigationData().execute(userinfo);
           } catch (Exception e) {
              // TODO Auto-generated catch block
           }
        }
     });
  }
};
timer.schedule(doAsynchronousTask, 0, 2000);
```

Bu kod üzerindeki new getNavigationData().execute(userinfo) satırı periyodik olarak tekrar edecek işlemi temsil etmektedir. Bu satırdaki getNavigationData sınıfı Android'in AsyncTask sınıfından türetilmiş bir sınıftır. Yine aynı satır üzerindeki "userinfo" parametresi ise kullanıcının gideceği yer bilgisinin yanı sıra YSA'nın giriş parametreleri olacak cinsiyet, yaş, hastalık durumu vb. parametreleri tutan bir dizidir.

Tez kapsamında geliştirilen Andorid uygulamada zamanlayıcı tarafından her iki saniyede bir çağrılan getNavigationData isimli AsyncTask nesnesi "userinfo" parametresini alarak sunucuya bağlanmakta ve sunucudan yönlendirme ile ilgili bilgileri mobil cihaza indirmektedir.

2.6.4. AsyncTask

Bu kısımda tahliye sistemi kapsamında geliştirilen Android uygulamada kullanıcıyı telefonu üzerinde yönlendirecek ortam fotoğraflarının sunucudan telefona yüklenmesinde kullanılan Asynctask sınıfını tanıtılmıştır.

AsyncTask, uygulamada arka planda verilen bir görev icra edilirken, ekranda kullanıcı ile uygulamanın etkileşiminin devam etmesini sağlayan bir Android sınıfıdır. AsyncTask, Thread ve Handler nesnelerine ihtiyaç duymadan arka planda gerçekleştirilen işlemlerin sonuçlarını ekrana getirir. AsyncTask genellikle 3-4 saniyelik kısa süreli işlemler için kullanılır. Örneğin internetten boyutu 10 MB'yi geçmeyen bir dosya indirme gibi kısa süreli bir işlem için AsyncTask kullanılabilir. Arka planda bir iş parçacığı tarafından indirme işlemi gerçekleşirken uygulamanın UI ana iş parçacığında kullanıcıya indirme işlemi ile ilgili bilgi verilir. Eğer arka planda uzun süreli çalışacak bir iş parçacığına ihtiyaç duyulursa, AsyncTask yerine java.util.concurrent paketindeki Executor, ThreadPoolExecuter ve FutureTask API'lerinden biri tercih edilebilir.

AsyncTask Params, Progress ve Result adında üç genel tipe sahiptir. Örnek bir AsyncTask nesnesinin tanımlanması aşağı verilmiştir:

```
private class SampleTask extends AsyncTask<Params,Progress,Result>{ ... }
```

AsyncTask nesnesini çalıştırmak için ise execute(Params…) metodu kullanılır. Ayrıca AsyncTask, onPreExecute, doInBackground, onProgressUpdateve onPostExecute adında dört metoda sahiptir. Bu metotlar manuel olarak dışardan çağrılamazlar. Ancak AsyncTask nesnesi çalıştırıldığında belirli bir sırayla otomatik olarak çalışırlar. Bu metotlar çalışma sırasıyla [62];

- onPreExecute(): UI ana iş parçacığı üzerinde çalışan ve arka planda çalışacak temel görev icra edilmeden önce çalışan metottur. Örneğin bu metotta kullanıcı arayüzünde bir ilerleme çubuğu (progress bar) tanımlanıp ekrana getirilebilir.

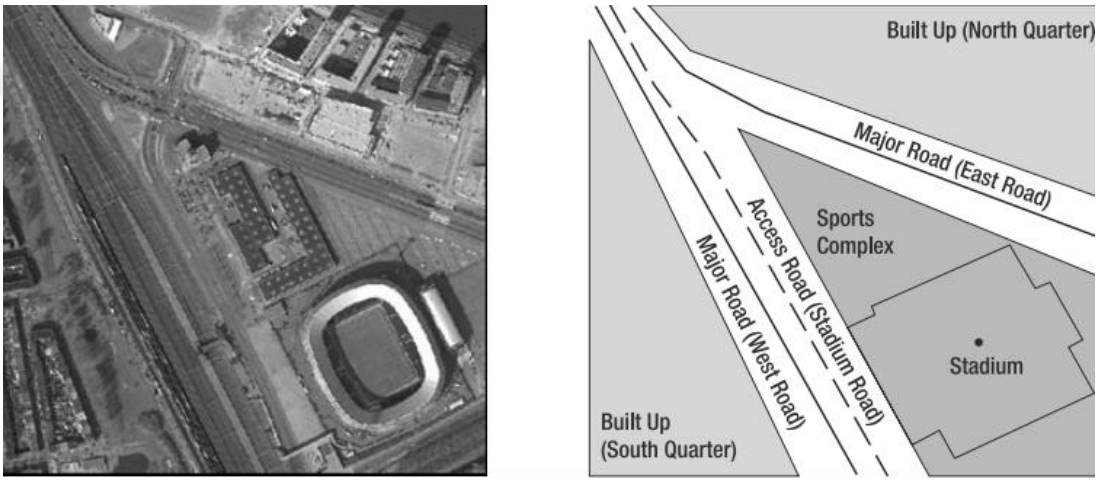
- doInBackground(Params...): onPreExecute() metodu çalışmasını tamamlar tamamlamaz başlayan ve farklı bir iş parçacığı üzerinde arka planda çalışacak temel görevi icra eden metottur. AsyncTask nesnesine parametreleri (Params...) bu metoda gönderilir. Arka planda işlemler devam ederken elde edilen sonuçlar belirli aralıklar ile publishProgress(Progress...) metodu ile onProgressUpdate metoduna gönderilir.
- onProgressUpdate(Progress...): doInBackGround metodunda işlemler devam ederken publishProgress(Progress...) çağrısı ile aktif olan bir metottur. Arka planda icra edilen görevdeki ilerlemeyi UI iş parçacığı üzerinden ekrana getirmek için kullanılır. Örneğin bir indirme işlemindeki ilerleme yüzde olarak ilerleme çubuğunda bu metotta gösterilebilir.
- onPostExecute(Result): AsyncTask'in en son çalışan metodudur. Arka plandaki görev tamamlandığında UI iş parçacığı üzerinde sonuçları göstermek için kullanılan metottur. doInBackground metodunda elde edilen sonuçlar bu metoda parametre olarak gönderilir.

2.7. ORACLE SPATIAL

Bu kısımda tezde kullanılan konumsal verilerin saklandığı Oracle'ın Oracle Spatial bileşeni ve kullanımı hakkında bilgiler verilmiştir.

CBS sistemlerinde kullanılan konumsal verilerin geleneksel veri tabanı modelleri ile saklanması ve sorgulanması birçok sıkıntıyı da beraberinde getirmekteydi. Bu sıkıntılardan kurtulmak için Oracle Veri Tabanı Yönetim Sistemi, 10g sürümüne Oracle Spatial desteği ekledi. Oracle Spatial konumsal verilerin depolanması, işlenebilmesine ve veri tabanlarındaki diğer verilerle ilişkilendirilmesine olanak sağlayan bir veri tabanı modelidir. Oracle Spatial, depolanacak konumsal nesnelerin geometrik şekillerini anlamlandırmak için semantik modeller, konumsal verilerin topolojik ilişkilerini sorgulamak için topolojik modeller, konumsal nesnelerin birbiri ile olan bağlantılarını sorgulamak için ağ modelleri ve bütün bu sorgulama ve analizleri yapmak için metotlar ve operatörler sunar. Ayrıca Oracle Spatial bünyesinde konumsal indeksleme mekanizmaları, konumsal verilerin elde edilmesi, görselleştirilmesi işlenmesi için çeşitli fonksiyonları da bulundurur [63].

Şekil 2.13.a'da orjinal resmi gösterilen bölgenin Şekil 2.13.b'de vektörel modeli gösterilmiştir. Bu vektörel model Oracle Spatial'da yer alan Nokta (Point), Çizgi (Line) ve Yüzey (Poligon) nesneleri içermektedir. Bu nesneler konumsal veri tabanında saklanabilecek tipte verilerdir. Örneğin resimde, stadyum bir nokta veri tipi ile yollar çizgi veri tipi ile bölgeler ise poligon veri tipi ile gösterilmektedir. Bu resimdeki bütün bileşenler konumsal veri tabanında bir tabloda ya da farklı katmanlara ayrılarak birden fazla tabloda saklanabilir. Örneğin ana yollar bir tabloda, farklı bir katman olan bağlantı yolları ise farklı bir tabloda saklanabilir.



(a) Orjinal resim (b) Vektörel gösterim

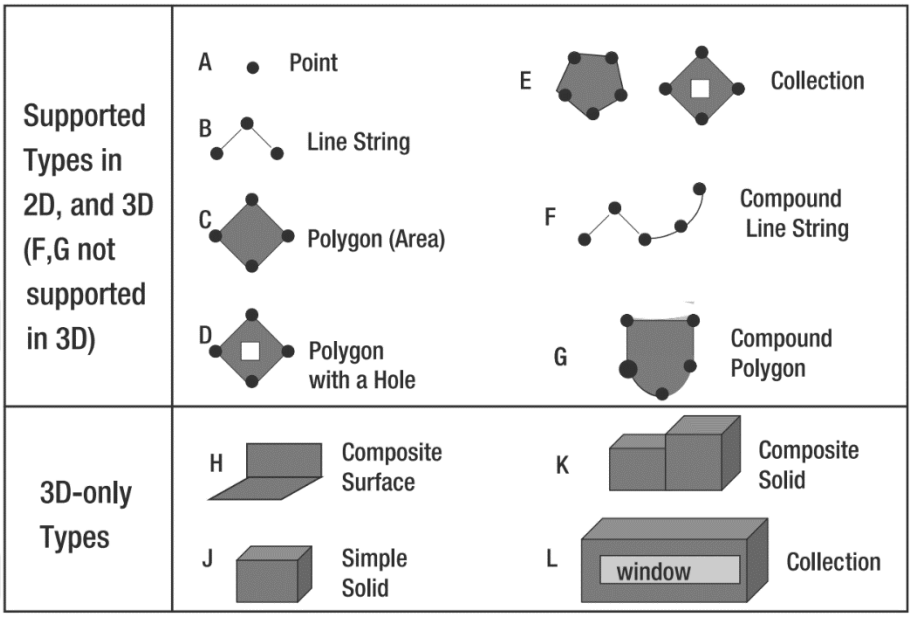
Şekil 2.13. Bir bölgenin Oracle Spatial'deki vektör gösterimi [64].

Oracle Spatial tarafından sunulan veri modelinde, Şekil 2.13'deki resim üzerinde yapılabilecek bazı sorgulama örnekleri şunlardır:

- İki nokta arasındaki en kısa mesafe
- Resimdeki bölgede kaç tane spor tesisi olduğu
- Resimdeki bölgede bulunan en büyük alış veriş merkezi
- Resimdeki bölgede bulunan bina sayısı
- Resimdeki yolların toplam uzunluğu

2.7.1. SDO_GEOMETRY Veri Tipi

Oracle Spatial veri tabanında 3B verinin depolanması için SDO_GEOMETRY veri tipi kullanılmaktadır. SDO_GEOMETRY birçok farklı konumsal veri tipini temsil etme özelliğine sahiptir (Şekil 2.14).



Şekil 2.14. Oracle Spatial'da temsil edilebilecek geometrik şekiller [64].

SDO_GEOMETRY veri tipi bir noktayı, bir çizgiyi ya da bir çizgi dizisini, bir yüzeyi, 3B bir nesneyi ya da bunların birleşimini temsil eder. SDO_GEOMETRY tipi, bu veri tiplerini temsil edebilmek için bir grup bileşenden oluşmaktadır [65,66]. Bir SDO_GEOMETRY nesnesinin temel bileşenleri ve genel olarak tanımlaması şu şekildedir:

```
CREATE TYPE sdo_geometry AS OBJECT (
SDO_GTYPE NUMBER,
SDO_SRID NUMBER,
SDO_POINT SDO_POINT_TYPE,
SDO_ELEM_INFO SDO_ELEM_INFO_ARRAY,
SDO_ORDINATES SDO_ORDINATE_ARRAY);
```

Buradaki SDO_GTYPE özelliği kullanılan geometrinin şeklini ifade eder ve şu değerleri alabilir:

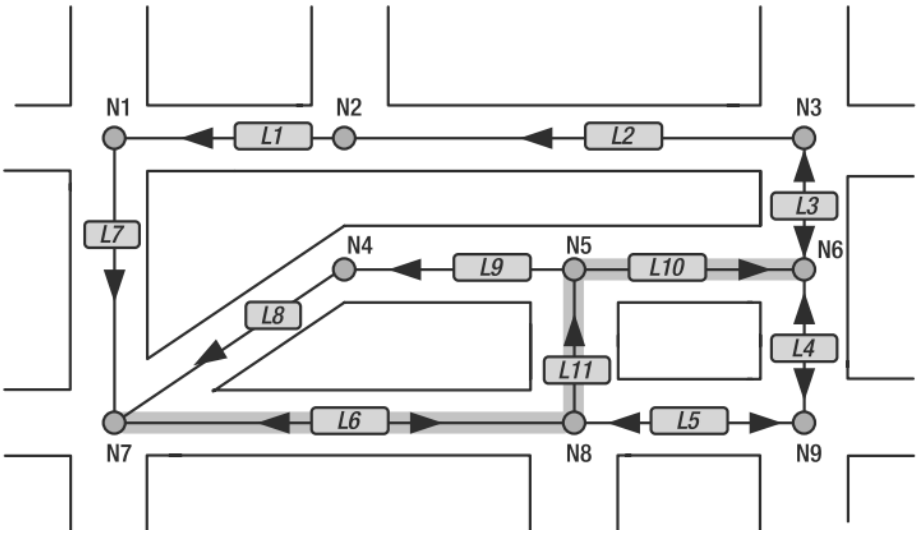
- Point (01)
- Line veya Curve (02)
- Polygon (03)
- Collection (04)
- Multipoint (05)
- Multiline veya Multicurve (06)
- Multipolygon (07)

SDO_SRID özelliği ise kullanılan konumsal referans sistemini belirler. SDO_POINT geometrik şeklin x, y, z koordinatlarına sahip noktasal verileri tanımlar. SDO_ORDINATES_ARRAY geometriyi oluşturan değişken sayıda elemanın koordinatlarını saklar. SDO_ELEM_INFO_ARRAY, SDO_ORDINATE_ARRAY elemanlarının birbirleriyle olan ilişkisini yani elemanların dizilimleri ile ilgili bilgileri tutar [66,67].

2.7.2. Ağ (Network) Modeli

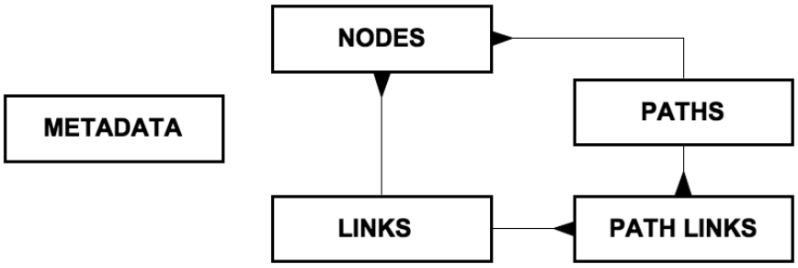
Oracle Spatial, bir ağda düğümler ve bağlantılar olarak modellenmiş nesneleri temsil etmek için ağ veri modeline sahiptir. Ağ, birbiri ile bağlantısı bulunan nesnelerin aralarındaki ilişkilerin bir tür matematiksel graf şeklinde gösterilmesidir. Şekil 2.15'de temsili bir bölgeye ait sokaklar ve bu bölgenin ağ yapısı gösterilmektedir.

Ağ, düğüm ve bağlantılar adı verilen bileşenlerden oluşur. Düğüm, bir ağdaki önemli noktaları ifade eder. Örneğin iki yolun kesişim noktası bir düğüm ile temsil edilir. Bağlantı ise iki düğüm arasındaki ilişkiyi gösterir ve iki düğümü birbirine bağlar. Bağlantılar tek yönlü ve çift yönlü olmak üzere ikiye ayrılır. Şekil 2.15'deki L11 tek yönlü bir bağlantı iken, L6 çift yönlü bir bağlantıdır.



Şekil 2.15. Ağ modelindeki düğüm ve bağlantı yapısı [64].

Yol (path), ağ üzerinde bir güzergâhı temsil eder ve iki düğüm arasında tekrar etmeyen, ardışık düğüm ve bağlantıları içerir. İki düğüm arasında tek bir yol olabileceği gibi birden fazla alternatif yollar da olabilir. Ağ modelinde bir yolun maliyeti zaman ve mesafe temelinde ölçülebilir. Bunun içinde her bir düğüm ya da linke bir maliyet parametresi eklenebilir. Oracle Spatial'da bir ağ modeli tanımlamak için veri tabanında düğüm ve bağlantı tablolarının olması ön şarttır. Bu iki tablonun isimlendirilmesi kullanıcıya bırakılmıştır. Fakat bu tablolara ait alanlardan bir SDO_GEOMETRY tipinde tanımlanmış olmalıdır. Yol ve yol bağlantıları tablosu ise isteğe bağlı olarak tanımlanabilir. Bu iki tablo Java API'leri kullanılarak yapılan analiz sonuçlarını depolamak için kullanılır.



Şekil 2.16. Ağ tabloları [64].

USER_SDO_NETWORK_METADATA tipindeki ağ modeli meta verisindeki ağdaki tabloların isimleri ve alanları tanımlanır. Örnek bir metadata tanımlaması aşağıda verilmiştir:

```
INSERT INTO USER_SDO_NETWORK_METADATA (
NETWORK,
NETWORK_CATEGORY,
GEOMETRY_TYPE,
NO_OF_HIERARCHY_LEVELS,
NO_OF_PARTITIONS,
LINK_DIRECTION,
NODE_TABLE_NAME,
NODE_GEOM_COLUMN,
LINK_TABLE_NAME,
LINK_GEOM_COLUMN,
LINK_COST_COLUMN,
)
VALUES (
'INDOOR_MODEL', -- network (primary key)
'SPATIAL', -- network_category
'SDO_GEOMETRY', -- geometry_type
1, -- no_of_hierarchy_levels
1, -- no_of_partitions
'UNDIRECTED', -- link_direction
'INDOOR_NODE', -- node_table_name
'SDO_LOCATION', -- node_geom_column
'INDOOR_LINK', -- link_table_name
'SDO_GEOM', -- link_geom_column
'LINK_LENGTH', -- link_cost_column
);
```

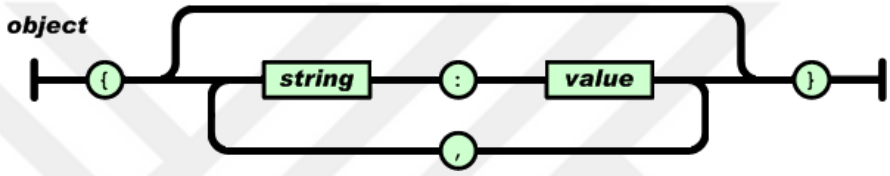
2.8. JSON (JAVASCRIPT OBJECT NOTATION)

JSON, geliştirilen mobil tahliye sisteminde sunucu ile akıllı telefonlar arasındaki veri transferi standardı olarak kullanılmıştır. JSON JavaScript tabanlı, sisteme fazla yük getirmeyen bir veri değişim formatıdır. İnsanlarca okunup yazılabilmesi, bilgisayarlar tarafından ise anlamlandırılması ve oluşturulması oldukça kolay bir standarttır. Veri boyutunun XML'e oranla çok daha küçük olması nedeniyle özellikle web uygulamalarında ve mobil uygulamalarda veri transfer standardı olarak yaygın bir şekilde kullanılmaktadır. JSON programlama dillerinden bağımsız bir veri tanımlama dili ve veri değişim formatıdır. Farklı programla dilleri arasında veri transferi için kullanılmaktadır.

JSON evrensel ve bütün modern programlama dillerinde farklı şekillerde kullanmakta olan iki farklı veri yapısına sahiptir:

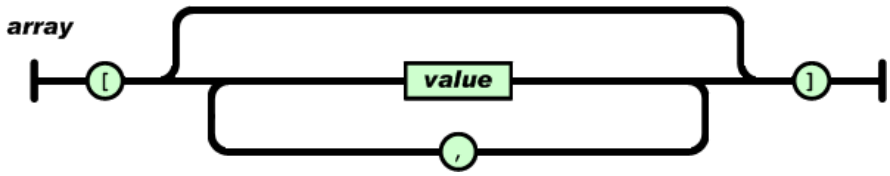
- Anahtar-Değer çifti koleksiyonu: Çeşitli programlama dillerinde bu çiftli yapı, “object, record, struct, dictionary, hash table, keyed list veya associative array” olarak da tanımlanır.
- Sıralı “değer listesi: Çoğu programlama dilinde “array, vector, list veya sequence” olarak tanımlanır.

Bir JSON nesnesi, anahtar-değer çiftlerinin sırasız birleşiminden oluşur. Nesne tanımlaması, “{” işareti ile başlar ve “}” işareti ile biter. Her anahtardan sonra “:” işareti gelir ve anahtar-değer çiftleri “,” işareti ile ayrılır (Şekil 2.17).



Şekil 2.17. JSON nesnesinin yapısı [68].

Diziler, sıralı değer listesidir. Bir dizi “[” işareti ile başlar ve “]” işareti ile biter. Değerler “,” işareti ile birbirinden ayrılır (Şekil 3.18).



Şekil 2.18. JSON dizinin yapısı [68].

Bir JSON nesnesinin değer kısmı şu veri tiplerini destekler [69]:

- Number
- String
- Boolean
- Array
- Object
- null

Bir öğrencilerin bilgilerinin tutulduğu JSON örneği aşağıda verilmiştir:

```
{
   “adi”: “Hasan”,
   “soyadi”: “UZUN”,
   “numarasi”:”200912349576”,
   “fakultesi”:”Muhendislik Fakultesi”,
   “bolumu”:”Bilgisayar Muhendisligi”,
   “dersleri”:
   [
      {“ders_adi”:”Mikroislemciler”,”kodu”:”BLM321”},
      {“ders_adi”:”Algoritmalar”, “kodu “: “BLM222”},
      {“ders_adi”:”Veritabanı”, “kodu”:”BLM224”},
      {“ders_adi”:”Elektronik”, “kodu”:”BLM220”},
   ]
}
```

BÖLÜM 3

MOBİL AKILLI TAHLİYE SİSTEMİ UYGULAMASI

Tez kapsamında yapılan bu çalışmada, kullanıcının kişisel özellikleri ile yangın durumunda binadaki değişen koşullar göz önünde bulundurularak, kullanıcıyı dinamik ve gerçek zamanlı olarak güvenli bir şekilde çıkışa kadar yönlendiren akıllı bir tahliye sistemi hazırlanmıştır. Hazırlanan bu tahliye sistemi üç temel bileşenden oluşmaktadır.

Birinci bileşen RFID teknolojisi ile kullanıcının bina içindeki konumunu tespit edip sunucuya gönderen konum belirleme modülüdür. Bu uygulama için bina içi pasif RFID etiketleri ile donatılmıştır. Bu etiketlerin koordinatları total station cihazı ile ölçülüp veri tabanına kaydedilmiştir. Yayanın tahliye işlemi boyunca, konumu bu etiketler ile tespit edilmiştir. Yaya bina içinde ilerlerken elinde taşıdığı RFID okuyucusu birer saniyelik aralıklar ile okuduğu etiket değerini sunucuya göndermektedir. Sunucu üzerindeki veri tabanı yardımı ile gelen etiketin konumunu belirlenerek kullanıcının bina içindeki konumu tespit edilmektedir.

İkinci bileşen ise merkezi bir sunucuda çalışan ve kişiye özel çıkış güzergâhları sunan navigasyon modülüdür. Bu modül üzerinde kullanıcının bina dışına tahliyesini sağlayan bir YSA modellenmiştir. Bu YSA kullanıcıya ait on kişisel özelliği ve acil durumlarda binadaki değişen fiziksel koşulları gösteren altı farklı veriyi parametre olarak alıp, kullanıcı için en uygun çıkış güzergâhını hesaplamaktadır. Binadaki fiziksel koşullar, bina içine yerleştirilmiş çeşitli sensörlerden elde edilmektedir. Bu navigasyon modülü her bir kullanıcıya, kullanıcının kendi kişisel özelliklerini dikkate alarak bir çıkış güzergâhı sunmakta ve tahliye işlemi süresince kullanıcıyı dinamik olarak yönlendirmektedir.

Sistemin üçüncü bileşeni ise tahliye sisteminin kullanıcı ile etkileşimini sağlayan ve kullanıcının akıllı telefonunda çalışan mobil navigasyon uygulamasıdır. Bu uygulama

kullanıcının yangın durumunda sunucu ile haberleşip, sesli ve görsel öğeler ile yönlendirildiği bir arayüzdür. Uygulama kullanıcının bulunduğu ortamı daha önceden çekilmiş fotoğraflar ile görselleştirmekte ve bu fotoğraflar üzerindeki görsel yönlendirme işaretleri ile kullanıcıyı yönlendirmektedir. Yine kullanıcı, yönlendirmeyi etkileyen kişisel özelliklerini bu mobil uygulama aracılığı ile sisteme tanımlamaktadır. Ayrıca bu çalışma kapsamında binada yangını simüle etmek için bir yangın simülasyonu hazırlanmıştır. Bundan sonraki kısımlarda geliştirilen tahliye sisteminin detayları adım adım anlatılmaktadır.

3.1. KONUM BELİRLEME SİSTEMİ

3.1.1. Ön Çalışma

Bu çalışma kapsamında bina içindeki yönlendirme açısından önemli olan noktalar düğüm olarak belirlenmiş ve düğümler arasındaki yollar ise bağlantı olarak tanımlanmıştır. Düğüm, bina içerisinde en az iki bağlantının kesişim bölgesi olarak tanımlanabilir. Binadaki odalar, sınıflar ve koridorlardaki bazı noktalar düğüm olarak tanımlanmıştır. Geliştirilen akıllı tahliye sisteminin uygulanacağı bina, uygulama öncesi bina içi konumlandırma işlemi için RFID etiketleri ile donatılmıştır. Etiketler, düğüm bölgelerinin tavan yüzeylerine yapıştırılarak kullanıcıların konumlarının tespitinde kullanılmaktadır. Bu çalışmada düğüm tek bir noktayı temsil eden bir koordinat değil en az bir etiketi içine alan birkaç metre karelik bir bölgedir. Düğüm bölgesinin merkezi koordinatları, bu düğümün içerdiği etiketlerin konumlarının aritmetik ortalaması alınarak hesaplanmaktadır (Şekil 3.1).

Eğer düğüm bölgesinde n adet etiket var ise, düğümün konumu Formül 3.1'deki gibi hesaplanmaktadır:

(3.1)

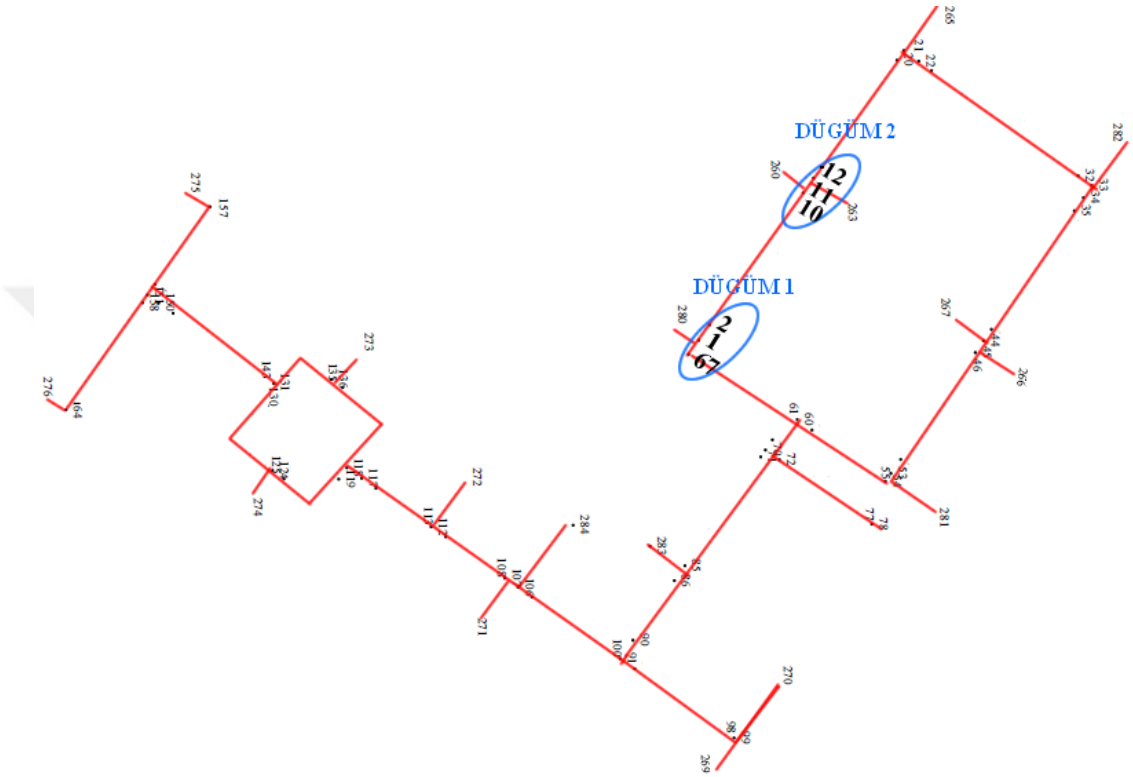
Şekil 3.1. Düğüm koordinatlarının hesaplanması.

Bu etiketler asma tavan olan mekânlarda direk tavanlara, asma tavan olmayan mekânlarda ise arkasına 3 cm'lik strafor yapıştırılarak duvarlara monte edilmiştir. RFID etiketlerin direk duvara yapıştırıldığında RFID okuyucuları tarafından sağlıklı bir şekilde okunamamaktadır. Çünkü RFID sinyalleri metal ve duvar gibi sert zeminlerden olumsuz etkilenmektedir. Bu durum RFID etiketlerinden gelen sinyallerin bozulmasına ve konum belirlemede sapmalara sebep olmaktadır [70]. Bu sebep ile asma tavan olan yüzeyler tercih edilmiş, asma tavan olmayan bölgelerde ise 3 cm'lik straforlar kullanılmıştır.

Binada odalar, merdivenlerdeki ve koridorlardaki yönlendirme açısında önemli noktalar ve çıkışlar birer düğüm olarak belirlenmiş ve etiketler ile donatılmıştır. Düğümler birden fazla bağlantının kesiştiği kritik yönlendirme noktalarıdır. Yalnız odalar birden fazla bağlantıya sahip olmayan sadece koridora bağlantısı olan düğümlerdir. Sistemin kullanıcıları tahliyesi sırasında odalarda bulunan kullanıcıların yönlendirme hizmeti alabilmesi için odalar da birer düğüm noktası olarak tanımlanmıştır. Şekil 3.2'de üzerinde çalışılan binanın birinci katına ait etiket-düğüm şeması verilmiştir.

Şekil 3.2'deki etiketlerin bazıları tek başına bir düğümü temsil ettikleri gibi bazıları da grup halinde bir düğümü temsil etmektedir. Örneğin bir koridorun girişindeki 3-4 tane

etiket bir tek düğümü temsil etmektedir. Şekil 3.2'de de görüldüğü gibi 1,2,67 numaralı etiketler 1 numaralı düğümü temsil etmektedir. Yani kullanıcının elindeki RFID okuyucu bu üç etiketten birini okumuş ise kullanıcı 1 numaralı düğümde demektir. Bazen de oda içerisindeki tek bir etiket bir düğümü temsil etmektedir.



Şekil 3.2. Binanın birinci katındaki etiket-düğüm şeması.

Bu kapsamda her bir etiketin koordinatı total station cihazı ile 3B olarak ölçülmüş (Şekil 3.3) ve Formül 3.1'e göre düğüm merkezleri hesaplanmıştır. Ayrıca birbirleri ile bağlantılı olan düğümlerin merkezlerinin birbirine olan uzaklıkları hesaplanarak binadaki bağlantıların uzunlukları tespit edilmiştir. Ölçülen etiket koordinatları, hesaplanan düğüm merkezleri ve link uzunlukları sunucudaki veri tabanına kaydedilerek bina içi konumlandırma işlemi için gerekli ön hazırlıklar tamamlanmıştır.

Şekil 3.3. Etiket konumlarının ölçümü.

3.1.2. RFID İle Konum Belirleme

Geliştirilen bu tahliye sisteminde bina içi konumlandırma işlemi kullanıcının elinde taşıdığı RFID okuyucu aracılığı ile gerçekleştirilmektedir. RFID, bir tanımlama bilgisini (ID) radyo dalgaları ile ileten sistemleri tanımlamak amacıyla ifade edilen genel bir terimdir [71]. RFID sistemlerinde, veri akıllı kart sistemlerindeki gibi elektronik bir etiket üzerinde saklanır. Etiketlerde saklanan veriyi alabilmek için bir okuyucuya ihtiyaç vardır. Okuyucu ile etiket arasındaki veri değişimi manyetik veya elektromanyetik dalgalarla sağlanır [72].

RFID etiketleri kullandıkları elektrik gücünün kaynağına göre aktif ve pasif olmak üzere iki genel kategoriye ayrılır. Aktif etiketler, genellikle güç kaynağı olarak bir pile sahiptirler. Pasif etiketler ise güçlerini okuyucunun sinyalinden alırlar [73]. Hazırlanan konum belirleme sisteminde pasif etiketler kullanılmıştır.

RFID ile konum belirleme yöntemleri iki temel sınıfa ayrılabilir. Birincisi etiketlerin hareketli okuyucuların sabit olduğu model, ikincisi ise etiketlerin sabit okuyucuların

hareketli olduğu modeldir [3]. Bu tez kapsamında geliştirilen konum belirleme sisteminde, Demiral'ın yüksek lisans tezinde geliştirdiği, etiketlerin sabit okuyucuların hareketli olduğu model kullanılmıştır [70].

Bu modele göre kullanıcının konumu, elinde taşıdığı RFID okuyucu ile coğrafi yakınlık yaklaşımına bağlı olarak tespit edilmektedir. Coğrafi yakınlık yaklaşımında RFID okuyucunun okuduğu etiket ile aynı konumda olduğu kabul edilmektedir. Okuyucu birden fazla etiketin etki alanına girmişse, bu durumda okuyucunun en iyi sinyal aldığı etiket ile aynı konumda olduğu kabul edilmektedir. Bu yaklaşım en basit ve uygulaması en kolay yaklaşımlardan bir tanesidir. Bu yaklaşıma göre en iyi sinyal alınan etiketin ID'si sunucuya gönderilmektedir. Sunucudaki veri tabanında bu ID'ye sahip etiketin hangi düğüme bağlı olduğu bulunarak, kullanıcının konumu tespit edilmektedir. Konum belirlemede hata oranı etiketlerin bina içerisine seyrek yerleştirilmesine bağlı olarak yüksek çıkmaktadır. Çünkü binada sadece düğüm bölgeleri etiketlerle donatılmıştır. Binadaki düğümler arası mesafe ortalama 6,46 metre olduğu için kullanıcının tam olarak bulunduğu noktayı tespit etmek yerine bulunduğu bölgeyi tespit etmek, geliştirdiğimiz uygulamanın konum tespiti açısından yeterlidir.

3.1.3. Konum Verisinin Tahliye Sistemine Aktarımı

RFID okuyucu tarafından elde edilen konum bilgisinin tahliye sistemiyle bütünleştirilmesinde üç farklı yöntem kullanılabilir.

3.1.3.1. Bluetooth Bağlantısı

Bu yöntemde kullanıcını yönlendirme sırasında taşıdığı RFID okuyucu ile akıllı telefonu Bluetooth üzerinden birbirlerine bağlanacak ve RFID okuyucudan akıllı telefona konum verisi periyodik aralıklarla aktarılacaktır (Şekil 3.4). Bu yöntemin kullanılabilmesi için hem RFID okuyucunun hem de kullanıcının akıllı telefonunun Bluetooth teknolojisini desteklemesi gerekmektedir.

Şekil 3.4. Bluetooth ile bağlantı.

Android platformu, Bluetooth teknolojisi ile kablosuz veri alışverişini desteklemektedir. Bluetooth cihazların birbirlerine bağlanması Android kütüphanelerindeki Bluetooth API'leri ile gerçekleştirilir. Bu API'ler noktadan noktaya ya da çoklu kablosuz bağlantıları destekler.

Bluetooth API'lerini kullanan bir Android uygulaması aşağıdaki işlemleri yapabilir:

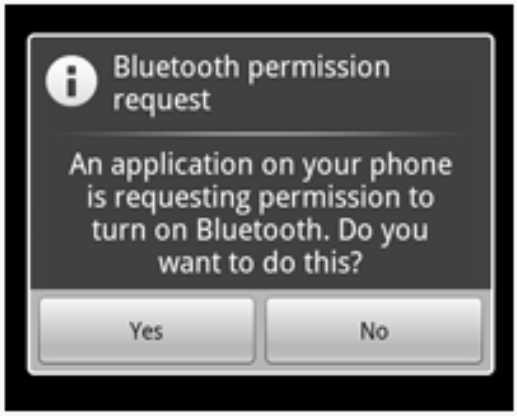
- Ortamda var olan diğer Bluetooth cihazların taranması
- Cihaz üzerindeki Bluetooth adaptöründe daha önceden eşleşmiş Bluetooth cihazların sorgulanması
- RFCOMM (Radio Frequency Communication) kanalını kurma (RFCOMM, Bluetooth cihazların haberleşmesinde kullanılan basit bir haberleşme protokolüdür.)
- Servis bulma (service discovery) yoluyla diğer cihazlara bağlanma
- Diğer cihazlarla veri alışverişinde bulunma
- Çoklu bağlantıları yönetebilme

Android platformundaki android.bluetooth paketinde bütün Bluetooth API'leri mevcuttur. Uygulamada bağlantı talebinde bulunma, bağlantı isteğini kabul etme ya da

veri transferini yapma işlemi gibi Bluetooth iletişim işlemlerini gerçekleştirebilmek için uygulamanın Manifest dosyasında BLUETOOTH izni verilmelidir.

```
<manifest ... >
   <uses-permission android:name="android.permission.BLUETOOTH" />
   <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
</manifest>
```

Uygulamada Bluetooth bağlantısı yapmadan önce cihazın Bluetooth desteği olup olmadığını kontrol edilmelidir. Cihazın Bluetooth desteği varsa, Bluetooth özelliğinin aktif olup olmadığı sorgulanmalı, eğer aktif değilse aktifleştirmek için kullanıcı onayı istenmelidir (Şekil 3.5).



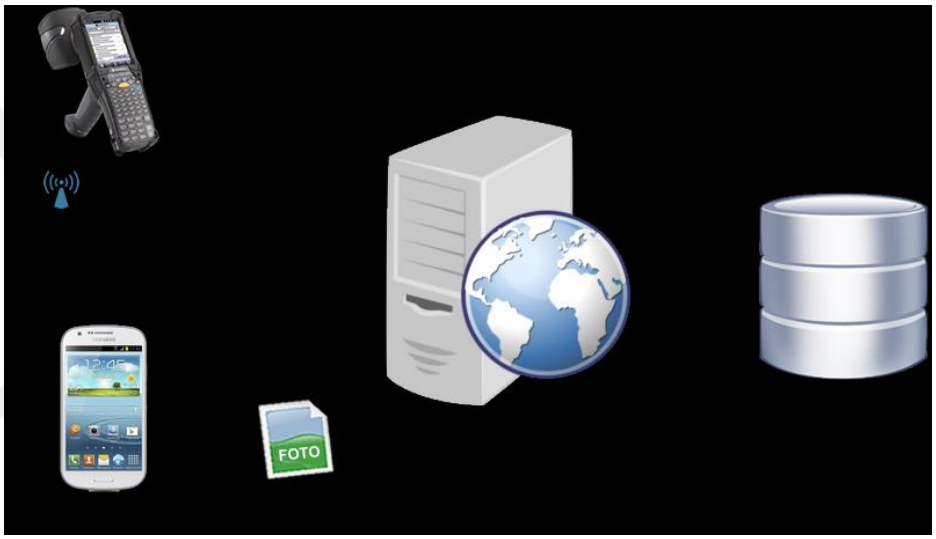
Şekil 3.5. Bluetooth aktifleştirme izin ekranı [74].

Android.bluetooth paketindeki BluetoothAdapter sınıfı kullanılarak diğer Bluetooth cihazlar bulunup eşleştirme yapılabilir.

Sunucu-istemci mimarisinde olduğu gibi eşleştirilen cihazlardan biri sunucu diğeri ise istemci durumundadır. Sunucu rolünde olan cihaz, Sunucu Bluetooth Soketini (Server Socket) oluşturur ve açar. İstemci rolündeki cihaz ise bu sokete bağlanarak bağlantıyı başlatmış olur. Aynı RFCOMM kanalındaki BluetoothSocket'a bağlanmış olan sunucu ve istemci cihazlar veri alışverişine hazır hale gelir. Bu noktadan sonra InputStream ve OutputStream sınıfları kullanılarak veri alış verişi gerçekleştirilir [74].

3.1.3.2. Kablosuz Ağ İle Noktadan Noktaya Bağlantı

Bu yöntemde Android cihazların noktadan noktaya bağlantı (Peer to Peer (P2P)) desteğinden faydalanılmaktadır. Yayanın üzerinde bulundurduğu RFID cihazı ile akıllı telefon Wi-Fi P2P özelliği ile birbirlerine bağlanacak ve RFID cihazından akıllı telefona konum verisi periyodik aralıklarla aktarılacaktır (Şekil 3.6). Wi-Fi P2P teknolojisi, Bluetooth'a göre daha geniş alanda cihazları birbirine bağlama özelliğine sahiptir.



Şekil 3.6. Wi-Fi ile bağlantı.

Wi-Fi P2P özelliği Android 4.0 ve daha üst sürümlerde çalışmaktadır. Wi-Fi P2P arada herhangi bir erişim noktasına (access point) ihtiyaç duymadan iki Android cihazı birbirine bağlama imkânı sunmaktadır. Wi-Fi P2P API'leri kullanılarak Bluetooth'un sağladığı etki alanından çok daha geniş bir alanda yine Wi-Fi P2P özelliğine sahip diğer cihazlar ile bağlantı sağlanabilir. Böylece cihazlar arasında veri paylaşımı yapılabilir.

Wi-Fi P2P API'leri aşağıdaki temel bileşenleri içerir:

- Cihazların birbirlerini tespit etmesi, birbirlerine bağlantı talebinde bulunması ve bağlanması işlemlerini içeren metotlar WifiP2pManager sınıfında tanımlanmıştır.
- WifiP2pManager sınıfına ait metotlardan biri çağrıldığında bu işlemin sonucunu dinleyen dinleyiciler (listener), çağrılan metoda parametre olarak gönderilir.
- Bağlantının kopması ya da yeni bir cihazın bulunması gibi durumlarda Wi-Fi P2P çatısı bu olayları tespit edip yeni işlemler yapmaya olanak sağlar (Intent).

WifiP2pManager sınıfı cihazlar üzerindeki Wi-Fi donanımları ile etkileşime geçerek cihazların tespiti ve onlara bağlanma gibi metotlar içerir (Çizelge 3.1).

Çizelge 3.1. Wi-Fi P2P metotları.

Metot	**Tanımı**
initialize()	Uygulamanın Wi-Fi çatısına kaydını yapar Herhangi bir Wi-Fi P2P metodu çalıştırılmadan çağrılmalıdır.
connect()	Belirtilen bir cihazdan diğer bir cihaza P2P bağlantı sağlar.
cancelConnect()	Cihazlar arasındaki P2P bağlantıyı iptal eder.
requestConnectInfo()	Cihazın bağlantı bilgisini talep eder.
createGroup()	Çağıran cihazı grup sahibi yaparak bir P2P grup oluşturur.
removeGroup()	P2P grubunu sonlandırır.
requestGroupInfo()	P2P grup bilgisini talep eder.
discoverPeers()	Cihaz arama (tespit) işlemini başlatır.
requestPeers()	Mevcut tespit edilmiş cihazların listesini talep eder.

Wi-Fi P2P çatısı yapılan çağrıları anlamlandırabilmesi için WifiP2pManager metotlarına bir dinleyici parametresi eklenir. Bu dinleyici arayüzleri (interface) Çizelge 3.2'de gösterilmiştir.

Çizelge 3.2. Wi-Fi P2P dinleyicileri.

Dinleyici Arayüzler	**Bağlantılı Metotlar**
WifiP2pManager.ActionListener	connect(), cancelConnect(), createGroup(), removeGroup(), and discoverPeers()
WifiP2pManager.ChannelListener	initialize()
WifiP2pManager.ConnectionInfoListener	requestConnectInfo()
WifiP2pManager.GroupInfoListener	requestGroupInfo()
WifiP2pManager.PeerListListener	requestPeers()

Wi-Fi P2P API'leri, bir cihazın Wi-Fi durumu değişmesi veya yeni bir cihaz tespit edilmesi gibi durumlarda Intent yayınlarlar (Çizelge 3.3). Uygulamaya yayın alıcıları eklenerek bu Intent'ler değerlendirilir.

Çizelge 3.3. Wi Fi P2P Intent'leri.

Intent	**Tanımı**
WIFI_P2P_CONNECTION_CHANGED_ACTION	Cihazın Wi-Fi bağlantı durumu değiştiğinde yayınlanır.
WIFI_P2P_PEERS_CHANGED_ACTION	discoverPeers() metodu çağrıldığında yayınlanır. Cihaz listesinde bir değişiklik olup olmadığını gösterir.
WIFI_P2P_STATE_CHANGED_ACTION	Cihazdaki Wi-Fi P2P özelliği aktif ya da pasif olarak değiştirilirse yayınlanır.
WIFI_P2P_THIS_DEVICE_CHANGED_ACTION	Bir cihazın ismi gibi detayları değiştiğinde yayınlanır.

Bu üç API bileşeni (WifiP2pManager, Listener, Intent) genellikle birlikte kullanılır. Örneğin discoverPeers() metoduna WifiP2pManager.ActionListener parametre olarak gönderilebilir. Böylece ActionListener.onSuccess() ve ActionListener.onFailure() metotları ile yeni cihazların tespit edilip edilemediği değerlendirilebilir. Eğer discoverPeers() metodu yeni cihazlar tespit etmiş ise WIFI_P2P_PEERS_CHANGED_ACTION Intent'ini yayınlanır. Uygulamada Wi-Fi P2P özelliğini aktifleştirmek için Manifest dosyasında aşağıdaki izinlerin eklenmesi gerekmektedir [75].

```
<manifest ... >
<uses-sdkandroid:minSdkVersion="14"/>
<uses-permissionandroid:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permissionandroid:name="android.permission.CHANGE_WIFI_STATE"/>
<uses-permissionandroid:name="android.permission.CHANGE_NETWORK_STATE"/>
<uses-permissionandroid:name="android.permission.INTERNET"/>
<uses-permissionandroid:name="android.permission.ACCESS_NETWORK_STATE"/>
</manifest>
```

3.1.3.3. RFID Cihazı İle Doğrudan Sunucuya Bağlanma

Bu yöntemde ise RFID okuyucusu, konum bilgisini yayanın akıllı telefonuna göndermek yerine HTTP protokolü kullanılarak periyodik aralıklar ile direkt olarak sunucuya göndermektedir. Sunucu üzerinde, kodlanan bir web servisi ile RFID

okuyucudan gönderilen konum bilgisi sisteme dahil edilmektedir. Sunucu RFID okuyucu ve akıllı telefon ile farklı kanallar üzerinden haberleşmektedir.

Şekil 3.7. RFID ile doğrudan sunucuya bağlanma.

Kullanıcının tespit edilen konumu, okuyucu üzerinde çalışan mobil bir uygulama aracılığı ikişer saniye aralıklarla sunucuya gönderilmektedir. Bu mobil uygulama HttpWebRequest nesnesini kullanarak sunucu ile iletişim kurmakta ve konum verilerini "POST" metodu ile sunucuya göndermektedir. Sunucudan gelen cevap ise WebResponse nesnesi ile ele alınmaktadır. Okuyucu ile sunucu arasındaki bağlantı hem WIFI internet bağlantısı hem de telefondaki internet paketi kullanımı ile gerçekleştirilebilmektedir. Böylece kullanıcının konumu sunucuda çalışan navigasyon modülüne iletilmiş olmaktadır.

Geliştirdiğimiz tahliye sisteminde bu üç yöntemden sonuncusu tercih edilmiştir. Çünkü ilk iki yöntemde akıllı telefon ile RFID okuyucunun direk haberleşmesi gerekmektedir. RFID okuyucu üzerinde Microsoft firmasının Windows Mobile işletim sistemi, akıllı telefonda ise Linux tabanlı Android işletim sistemi yüklüdür. Bu iki farklı platform arasındaki uyuşmazlıklardan kurtulmak ve veri transferinde ortaya çıkabilecek muhtemel kayıpları önlemek için üçüncü yöntem tercih edilmiştir.

3.2. SUNUCU NAVİGASYON MODÜLÜ

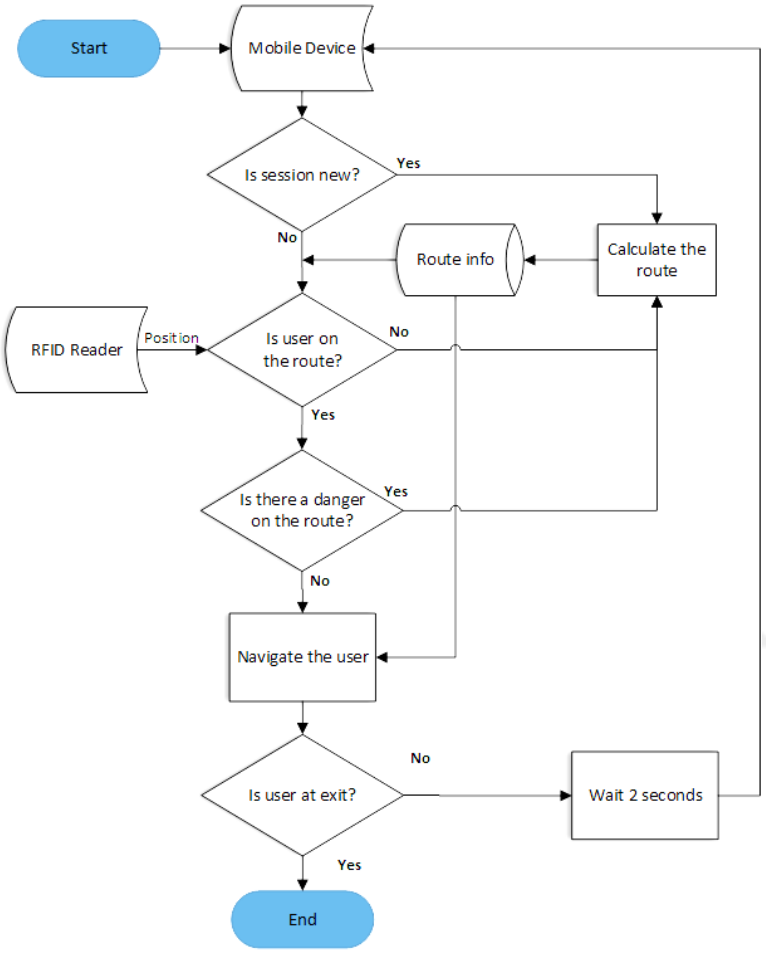
Geliştirilen akıllı tahliye sistemindeki işlemlerin büyük bir kısmı sunucu üzerinde çalışan navigasyon modülü tarafından gerçekleştirilmektedir. Bu navigasyon modülü kullanıcının mobil cihazından gelen kişisel bilgiler, bina içindeki sensörlerden elde edilen veriler ve RFID okuyucudan gelen konum bilgisini kullanarak gerçek zamanlı bir yönlendirme yönergesi oluşturmaktadır. Aynı zamanda bu yönergenin oluşturulmasında, acil durumlarda bina içi bağlantılardaki yangın, duman, görüş mesafesi, insan yoğunluğu gibi değerleri ölçen sensör verileri de dikkate alınmaktadır. Geliştirilen uygulamada binanın akıllı bir bina olduğu ve gerekli sensörlere sahip olduğu kabul edilmiştir. Bu kapsamda yine sunucu üzerinde bir yangın simülasyonu kodlanmış ve hazırlanan yangın senaryosu doğrultusunda sensörlerden veri geldiği kabul edilmiştir. Kullanıcıya ait fiziksel özellikler ve binadaki sensörlerden geldiği var sayılan değerler yönlendirmeyi etkileyen parametrelerdir. Bu parametreler Çizelge 3.4'de verilmiştir:

Çizelge 3.4. Yönlendirmeyi etkileyen parametreler.

No	Parametre	Tipi
1	Yaş	Kullanıcı Parametreleri (Kullanıcının Kişisel Özellikleri)
2	Cinsiyet	
3	Vücut Tipi	
4	Fiziksel Engel	
5	Binaya Aşinalık	
6	Eklem, Kas Rahatsızlıkları	
7	Kalp Rahatsızlığı	
8	Solunum Yolları Rahatsızlığı	
9	Gaz Maskesi	
10	Yangın Giysisi	
11	Sıcaklık	Bağlantı Parametreleri
12	Görüş Mesafesi	
13	Karbon Monoksit Oranı	
14	Yangın Büyüme Oranı	
15	İnsan Yoğunluğu	
16	Bağlantı Uzunluğu	

Kullanıcı akıllı telefonundan navigasyon modülüne ilk defa bağlantı yaptığında, yeni bir oturum oluşturulmaktadır. Sunucuya yapılan ilk bağlantıda kullanıcıya ait kişisel özellikler mobil uygulamadan navigasyon modülüne aktarılmaktadır. Navigasyon modülü, kullanıcının veri tabanındaki o anki konumunu başlangıç noktası olarak alıp

en kısa mesafeye sahip tahliye güzergâhını belirlemektedir. Bu sırada RFID okuyucu birer saniyelik zaman aralıklarıyla kullanıcının konumunu sunucudaki veri tabanına göndermektedir. Sistemdeki gerçekleşen iş akışı genel olarak Şekil 3.8'de verilmektedir.



Şekil 3.8. Tahliye sisteminin iş akış diyagramı.

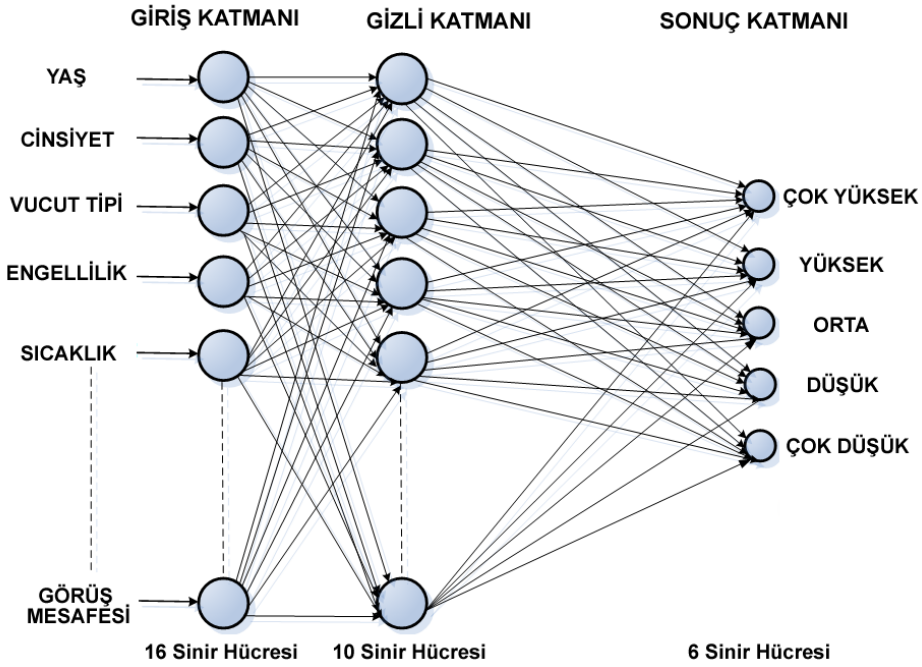
Navigasyon modülünün ürettiği tahliye güzergâhı, kullanıcının takip etmesi gereken bağlantıları ve düğümleri içeren bir dizidir. Bu bağlantılar yangın ve yangının etkileri açısından kullanıcının tahliye edilmesine engel olmayacak özellikte olmalıdır. Kullanıcı, sistem tarafından üretilen talimatlara uyduğu sürece bina dışına tahliye edilinceye kadar ilk üretilen tahliye güzergâhını kullanmaktadır. Eğer zamanla tahliye güzergâhındaki bağlantılarda yangın veya etkileri görülmeye başlanırsa ve bu durum kullanıcı için risk teşkil ederse, sistem kullanıcı için daha güvenli yeni bir tahliye güzergâhı hesaplamaktadır. Bunun yanı sıra, kullanıcı yönlendirme talimatına

uymayıp güzergâh dışına çıkarsa, sistem kullanıcının yeni konumunu başlangıç noktası olarak alıp yeni bir tahliye güzergâhı oluşturmaktadır.

Tahliye güzergâhı belirlenirken navigasyon modülünde kullanıcıya ait kişisel özelliklerin yanı sıra binadaki bağlantılara ait veriler, tahliye işleminde etkin role sahip YSA'nın giriş parametreleridir. YSA bu giriş parametrelerine göre binadaki her bir bağlantının kullanıcı için risk seviyesini belirlemektedir.

3.2.1. YSA Destekli Akıllı Tahliye Sistemi

Geliştirilen sunucu navigasyon modülü YSA ile desteklenerek akıllı bir tahliye sistemine dönüştürülmüştür. Bu sistemde, binada yangın çıkması gibi acil durumlarda ortamın fiziksel şartlarını ve tahliye edilecek olan kullanıcının kişisel özelliklerini birer parametre olarak alan ve bina içindeki bütün bağlantıların kullanılabilirliğini araştıran, Atila'nın doktora tezinde önerdiği, ileri beslemeli geri yayılımlı çok katmanlı algılayıcı YSA modeli kullanılmıştır [63]. Bu YSA giriş katmanı, gizli katman ve çıkış katmanı olmak üzere üç katmanlı bir mimariye sahiptir (Şekil 3.9).



Şekil 3.9. YSA'nın yapısı.

Giriş katmanında, kullanıcının kişisel özelliklerini ve binaya ait fiziksel şartları temsil eden 16 parametre, 16 farklı yapay sinir hücresi ile temsil edilmektedir. Öncelikle bu 16 parametrenin alabileceği gerçek değerler belirlenmiş ve bu değerler ilgili yapay sinir hücrelerinin alacağı giriş değerlerine dönüştürülmüştür. Dönüştürülmüş her bir giriş değeri aynı zamanda yangın açısından riski temsil etmektedir. Yapay sinir hücrelerine ait gerçek giriş değerleri ve bu değerlere ait risk seviyelerini gösteren dönüştürülmüş risk değerleri Çizelge 3.5'de verilmiştir.

Çizelge 3.5. YSA'nın giriş parametrelerinin değerleri.

No	Parametre	Gerçek Değer	Dönüştürülmüş Risk Değeri	Ağırlık (Ceza) Değeri
1	Yaş	18-40, 40-60, 60 üstü	1,2,3	272
2	Cinsiyet	Erkek / Kadın	1,2	124
3	Vucut Tipi	Atletik, Normal, Kilolu	1,2,3	366
4	Engellilik Durumu	Yok / Var	1,2	1251
5	Binaya Aşinalık	Yok / Var	2,1	579
6	Eklem-Kas Problemi	Yok / Var	1,2	504
7	Kalp Hastalığı	Yok / Var	1,2	258
8	Solunum Yolu Hastalığı	Yok / Var	1,2	1096
9	Gaz Maskesi	Yok / Var	2,1	785
10	Yangın Koruyucu Giysi	Yok / Var	2,1	690
11	Sıcaklık C°	0-45, 45-75, 75-150, 150 üstü	1,2,3,4	840
12	Görüş Mesafesi (metre)	0-10, 10-30, 30 üstü	3,2,1	964
13	Karbon Monoksit Yoğunluğu (ppm-particle per million)	0, 0-100, 100-6400, 6400-12600, 12600 üstü	1,2,3,4,5	753
14	Yangın Yayılma Oranı (kW/sn2)	0, 0.0001-0.0058, 0.0058-0.024, 0.024-0.094, 0.094 üstü	1,2,3,4,5	851
15	Insan Yoğunluğu (insan sayısı/m^2)	0-0.8, 0.8-1.8, 1.8-4, 4 üstü	1,2,3,4	667
16	Bağlantı Uzunluğu (metre)	0-10, 10-30, 30 üstü	1,2,3	603

YSA'nın gizli katmanında on sinir hücresi kullanılırken, çıkış katmanında ise her biri farklı bir risk seviyesini temsil eden beş farklı sinir hücresi kullanılmıştır. Bu YSA girilen 16 farklı parametreye göre binadaki bütün bağlantıların ilgili kullanıcı için hangi risk grubuna girdiğini tahmin etmektedir. Risk grupları bağlantının kullanım açısından ne kadar risk taşıdığını göstermektedir. YSA tarafından bina içindeki bütün

bağlantıların risk grupları belirlendikten sonra, navigasyon modülü kullanıcı için en az riski taşıyan bağlantıları içeren tahliye güzergâhını belirlemektedir.

Bu kapsamda belirtilen 16 parametreye ceza puanı niteliğinde bir ağrılık değeri atanmıştır (Çizelge 3.5). Ağırlık değeri bir anlamda ilgili parametrenin yönlendirmeyi ne derece etkilediğini de göstermektedir. Bu ağırlık değerleri, yangın tahliye sistemleri konusunda bilgili 78 kişinin katılımı ile gerçekleşen bir anket sonucunda elde edilmiştir. Bu ankette, katılımcılar sistemin 16 parametresinden her birine, yangında tahliye işlemini etkileme konusundaki önem derecelerine göre 1 ile 16 arasında bir puan vermişlerdir. YSA'da her bir bağlantının ilgili kullanıcı için toplam risk değeri Formül 3.2'deki hesaplanmaktadır.

(3.2)

YSA, bu parametreleri göz önüne alarak söz konusu kullanıcı için binadaki bütün bağlantıları beş risk gruplarına ayırmaktadır. Bu gruplandırmanın sonunda, navigasyon modülü kullanıcı için en az riske sahip bağlantılar içeren güzergâhı hesaplamaktadır. YSA'nın bağlantıları sahip oldukları risk değerlerine göre gruplandırmasındaki eşik değerler Çizelge 3.6'da gösterilmiştir.

Çizelge 3.6. Risk grupların sınıflandırılmasındaki eşik değerler.

Toplam Risk Değeri	**Risk Grubu**
≤ 12.000	ÇOK DÜŞÜK
> 12.000 ve ≤ 13.000	DÜŞÜK
> 13.000 ve ≤ 14.000	ORTA
> 14.000 ve ≤ 15.000	YÜKSEK
> 15.000	ÇOK YÜKSEK

Söz konusu 16 parametreye bağlı olarak YSA'nın çözüm uzayında 39 milyonun üzerinde kayıt oluşmaktadır. Fakat bu kayıtların hepsi anlamlı kayıt niteliğinde olmayıp yalnız 14.827.008 tanesi anlamlı kayıt niteliğindedir. Örneğin kullanıcının hem fiziksel engelli hem de atletik yapıda olması imkânsız olduğu için bu ve bunun gibi anlamsız kayıtlar çözüm uzayından ayıklanmış ve sonuçta kayıt sayısı

14.827.008'a düşürülmüştür. YSA'nın eğitilmesi için bu çözüm uzayından 30.000 kayıtlık bir eğitim kümesi seçilmiştir. YSA'nın eğitiminde üç farklı öğrenme algoritması denenmiştir. Bu algoritmaların başarısını test etmek için eğitim setinin dışında 3000 kayıtlık bir test kümesi daha seçilmiş ve sonuçlar karşılaştırılmıştır (Çizelge 3.7).

Geliştirilen navigasyon modülünde öğrenme algoritması olarak diğerlerine oranla daha başarılı olduğu için Levenberg-Marquard algoritması kullanılmıştır. Geliştirilen uygulamada YSA'nın eğitiminde Atila'nın doktora tezinde kullandığı eğitim seti kullanılmıştır [63]. Ayrıca YSA'nın sonuçları tahmin etmesi için gerekli olan kodlamalar Java programlama dilinde gerçekleştirilmiştir. Bu işlemler için Encog kütüphanesinden faydalanılmıştır.

Çizelge 3.7. Öğrenme algoritmalarının başarı oranları.

Öğrenme Algoritması	**Ortalama Karesel Hata**	**Başarı Oranı (%)**
Levenberg-Marquard [76]	2,5733e-04	99,9
Resilient Propagation [77]	0,01356	97,9
BFGS-Quasi Newton [78]	0,00623	98,8

YSA destekli sunucu navigasyon modülünün ürettiği güzergâh binadaki değişen şartlara ayak uydurabilecek niteliktedir. Gerektiğinde sistem kullanıcıyı farklı bir güzergâha yönlendirebilmektedir. Ya da kullanıcı bazen sistemin ürettiği güzergâhın dışına çıkabilmektedir. Bu gibi sebeplerden dolayı sistemde mobil uygulama, sunucudan her iki saniye de bir yönlendirme talebinde bulunmaktadır. Bu yönlendirme talebinde kullanıcının konumu kontrol edilerek, navigasyon modülü tarafından mobil uygulamaya yönlendirme verileri gönderilmektedir Yönlendirme verisi kullanıcının o anki konumuna ait ortam fotoğrafını ve yönlendirme talimatını içermektedir. Sunucu üzerinde kullanıcının mevcut konumu ve güzergâhı üzerindeki noktalar kullanılarak kullanıcının hareket etmesi gereken yön hesaplanır ve bu yöne göre yönlendirme talimatı oluşturulur.

3.2.2. Navigasyon Esnasında Kullanıcıların Yönlendirilmesi

Bir navigasyon sistemindeki önemli problemlerden biri de kullanıcının doğrultusunun ve navigasyon esnasında hareket etmesi gereken yönün belirlenmesidir. Geliştirdiğimiz akıllı tahliye sisteminde kullanıcı bina içinde ilerlerken, geride bıraktığı düğümlere bağlı olarak doğrultusu ve bulunduğu düğüme bağlı olarak konumu belirlenmektedir. Yönlendirme işlemi ise kişinin gitmesi gereken bir sonraki ve iki sonraki düğümlerin koordinatlarına bağlı olarak gerçekleştirilmektedir. Ayrıntıları aşağıda açıklandığı üzere, sistem kullanıcıyı "sağa dön, sola dön, düz ilerle, yukarı çık, aşağı in, katta ilerle, yukarı çık katta ilerle, aşağı in katta ilerle" gibi talimatlarla yönlendirmektedir. Çalışmada bu talimatların belirlenmesinde jeodezide kullanılan yan nokta hesabındaki dik boyu hesaplamalarından faydalanılmıştır.

3.2.2.1. Semt (Açıklık) Açısı

Harita uygulamalarında kullanılan koordinat sistemi matematikte kullanılan koordinat sisteminden farklıdır. Jeodezide +x ekseni her zaman kuzeyi göstermektedir. Buna paralel olarak -x güney, +y ekseni doğu ve -y ekseni batı yönlerini göstermektedir. Ayrıca jeodezide kullanılan koordinat sistemindeki bölgeler de matematikteki koordinat sistemindeki bölgelerden farklılık göstermektedir. Şekil 3.10'da jeodezik koordinat sisteminin eksenleri ve bölgeleri gösterilmiştir.

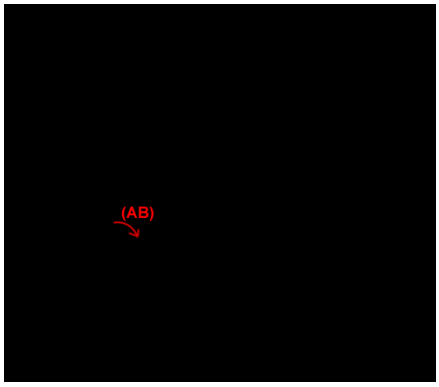
Şekil 3.10. Jeodezik koordinat sistemi.

A noktasından B noktasına giden bir doğrunun +x ekseni ile saat yönünde yapmış olduğu açıya semt (açıklık) açısı denir ve (AB) ile gösterilir. Semt açısı 0^g (0°) ile 400^g (360°) arasında değerler alabilir. Şekil 3.11'de farklı doğrultulardaki AB doğrularına ait semt açıları gösterilmiştir.

Şekil 3.11. Dört farklı bölgeye ait semt açıları.

Şekil 3.12'deki koordinatları bilinen A ve B noktalarının oluşturduğu AB doğrusunun semt açısı şu şekilde hesaplanır [79]:

— — ——— (4.1)



Şekil 3.12. Semt açısının hesaplanması.

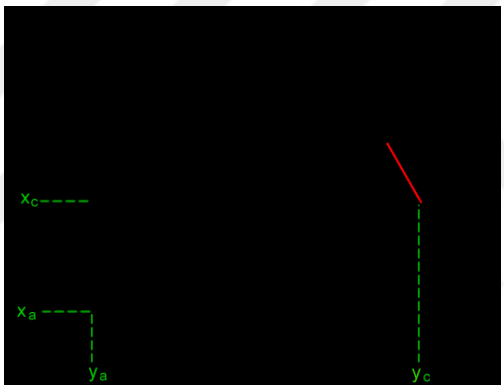
Semt açısının hesabı jeodezik koordinat sisteminin her bölgesi için farklılık gösterir. Çizelge 3.8'de bölgeye göre eklenmesi gereken açı değeri gösterilmiştir. AB doğrusunun semt açısı hesaplanırken öncelikle AB doğrusunun bulunduğu bölge tespit edilir. Daha sonra Formül 4.1'deki hesaplama yapılır ve elde edilen açı değerine Çizelge 3.8'deki ilgili açı değeri eklenerek semt açısı bulunmuş olur.

Çizelge 3.8. Bölgelere göre semt açısına eklenen açı değerleri.

Bölge			Eklenen Açı Değeri
I	+	+	0^g
II	+	-	200^g
III	-	-	200^g
IV	-	+	400^g

3.2.2.2. Dik Boyu Hesabı

A ve B noktaları aynı doğru üzerinde, C noktası B noktasından inilen bir dikmenin üzerinde ise ve bu üç noktanın koordinatları biliniyorsa, dik boyu hesabı ile BC dikmesinin boyu hesaplanabilir.



Şekil 3.13. Dik boyunun hesaplanması.

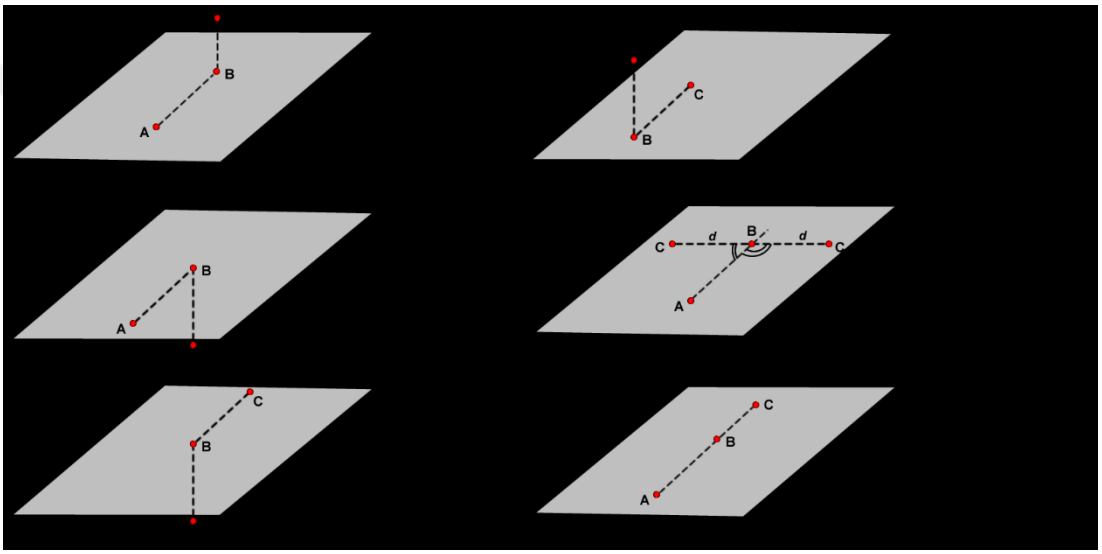
Şekil 3.13'deki 'nin hesaplanması için öncelikle Formül 4.1 kullanılarak (AB) semt açısının hesaplanması gerekmektedir. Daha sonra üçgenlerde benzerlik teoremi kullanılarak elde edilen Formül 4.2 ile dik boyu hesaplanabilir.

(4.2)

3.2.2.3. Yönlendirme Talimatlarının Oluşturulması

Kullanıcı herhangi bir hatta ilerlerken, bir düğüm noktasına gelmeden önce, o düğüm noktasında hangi yöne gideceği belirlenmektedir. Bunun için önce kullanıcının

bulunduğu düğümden sonraki ilk düğüm noktası ile ikinci düğüm noktasının yükseklik değerleri karşılaştırılmaktadır [80]. İkinci düğüm birinci düğümden daha yüksek bir noktada ise yukarı çıkılması gerektiği anlaşılmaktadır (Şekil 3.14a). Tersi durumda yani ikinci düğüm birinci düğümden daha düşük bir noktada ise bu kez de aşağı inilmesi gerekmektedir (Şekil 3.14b). Bir sonraki ve iki sonraki düğümlerin yükseklikleri eşitse, fakat mevcut düğümün kotu onlardan daha düşük ise "yukarı çık ve katta ilerle" komutu (Şekil 3.14c), mevcut düğümün kotu daha yüksekte ise "aşağı in ve katta ilerle" komutu (Şekil 3.14d) geçerli olmaktadır.



Şekil 3.14. Yönlendirme talimatlarının belirlenmesi [80].

Eğer her üç düğümün de yüksekliği eşitse, kullanıcı "sağa dön", "sola dön" ya da "düz ilerle" komutları ile aynı katta yönlendirilmektedir. Yön tayininde dik boyu hesabından faydalanılmıştır. Kullanıcının üzerinde bulunduğu mevcut düğüm A noktası ve izleyeceği düğümler sırası ile B ve C noktaları olarak kabul edilsin. Bu durumda C düğümünün AB doğrusuna olan dik boyu Formül 4.2'ye göre hesaplanır. Eğer hesaplanan dik boyu; sıfırdan büyük ise C noktası AB doğrusunun sağında, sıfırdan küçük ise solunda anlamı taşımaktadır (Şekil 3.14e). Eğer hesaplanan değer sıfıra eşitse bu üç düğümün aynı hat üzerinde olduğu anlaşılmaktadır (Şekil 3.14f).

Geliştirilen sistemde, yukarıda açıklandığı gibi bulunan yön ve bu yöne bağlı olarak oluşturulan metin formatındaki talimatlar JSON veri formatına dönüştürülerek

kullanıcının akıllı telefonuna gönderilmektedir. Telefondaki mobil uygulama JSON formatındaki verileri seslendirerek kullanıcıyı yönlendirmektedir.

3.2.3. Veri Tabanı

Geliştirilen akıllı tahliye sitemine ait veriler, sunucu üzerindeki Oracle Spatial 11g veri tabanı yönetim sisteminde saklanmaktadır. Oracle Spatial'ın Java programlama diline sunduğu ağ analizleri kütüphanesini kullanabilmek için binanın bir ağ (network) modeli oluşturulmuştur. Oracle Spatial'da ağ modeli oluşturmak için, binadaki düğümlere ve bağlantılara ait konumsal verilerin tutulduğu ayrı birer tablo tanımlanması gerekmektedir. Bu kapsamda, binadaki düğümlere ait verileri saklamak için ENG_NODE, yine binadaki bağlantılara ait verileri saklamak için ise ENG_LINK isimli tablolar oluşturulmuştur. Bu iki tablodan yararlanarak Oracle Spatial'da ENG_MODEL isminde bir ağ modeli oluşturulmuştur. Konumsal analizler ve en kısa yolun bulunmasında bu ağ modeli kullanılmıştır.

ENG_NODE tablosu binadaki düğümlere ait konumsal verileri saklar. Sistemdeki ağ modelini oluşturulurken düğüm bilgilerine erişim için bu tablo kullanılır. Veri tabanı yönetim sisteminde bu tablo;

```
CREATE TABLE ENG_NODE(
Node_ID number,
SDO_LOCATION SDO_GEOMETRY,
NODE_NAME varchar(30),
CONSTRAINT pk_node PRIMARY KEY (Node_ID));
```

SQL komutu ile oluşturulmuş olup, NODE_ID düğüm numarasını gösteren birincil anahtar alanıdır. SDO_LOCATION düğüme ait koordinatları gösteren diğer bir alan olup, SDO_GEOMETRY tipinde veriler içerir. NODE_NAME ise düğümü tanımlayıcı bilgiler içeren metin tipinde bir alandır.

ENG_LINK tablosu binadaki bağlantılara ait konumsal verilerin saklandığı tablodur. Sistemdeki ağ modelini oluştururken bağlantı bilgilerine erişim bu tablodan sağlanır. Binadaki her bir bağlantı çift yönlü olarak tanımlanmıştır. Veri tabanı yönetim sisteminde bu tablo;

```
CREATE TABLE LINK(
LINK_ID number,
LINK_NAME varchar(30),
START_NODE_ID number NOT NULL,
END_NODE_ID number NOT NULL,
LINK_LENGTH number NOT NULL,
SDO_GEOM SDO_GEOMETRY,
BIDIRECTED char(1),
LINK_TYPE varchar(30),
CONSTRAINT pk_link PRIMARY KEY (Link_ID));
```

SQL komutu ile oluşturulmuş olup, LINK_ID alanı bağlantı numarasını gösteren birincil anahtar alanı, LINK_NAME bağlantıyı tanımlayan bir metni, START_NODE bağlantının başlangıç düğümünü, END_NODE bağlantının bitiş düğümünü, LINK_LENGTH bağlantının uzunluğunu, SDO_GEOM bağlantının konumsal bilgilerini ve LINK_TYPE ise bağlantı türünü temsil eden alanlardır. LINK_TYPE ya merdiven ya da düz koridor şeklinde iki farklı veri tipini içerebilir.

Veri tabanında ENG_NODE ve ENG_LINK tablolarını kullanılarak ağ analizlerinde kullanılan ENG_MODEL adındaki ağ modeli aşağıdaki SQL komutu ile oluşturulmuştur:

```
INSERTINTO USER_SDO_NETWORK_METADATA (
NETWORK,
NETWORK_CATEGORY,
GEOMETRY_TYPE,
NO_OF_HIERARCHY_LEVELS,
NO_OF_PARTITIONS,
LINK_DIRECTION,
NODE_TABLE_NAME,
NODE_GEOM_COLUMN,
LINK_TABLE_NAME,
LINK_GEOM_COLUMN,
LINK_COST_COLUMN,
)
VALUES(
'ENG_MODEL',-- network (primary key)
'SPATIAL',-- network_category
'SDO_GEOMETRY',-- geometry_type
1,-- no_of_hierarchy_levels
1,-- no_of_partitions
'UNDIRECTED',-- link_direction
'ENG_NODE',-- node_table_name
'SDO_LOCATION',-- node_geom_column
'ENG_LINK',-- link_table_name
'SDO_GEOM',-- link_geom_column
'LINK_LENGTH',-- link_cost_column
);
```

Oluşturulan bu ağ modeli sunucuda en kısa yol analizleri için kullanılmaktadır. Sunucudaki navigasyon modülündeki servlet sınıfları en kısa yol bulmak için Oracle Spatial'in Java platformuna yönelik sunduğu *sdonm.jar* ve *sdoapi.jar* kütüphanelerini kullanmaktadırlar. Bu kütüphaneler ağ modeli üzerinde birçok analizi yapabilecek metotlara sahiptir. Bu kapsamda sunucudaki yönlendirme için kullanılan servlette tanımlanan ENG_MODEL isimli ağ modelini kullanan bir ağ nesnesi tanımlanmıştır.

```
network = NetworkManager.readNetwork(connection,"ENG_MODEL");
```

network, ENG_MODEL'e ait bütün bağlantı ve düğümleri içeren bir ağ nesnesidir. Sisteme bağlı her bir kullanıcıya farklı bir yönlendirme hizmeti sunulacağı için sunucu üzerinde her bir kullanıcıya yönelik bu modelin ayrı bir kopyası oluşturulur.

```
network.clone();
```

Böylece sunucudan yönlendirme talebinde bulunan her bir kullanıcı için yeni bir ağ modeli oluşturulmuş olur. Bu ağ modeli kullanıcı ilk bağlantı yaptığında oluşturulur. Kullanıcının daha sonraki her yönlendirme talebinde bu mevcut model üzerinde analizler yapılır. Bu ağ modeli sunucu üzerinde ana bellekte tutulduğu için sistem kullanıcının taleplerine cevap vermekte herhangi bir gecikme yaşamaz.

Binadaki değişen şartlar ve kullanıcının kişisel özelliklerini dikkate alarak, bağlantıların kullanılabilirlik durumlarını inceleyen YSA, kullanıcı için tehlike arz eden linkleri belirler ve bu linkleri, aşağıdaki kod parçasında gösterildiği gibi, kullanıma kapalı olan bağlantılar olarak modele bildirir.

```
constraint=new SystemConstraint(network_A);
constraint.setMustAvoidLinks(avoidLinkList);
```

Sonuç olarak binanın fiziksel koşullarına göre ve kullanıcının kişisel özelliklerine göre ağ üzerindeki bazı bağlantılar ve düğümler bu kullanıcının kullanımına kapatılmış olur. Yani sistem kullanıcıya bu kapatılan bağlantılar ve düğümler haricinde bir çıkış güzergâhı sunmak durumundadır.

ENG_TAG tablosu, binadaki etiketlerin hangi düğüme bağlı olduğunu gösteren tablodur. Bu tablo;

```
CREATE TABLE ENG_TAG(
ID number,
NODEID number REFERENCES ENG_NODE(NODE_ID),
CONSTRAINT pk_tag PRIMARY KEY (id));
```

SQL komutu ile oluşturulmuş olup, ID alanı her bir etiketin numarasını gösteren birincil anahtar alanı, NODEID ise bu etiketin hangi düğüme ait olduğunu gösteren alanlardır. Ayrıca NODEID, ENG_NODE tablosundaki NODE_ID alanını referans eden ikincil anahtar özelliğindedir.

ENG_PHOTO tablosu binadaki bağlantılara ait fotoğrafların sunucudaki adreslerinin saklandığı tablodur. Bu tabloda her bir bağlantı için ikişer fotoğraf tutulmaktadır. Çünkü A düğümünden B düğümüne giderken kullanılan fotoğraf ile B düğümünden A düğümüne giderken kullanılan fotoğraf farklı olmalıdır. Yani bir bağlantının geliş ve gidiş yönlerinde çekilmiş iki fotoğrafı olmalıdır. Bu sebeple bu tabloya LINKID alanı dışında bir de yön belirtmek için STARTNODE ve ENDNODE alanları eklenmiştir. Bu tablo;

```
CREATE TABLE ENG_PHOTO(
ID number,
LINKID number REFERENCES ENG_LINK(LINK_ID),
STARTNODE number,
ENDNODE number,
PHOTO_NAME varchar(20),
CONSTRAINT pk_photo PRIMARY KEY (id));
```

SQL komutu ile oluşturulmuş olup, ID alanı fotoğrafın numarasını tutan birincil anahtar alanıdır. LINKID alanı ise fotoğrafın binadaki hangi link üzerinde olduğunu göstermektedir ve ENG_LINK tablosu ile bağlantıyı sağlamaktadır. STARTNODE ve ENDNODE alanları ise fotoğrafın ilgili bağlantıya ait yönünü gösteren alanlardır. PHOTO_NAME alanı ise fotoğrafın adresini tutmaktadır.

Tahliye süresince kullanıcının bulunduğu ortamın, mobil uygulamada görselleştirilebilmesi için binadaki her bir bağlantının çift yönlü olarak fotoğrafları çekilmiştir. Çekilen her bir fotoğraf sunucuya yüklenmiş ve yönü de göz önünde bulundurularak veritabanındaki ilgili bağlantı ile ilişkilendirilmiştir. Böylece

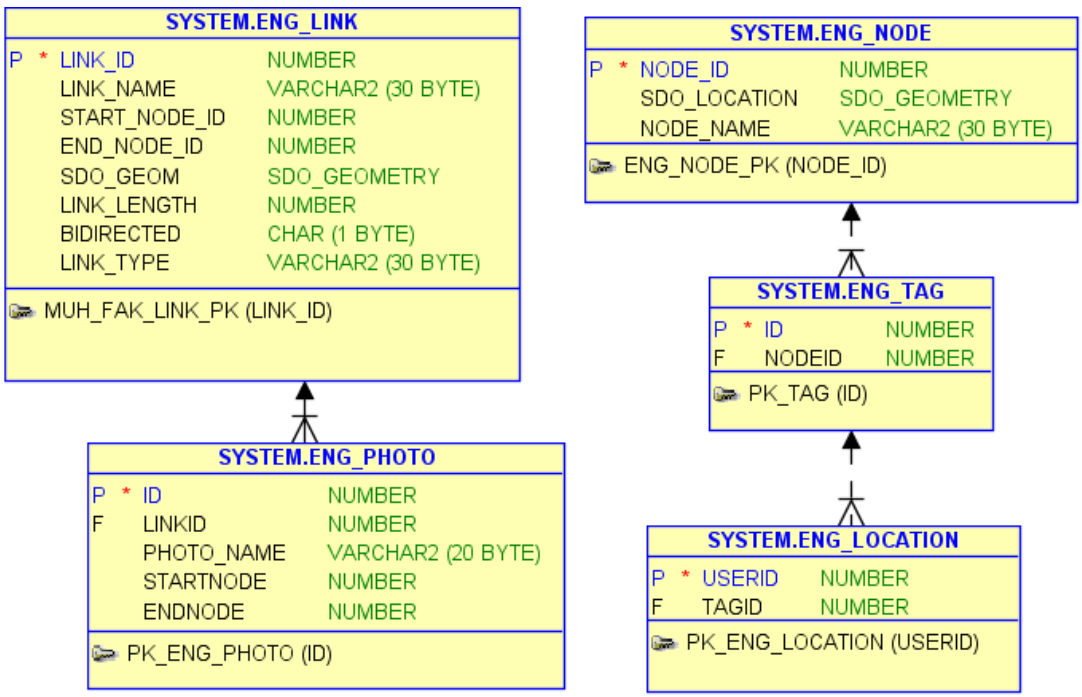
kullanıcının tahliyesi boyunca yönlendirme işleminde kullanılacak fotoğraflar hazır hale getirilmiştir.

Veri tabanında ENG_LOCATION adındaki diğer bir tablo ise anlık olarak kullanıcıların bulunduğu etiketlerin kaydını tutar. Yani RFID okuyucunun göndermiş olduğu, kullanıcının konumunu temsil eden etiket bilgilerini tutar. ENG_LOCATION tablosundaki etiket değeri ENG_TAG tablosu ile eşleştirilerek kullanıcının hangi düğümde olduğu tespit edilir. Bu tablo;

```
CREATE TABLE ENG_LOCATION(
USERID number,
TAGID number REFERENCES ENG_TAG(ID),
CONSTRAINT pk_RFID PRIMARY KEY (userID));
```

SQL komutu ile oluşturulmuş olup, USERID her bir kullanıcıya ait kimlik tanımlayıcısı, TAGID ise kullanıcının o an üzerinde bulunduğu etiketin numarasıdır.

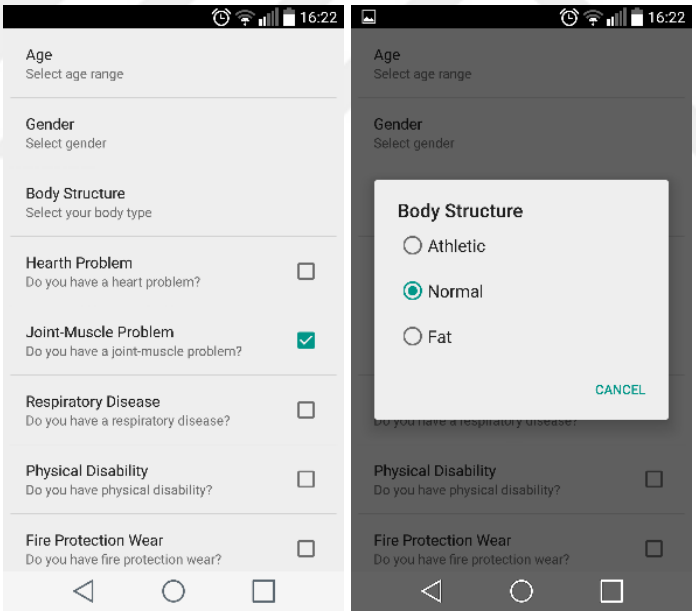
Oluşturulan veri tabanına ait ER diyagramı Şekil 3.15'degösterilmiştir. Binada yapılan ölçümler ve incelemeler sonucu elde edilen veriler veri tabanındaki ilgili tablolara kayıt edilmiştir.



Şekil 3.15. Veri tabanına ait ER Diyagramı.

3.3. MOBİL NAVİGASYON UYGULAMASI

Akıllı tahliye sisteminin istemci tarafını oluşturan mobil uygulama, kullanıcı ile tahliye sistemi arasındaki etkileşimi sağlamaktadır. Geliştirilen mobil uygulama Android tablet ve akıllı telefonlara yöneliktir. Uygulama mobil cihaza ilk yüklendiğinde kullanıcı ayarlar kısmına girmekte ve Çizelge 3.5'de belirtilen kendisine ait on kişisel özelliği tanımlamaktadır (Şekil 3.16). Kullanıcı ilk sunucuya bağlandığında kullanıcıya ait bu özellikler sunucuda saklanmaktadır. Böylece sunucuya gönderilen her yönlendirme talebinde bu kişisel özelliklerin tekrar tekrar sunucuya gönderilmesine gerek kalmamaktadır. Yani kullanıcı bu mobil uygulamayı cihazına yüklediğinde bir seferlik bu özellikleri sisteme tanımlaması yeterli olmaktadır. Uygulama kişiye has bu bilgileri Shared Preference nesnesinde saklamaktadır.

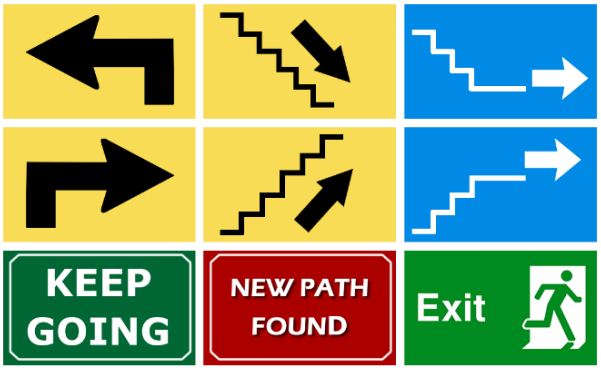


Şekil 3.16. Mobil navigasyon uygulamasının ayarlar ekranı.

Kullanıcı binadaki yangından haberdar olduğunda mobil cihazındaki uygulamayı kendisi başlatmaktadır. Uygulama ilk başlatıldığında sunucuda kullanıcının kişisel özellikleri ve binanın fiziksel koşulları göz önünde bulundurularak güvenli bir çıkış güzergâhı üretilmektedir. Bu güzergâh üzerindeki bağlantılar her bir kullanıcı için sunucuda saklanmaktadır. Uygulama iki saniye aralıklar ile sunucuya istek gönderip,

yönlendirme talebinde bulunmaktadır. Bu işlem için mobil uygulamada iki saniye aralıklar ile çalışan bir zamanlayıcı tanımlanmıştır. Mobil uygulamadan sunucuya ilk bağlantı yapıldığında, navigasyon modülü kullanıcı için bir çıkış güzergâhı oluşturmakta ve bu güzergâh üzerindeki bağlantılar bir listede saklamaktadır. Mobil uygulamadan sunucuya yapılan her bir yönlendirme talebinde, navigasyon modülü kullanıcının konumunu tespit ederek, kullanıcının mevcut çıkış güzergahı üzerinde olup olmadığını kontrol etmektedir. Kullanıcı mevcut güzergâh üzerinde yürümeye devam ettiği sürece, mobil uygulama kullanıcıyı hesaplanan güzergahta hem görsel ve hem de işitsel öğeler ile yönlendirilmektedir. Eğer kullanıcı güzergâh dışına çıkarsa, sunucuda kullanıcının konumuna bağlı olarak yeni bir çıkış güzergâhı üretilmekte, bu güzergâh bilgileri kullanıcının mobil cihazına gönderilmekte ve kullanıcı bu yeni güzergâha göre yönlendirilmektedir.

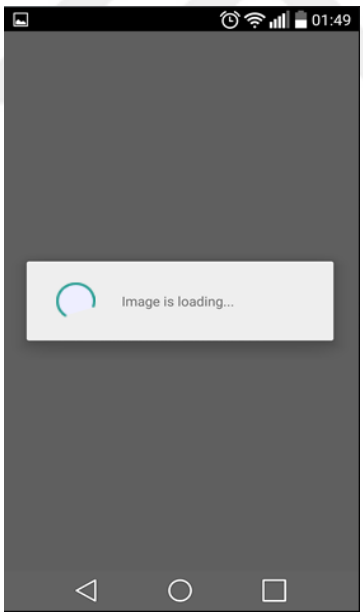
Yönlendirmede kullanılan görsel öğeler, üzerinde bulunan bağlantıya ait ortam fotoğrafları ve takip edilmesi gereken yönleri gösteren yönlendirme işaretlerini içermektedir (Şekil 3.17). Linklere ait ortam fotoğrafları veri tabanındaki ENG_PHOTO tablosundan çekilmektedir. Ayrıca kullanıcının güzergâhı üzerindeki bir sonraki linke ait sıcaklık değeri ve kullanıcının çıkışa olan uzaklığı metre cinsinden ekranda görüntülenmektedir. Sunucuda üretilen ortam fotoğrafları ve yönlendirme talimatlarının metni JSON formatına dönüştürülüp mobil cihaza gönderilmektedir. Mobil cihaz ise JSON formatında gelen verileri geri dönüştürerek ekrana getirmektedir.



Şekil 3.17. Mobil cihazdaki yönlendirme işaretleri.

Yine yönlendirmede kullanılan işitsel öğeler ise kullanıcının izlemesi gereken yönü belirten metinlerin seslendirilmiş halleridir. Android'de metinlerin seslendirilmesi için Text to Speech kütüphanesinden faydalanılmıştır. Uygulama İngilizce olarak geliştirildiği için yönlendirme talimatları "GO UPSTAIRS, GO DOWNSTAIRS, GO DOWN AND GO ON THE FLOOR, GO UP AND GO ON THE FLOOR, TURN LEFT, TURN RIGHT, KEEP GOING" olarak belirlenmiştir. Kullanıcı tahliye işlemi boyunca akıllı telefonundan bu sesli talimatlar ile yönlendirilmektedir. Kullanıcı çıkışa ulaştığında, akıllı telefondan "YOU ARE AT THE EXIT" sesli mesajını alarak sistem tarafından başarı ile binanın çıkışına ulaştırıldığı bildirimini almaktadır.

Telefondaki mobil uygulama, fotoğrafları sunucudan indirirken AsyncTask sınıfını kullanılmaktadır. AsyncTask sınıfı, daha öncede bahsedildiği gibi iş parçacıklarını kullanarak, arka planda ilgili fotoğraf indirilirken, ana ekranda kullanıcıya fotoğrafın indirildiğini gösteren bir bilgi ekranı sunmaktadır (Şekil 3.18).



Şekil 3.18. Resim yükleme mesajı.

3.4. YANGIN SİMULASYONU

Geliştirilen akıllı tahliye sisteminin testi için binada gerçek bir yangın çıkarılamayacağından dolayı, sunucu üzerinde çalışan bir yangın simülasyonu

geliştirilmiştir. Bu simülasyonda hazırlanan senaryo doğrultusunda, binadaki bir noktada yangın başlatmakta ve yangını bina içerisine yayılmaktadır. Simulasyonda yangın başladığında binadaki sensörlerden geldiği varsayılan veriler, tanımlanan senaryoya sadık kalınarak simülasyonda üretilmektedir. Geliştirilen simülasyonda yer alan bina içi olaylar şunlardır:

- Senaryoda yangının başlatıldığı bağlantı başta olmak üzere yangının etkisinin görüldüğü bağlantılarda sıcaklık değeri ilk 40 dakika boyunca artmakta, sonra yangının sonlanmaya başladığı düşünülüp, sıcaklık değerleri yavaş yavaş düşürülmektedir.
- Yine senaryoda belirtilen bağlantılarda duman yayılımı başlamakta, buna bağlı olarak karbon monoksit oranı artmakta, görüş mesafesi azalmaktadır.
- Yangın başladıktan sonra yangının etkilerinin görüldüğü bağlantılarda insan yoğunluğu 20. saniyeden sonra azaltılmaktadır.
- Yangının birinci ile beşinci dakikaları arasında merdivenler üzerinde insan yoğunluğu artırılmakta, beşinci dakikadan sonra yavaş yavaş azaltılmaktadır.
- Yangının yedinci ile onuncu dakikası arasında çıkış noktalarındaki yoğunluk artırılmakta, onuncu dakikadan sonra yavaş yavaş azaltılmaktadır.

Simülasyon üzerindeki süreler test binanın yapısı ve büyüklüğü göz ününde bulundurularak belirlenmiştir. Bu yangın simülasyonu, sunucu üzerinde çalışan bir servis olarak Java dilinde geliştirilmiştir. Sunucu çalıştırıldığı sürece yangın simülasyonu arka planda çalışmaktadır.

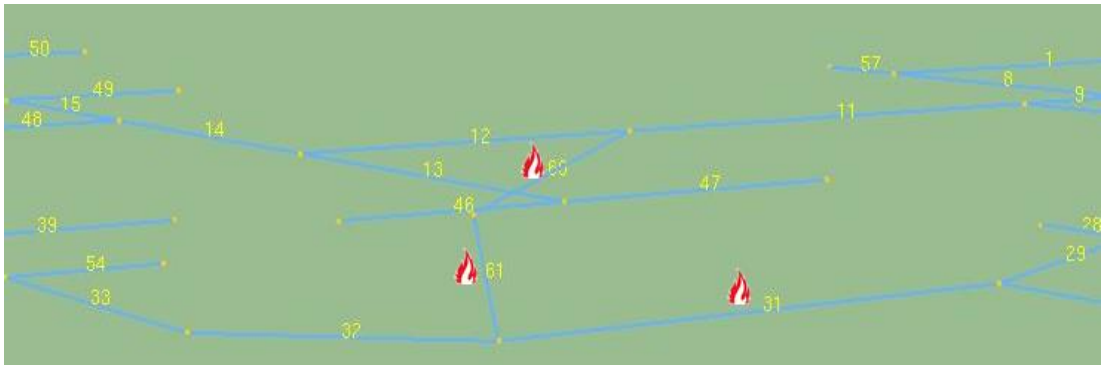
Ayrıca geliştirilen akıllı tahliye sisteminin server navigasyon modülü Java dilinde servlet kullanılarak kodlanmış ve Glass Fish Web Sunucusu üzerinde çalıştırılmıştır. Mobil navigasyon modülü ise Android Studio ortamında kodlanmıştır.

BÖLÜM 4

DENEYSEL ÇALIŞMALAR VE DEĞERLENDİRME

Geliştirilen akıllı tahliye sisteminin testleri için Karabük Üniversitesi Mühendislik Fakültesi binası seçilmiştir. Binadaki 63 nokta düğüm olarak tanımlanmıştır. Binadaki düğüm koordinatlarını tespit etmek için 120 adet pasif RFID etiketi kullanılmıştır. Yine binada bu düğümleri birbirine bağlayan 65 tane bağlantı tanımlanmış ve bu bağlantılara ait ortam fotoğrafları çekilmiştir. Konum belirleme için ATID 870 Hand-Held RFID okuyucusu kullanırken, mobil navigasyon modülünü çalıştırmak için LG G3 akıllı telefonu kullanılmıştır.

Test işlemi için binada bir yangın senaryosu üretilmiştir. Senaryo, hangi bağlantılarda ne zaman yangının başlayacağı ve hangi bağlantılara duman yayılımın olacağı bilgisini içermektedir. Sunucu üzerinde çalışan yangın simülasyonu bu senaryodaki adımlara göre çalışmaktadır. Hazırlanan senaryoda 10. saniyede 31 numaralı bağlantıda yangın başlamakta, yangın 46. saniyede 61 numaralı bağlantıya ve 62. saniyede ise 60 numaralı bağlantıya sıçramaktadır (Şekil 4.1). 60 ve 61 numaralı bağlantılar zemin katı birinci kata bağlayan merdivenlerdir.



Şekil 4.1. 37. saniyede simülasyonda binanın durumu.

Çizelge 4.1. Senaryo adımları ve olaylar.

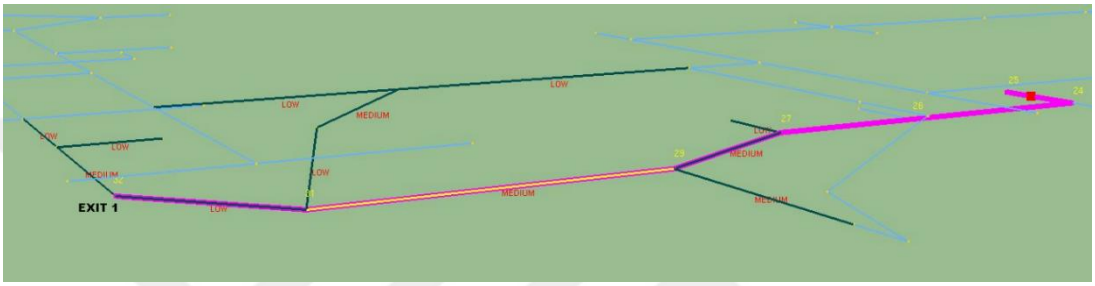
Adım No	Başlangıç Zamanı	Olay Tipi	Bağlantı No
1	00:00:10	Yangın	31
2	00:00:14	Duman Yayılımı	61
3	00:00:18	Duman Yayılımı	32
4	00:00:22	Duman Yayılımı	60
5	00:00:26	Duman Yayılımı	29
6	00:00:28	Duman Yayılımı	11
7	00:00:30	Duman Yayılımı	30
8	00:00:32	Duman Yayılımı	12
9	00:00:34	Duman Yayılımı	33
10	00:00:36	Duman Yayılımı	34
11	00:00:38	Duman Yayılımı	28
12	00:00:40	Duman Yayılımı	54
13	00:00:42	Duman Yayılımı	27
14	00:00:46	Yangın	61
15	00:00:48	Duman Yayılımı	65
16	00:00:50	Duman Yayılımı	26
17	00:00:52	Duman Yayılımı	38
18	00:00:54	Duman Yayılımı	39
19	00:00:56	Duman Yayılımı	35
20	00:00:58	Duman Yayılımı	25
21	00:01:00	Duman Yayılımı	24
22	00:01:02	Yangın	60
23	00:01:06	Duman Yayılımı	14
24	00:01:08	Duman Yayılımı	13
25	00:01:12	Duman Yayılımı	9
26	00:01:16	Duman Yayılımı	10
27	00:01:18	Duman Yayılımı	37
28	00:01:22	Duman Yayılımı	36

Senaryo toplam 28 adım içermektedir. 1., 14. ve 22. adımlarda ilgili bağlantılarda yangın, diğer 25 adımda ise duman yayılımı gösterilmektedir. Çizelge 4.1'de yangın senaryosundaki olayların zamanı, hangi bağlantıda oluştuğu ve olay tipi sırasıyla gösterilmektedir. Geliştirilen sistem sadece bu senaryoya bağlı değil, tanımlanacak her türlü senaryoda çalıştırılabilecek niteliktedir. Uygulama üç farklı kullanıcı tipi üzerinde test edilmiştir. Bu kullanıcılara ait özellikler Çizelge 4.2'de verilmiştir:

Çizelge 4.2. Test kullanıcılarının özellikleri.

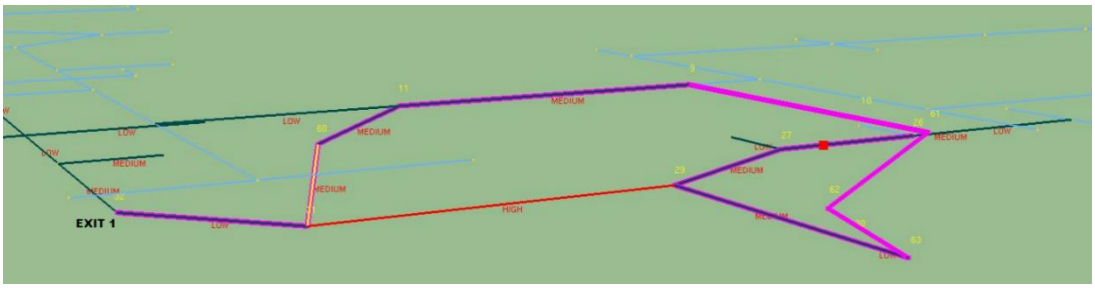
Özellikler	Kullanıcı-1	Kullanıcı-2	Kullanıcı-3
Yaş	61	38	28
Cinsiyet	Kadın	Erkek	Erkek
Vücut Tipi	Şişman	Normal	Atletik
Fiziksel Engel	Yok	Yok	Yok
Binaya Aşinalık	Yok	Var	Yok
Eklem, Kas Rahatsızlıkları	Var	Yok	Yok
Kalp Rahatsızlığı	Var	Yok	Yok
Solunum Yolları Rahatsızlığı	Var	Yok	Yok
Gaz Maskesi	Yok	Yok	Var
Yangın Giysisi	Yok	Yok	Var

Kullanıcı-1, yangının 30. saniyesinde 25 numaralı düğümden harekete başladığında, sistem kullanıcı için 56,2 m uzunluğunda Şekil 4.2'deki güzergâhını üretmiş ve kullanıcıyı binanın 1 numaralı çıkışına yönlendirmiştir. Kullanıcı 1'in çıkabileceği bütün güzergâhlar "ORTA" risk seviyesine sahip bağlantılar içerdiği için, bu güzergâhlardan en kısa mesafeli olanı kullanıcıya sunulmuştur. Kullanıcıya sunulan güzergâh üzerinde 31 numaralı (düğüm 29 ile düğüm 31 arasındaki) bağlantı "ORTA" risk seviyesindedir.



Şekil 4.2. Kullanıcı-1 için hesaplanan ilk güzergâh.

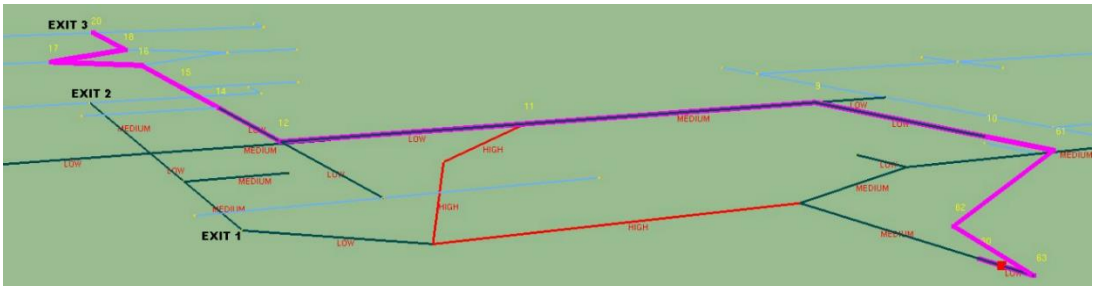
Kullanıcı-1'in hareketinden 17 saniye sonra 31 numaralı bağlantının risk seviyesi "YÜKSEK" seviyesine çıkmış ve sistem kullanıcıya daha uzun ama kendisi açısından daha güvenli ve risk seviyesi "ORTA" olan yeni bir güzergâh sunmuştur. Yeni güzergâhın çıkış noktası değişmemiş fakat uzunluğu 84,35 m olmuştur (Şekil 4.3).



Şekil 4.3. Kullanıcı-1 için 17. saniyede hesaplanan yeni güzergâh.

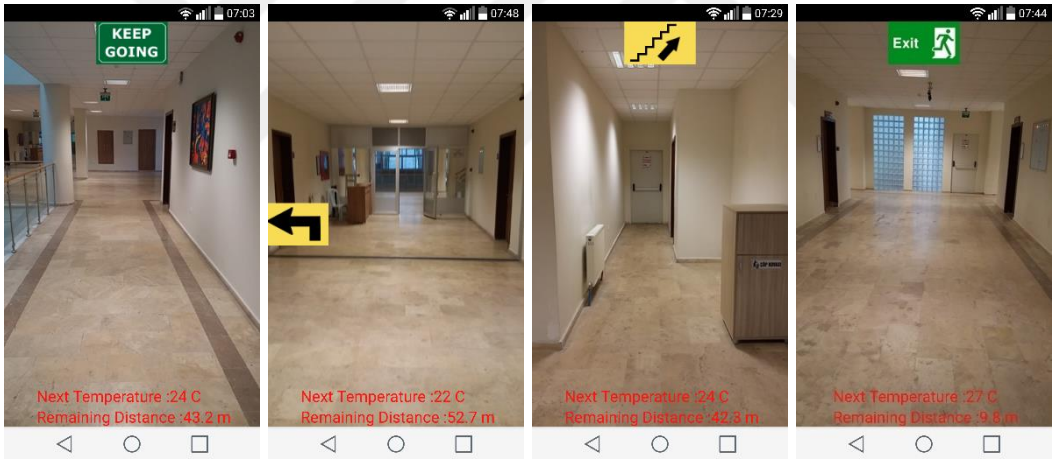
Kullanıcının hareketinin 37. saniyesinde ise mevcut güzergâhta düğüm 11 ile düğüm 31 arasında risk "YÜKSEK" seviyeye çıkmış ve sistemde risk seviyesi "ORTA" olan başka bir çıkış güzergâhı hesaplanmıştır. Yeni bulunan güzergâh kullanıcıyı binanın 3 numaralı çıkışına yönlendirmiştir. Yeni güzergâhın toplam uzunluğu 118,7 m olarak

hesaplanmıştır (Şekil 4.4). Kullanıcı-1 bina dışına, son bulunan güzergâhtan tahliye edilmiş ve bu işlem toplam 150 saniye sürmüştür.



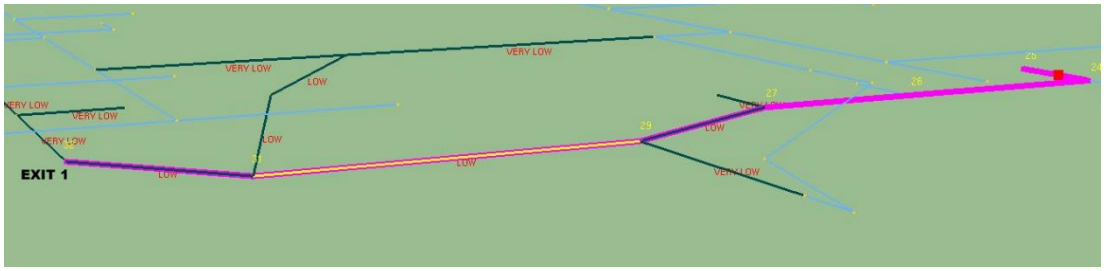
Şekil 4.4. Kullanıcı-1 için 37. saniyede hesaplanan yeni güzergâh.

Tahliye işlemi esnasında Kullanıcı-1'in mobil cihazında ekrana gelen görüntülerden bazıları Şekil 4.5'de gösterilmektedir.



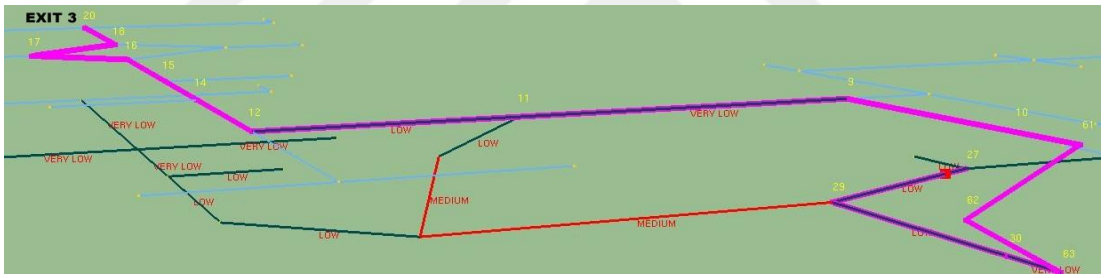
Şekil 4.5. Mobil navigasyon uygulamasının ekran görüntüleri.

Kullanıcı-2, yangının 30. saniyesinde 25 numaralı düğümden harekete başladığında sistem, kullanıcı için 56,2 m uzunluğunda Şekil 4.6'deki güzergâhını üretmiş ve kullanıcıyı binanın 1 numaralı çıkışına yönlendirmiştir. Kullanıcı 1'in çıkış güzergâhındaki bütün bağlantıların riski "DÜŞÜK" seviyesindedir. 29. düğüm ile 31. düğüm arasındaki yangın o an itibari ile Kullanıcı-2 için "DÜŞÜK" seviyede risk teşkil etmektedir.



Şekil 4.6. Kullanıcı-2 için hesaplanan ilk güzergâh.

Kullanıcı-2'in hareketinden 38 saniye sonra 31 numaralı bağlantıda yangının etkileri artmış ve bu kullanıcı için risk "YÜKSEK" seviyeye çıkmıştır. Bu durumda sistem, kullanıcıya daha uzun ama kendisi açısından daha güvenli ve risk seviyesi "DÜŞÜK" olan yeni bir güzergâh sunmuştur. Yeni güzergâhın uzunluğu 98 m ve son noktası 3 numaralı çıkış olmuştur (Şekil 4.7). Kullanıcı-2'nin bina dışına tahliyesi 130 saniye sürmüştür.

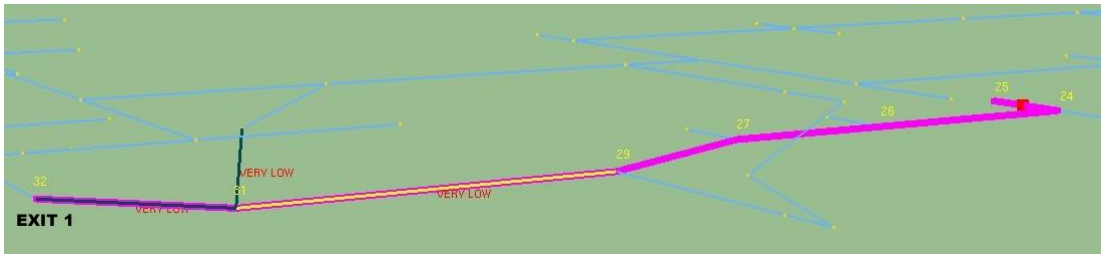


Şekil 4.7. Kullanıcı-2 için 38. saniyede hesaplanan güzergâh.

Kullanıcı-3, yangının 30. saniyesinde 25 numaralı düğümden harekete başladığında, sistem kullanıcı için 56,2 m uzunluğunda Şekil 4.8'deki güzergâhını üretmiş ve kullanıcıyı binadaki 1 numaralı çıkışa yönlendirmiştir. Kullanıcı 1'in çıkış güzergâhındaki bütün bağlantıların riski "ÇOK DÜŞÜK" seviyesindedir. Tahliye sürecinde, 29. düğüm ile 31. düğüm arasındaki yangının etkileri artıp, risk seviyesini Kullanıcı-3 için "DÜŞÜK" seviyesine çıkarsa da güzergâhta bir değişiklik olmadan, kullanıcı başladığı güzergâhtan tahliye edilmiştir. Tahliye işlemi toplam 42 saniye sürmüştür.

Testlerde, kullanıcıların her birinin yangının 30. saniyesinde 25 numaralı düğümde bulunduğu ve bu noktadan mobil navigasyon modülü aracılığı ile sistemden tahliye

talebinde bulunduğu varsayılmıştır. Buna göre her bir kullanıcı için sistemin ürettiği güzergâh bilgileri Çizelge 4.3'de gösterilmiştir.



Şekil 4.8. Kullanıcı-3 için hesaplanan güzergâh.

Çizelge 4.3'de görüldüğü gibi sistem Kullanıcı-1'in tahliyesi için üç kez yeni güzergâh üretmiştir. Kullanıcı-1 150. saniyenin sonunda 118,7 m'lik bir yol kat ederek bina dışına tahliye edilmiştir. Tahliye sistemi, Kullanıcı-2 için iki kez güzergâh üretmiş ve kullanıcıyı 130 saniye süren, toplamda 118,7 m'lik mesafe kat edilen bir güzergâhtan tahliye etmiştir. Kullanıcı-3 ise sistem tarafından kendisine üretilen 56,2 m'lik ilk güzergâhı kullanarak 42 saniyede tahliye edilmiştir. Sistem, başlangıçta bütün kullanıcılara en kısa mesafeli olan 56,2 m'lik çıkış güzergâhını sunmuştur. Fakat zamanla yangının etkilerinin artması ve bu etkilerin her bir kullanıcıya yansımasının farklı derecelerde olmasından dolayı, sistem kullanıcılar için farklı sayılarda güzergâh üretmiş ve farklı güzergâhlar sunmuştur. Kullanıcı-1 ile Kullanıcı-2'nin aynı çıkış noktasından çıktığı görülmektedir.

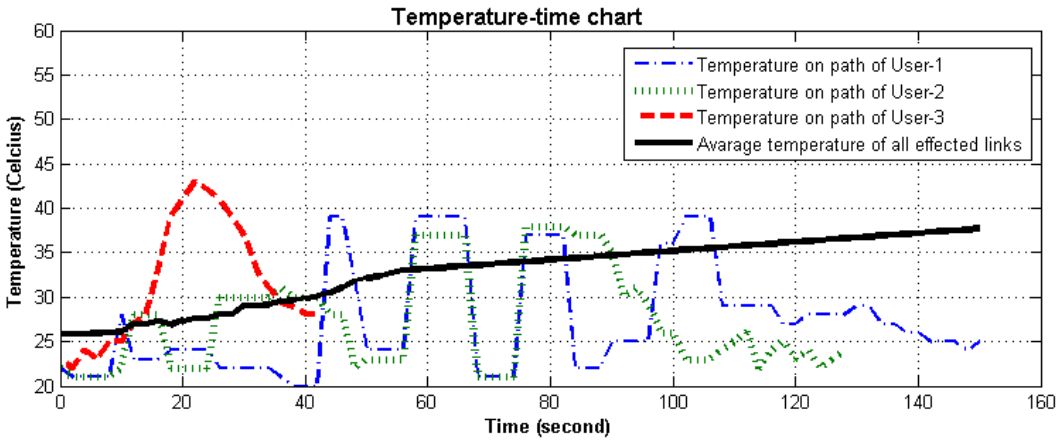
Çizelge 4.3. Testlerde elde edilen çıkış güzergâh bilgileri.

Tahliye Verileri	Kullanıcı-1	Kullanıcı -2	Kullanıcı -3
İlk Hesaplanan Güzergâhın Uzunluğu (m)	56,2	56,2	56,2
Son Hesaplanan Güzergâhın Uzunluğu (m)	118,7	118,7	56,2
Tahliye Süresince Güzergâh Hesaplama Sayısı	3	2	1
Kullanıcının Tahliye Boyunca Maruz Kaldığı Ortalama Sıcaklık(°C)	27,3	27,4	31,4
Tahliye Süresi (sn)	150	130	42

Kullanıcı-1'in tahliye süresinin uzun olması ve güzergâhın üç kez hesaplanmasında bu kullanıcının çeşitli hastalıklara sahip, yaşlı, şişman ve bayan bir kullanıcı olması etken olmuştur. Bu kullanıcı için güzergâh üzerindeki bağlantıların risk seviyesi çok kolay bir şekilde artabilmektedir. Kullanıcı-3 ise yangın giysisine sahip, gaz maskeli, genç

ve atletik bir kullanıcı olduğu için ilk hesaplanan güzergâh üzerinden, güzergâh değiştirmeye gerek duymadan, tahliye edilebilmiştir. Güzergâh üzerindeki bağlantılarda yangın ve etkileri görülmesine rağmen bu kullanıcı için risk seviyesi kolay bir şekilde yükselmemiştir.

Şekil 4.9'daki grafikte, kullanıcıların tahliye esnasında, binadaki bağlantıların sıcaklık değerleri verilmiştir. Grafikte de görüldüğü gibi, tahliye sistemi Kullanıcı-1 ve Kullanıcı-2'yi sıcaklık değeri 40 °C'nin altında olan bağlantılardan tahliye etmiştir. Çünkü sıcaklık değeri 40 °C'nin üstünde olan bağlantılar bu kullanıcılar için tehlike oluşturmaktadır. Kullanıcı-3 ise yangın giysisine sahip olduğu için 40 °C'nin üzerinde sıcaklığa sahip olan bağlantıları bile kullanabilmekte ve en kısa sürede tahliye edilen kullanıcı olmaktadır. Kullanıcı-3'ün tahliyesi boyunca geçtiği bağlantılarda ortalama sıcaklık 31,4 °C iken, Kullanıcı-1'in 27,3 °C, Kullanıcı-2'nin ise 27,4 °C'dir.



Şekil 4.9. Güzergâh üzerindeki sıcaklık-zaman çizelgesi.

Çizelge 4.4'de geliştirdiğimiz akıllı tahliye sistemi, geleneksel tahliye sistemleri ile kıyaslanmıştır. Geleneksel tahliye sisteminden kasıt, dinamik ve akıllı olmayan tahliye sistemleridir. Geleneksel tahliye sistemi yangın başladığında bütün kullanıcı tipleri için aynı çıkış güzergâhını üretmiş ve tahliye süresince bu güzergâhları hiç değiştirmemiştir. Bu nedenle kullanıcılar tahliye süresince sıcaklık değeri 44 °C olan alevli linklerden geçmek zorunda kalmışlardır. Eğer yangın başladığı andan itibaren hemen tahliye süreci başlatılmamış olsaydı, bu kullanıcılar çok daha yüksek sıcaklığa sahip linklerden geçmek zorunda kalacaklardı. Hâlbuki geliştirdiğimiz akıllı tahliye

sistemi, kullanıcıların kişisel özelliklerini ve binadaki değişen şartları da göz önünde bulundurarak her bir kullanıcı için farklı güzergâhlar üretmiştir. Kullanıcı-1, Kullanıcı-2, Kullanıcı-3'ün güzergâhlarındaki en yüksek sıcaklıklar sırasıyla 39 °C, 37 °C ve 43 °C'dir ve bu sıcaklıklar ilgili kullanıcı için tehlike oluşturmamaktadır.

Çizelge 4.4. Akıllı tahliye sistemi ile geleneksel tahliye sistemlerinin karşılaştırılması.

Tahliye Modeli	**Kullanıcı**	**Tahliye Mesafesi (m)**	**Güzergâhtaki Ortalama Sıcaklık (°C)**	**Güzergâhtaki Maksimum Sıcaklık (°C)**
Geleneksel Tahliye Sistemi	Kullanıcı-1	56,2	31,7	44,0
	Kullanıcı -2			
	Kullanıcı -3			
Akıllı Tahliye Sistemi	Kullanıcı -1	118,7	27,3	39,0
	Kullanıcı -2	118,7	27,4	37,0
	Kullanıcı -3	56,2	31,4	43,0

BÖLÜM 5

SONUÇLAR VE ÖNERİLER

Dünya üzerindeki çok katlı, karmaşık binaların ve zamanlarını bu tip binalarda geçiren insanların sayısı gün geçtikçe artmaktadır. Acil durumlarda, özellikle yangın durumunda, bu tip binalardan insanların tahliyesi, geçtiğimiz yıllarda yaşanan birçok olayda da görüldüğü gibi, büyük önem arz etmektedir. Geliştirdiğimiz akıllı tahliye sistemi de bu noktada ihtiyacı giderebilecek nitelikte bir uygulamadır.

Literatürde geliştirilmiş benzer tahliye sistemlerinin çoğunluğu; genel amaçlı, kullanıcının kişisel durumunu dikkate almayan uygulamalardır. Yine birçoğu binadaki değişen şartlara ayak uyduramayan tahliye sistemleridir. Başlangıçta kullanıcı için bir çıkış güzergâhı belirleyip, bütün tahliye süresince bu güzergâhı kullanmaya sevk eden sistemlerdir. Yapılan çalışmaların çoğunluğu, ilk güzergâh üzerinde yangının ve etkilerinin oluşması durumunda, bu güzergâh üzerinde güncelleme yapamayan, diğer bir ifade ile dinamik olmayan tahliye sistemleridir. Bazıları ise sadece linkler üzerindeki sıcaklık artışını dikkate alarak dinamik bir yönlendirme yapabilmektedir. Güzergâh üzerindeki duman, görüş mesafesi, karbon monoksit oranı ve insan yoğunluğunu gibi faktörleri dikkate almadan güzergâh üzerinde güncelleme yapmaktadır. Yine geliştirilen sistemin güncel mobil bir uygulama ile desteklenmesi ve günümüzde aşağı yukarı herkesin bir akıllı telefon sahibi olduğu gerçeği, uygulamanın kullanılabilirlik oranını oldukça artırmaktadır. Bu tip özellikler, geliştirilen bu tahliye sistemini hem akıllı hem de dinamik bir mobil bina içi navigasyon sistemine dönüştürmekte ve diğer tahliye sistemlerinden üstün kılmaktadır.

Yapılan testlerde, geliştirilen sistemin kullanıcının konumunu yeterli hassasiyet ile tespit edebilen, kullanıcıyı kendisi için risk taşımayan alanlardan tahliye edebilen, anlaşılır komutlar üreten ve kullanıcıya güven veren bir tahliye sistemi olduğu

görülmüştür. Böylece kullanıcının komutlarına uymakta tereddüt yaşamadığı bir tahliye sistemi üretilmiştir. Sistem bütün kullanıcıları herhangi bir zarara uğramadan bina dışına çıkarılabilmiştir. Ayrıca bu sistem, yangında değişen bina şartlarını ve kullanıcıların kişisel özelliklerini temel alarak yapay sinir ağı ile kişiye özel çıkış güzergâhları sunabilmekte ve gerektiğinde bu güzergâhlar üzerinde güncelleme yapabilmektedir. Örneğin testlerde bahsedilen Kullanıcı-1 ve Kullanıcı-2 için sistem birden fazla güzergâh hesaplamıştır. Kullanıcılar kendilerine sunulan ilk güzergahta ilerlerken, güzergahın ilerleyen kısımlarında kullanıcılar için risk teşkil eden gelişmeler olmuş ve sistem kullanıcılara için daha güvenli yeni güzergahlar sunabilmiştir. Kullanıcı-1 için üç defa Kullanıcı-2 için ise iki defa güzergâh hesaplanmıştır. Aynı binada aynı yangın senaryosunda Kullanıcı-1'e sunulan ilk güzergahın değişimi 17. saniyede, Kullanıcı-2'ye sunulan ilk güzergâhın değişimi ise 38. saniyede gerçekleşmiştir. Kullanıcıların güzergahlarındaki değişim sayısı ve bu değişimin aynı yangın senaryosunda farklı zamanlarda gerçekleşmesi sistemin hem dinamik hem de kullanıcıların kişisel özelliklerini dikkate aldığını göstermektedir. Ayrıcı tahliye sistemi Kullanıcı-3'ü; genç, atletik, yangın giysisi ve gaz maskesi olan bir kullanıcı olması sebebiyle diğer kullanıcıların çıkamadığı güzergahtan tahliye etmiştir. Çünkü Kullanıcı-3 yüksek sıcaklığa ve duman yayılımına dayanıklı teçhizata sahiptir. Farklı tipteki kullanıcılara farklı güzergahlar sunabilmesi sistemin aynı zamanda akıllı bir sistem olduğunun kanıtıdır. Ayrıca bu durum, sistemin kullanıcıların kişisel özelliklerinin yanı sıra binada bulunan bağlantılardaki insan yoğunluğu, karbon monoksit oranı, yangın büyüme hızı, görüş mesafesi ve bağlantının uzunluğu gibi faktörleri de hesaba kattığını ispatlamaktadır.

Üretilen tahliye sistemi kullanıcıyı öncelikle "EN DÜŞÜK" risk grubundaki bağlantılar üzerinden tahliye etmeye çalışmaktadır. Eğer mevcut çıkış güzergâhlarında riski seviyesi "EN DÜŞÜK" olan bir güzergâh yok ise risk seviyesini "DÜŞÜK" seviyesine artırılarak yeni bir güzergâh önermektedir. Eğer "DÜŞÜK" de yok ise "ORTA", "ORTA" da yok ise "YÜKSEK", "YÜKSEK" de yok ise "EN YÜKSEK" risk seviyesindeki güzergâhı kullanıcıya sunmaktadır. Böylece kullanıcının binanın herhangi bir bölgesinde tıkanıp kalması önlemiş olur. "YÜKSEK" ve "ÇOK YÜSKEK" risk seviyeli güzergâhlar kullanıcı için tehlikeli bağlantılar içerebilir. Fakat

kullanıcının yangında binanın belirli bir bölgesinde çaresiz beklemesinden daha etkili bir çözüm üretilmiş olur.

Kullanıcı akıllı tahliye sisteminden navigasyon hizmeti alabilmek için detaylı bir cihaz grubuna ihtiyaç duymamaktadır. Yapması gereken tek şey, geliştirilen mobil navigasyon uygulamasını gündelik hayatında kullandığı akıllı telefon ya da tablete kurmaktır. Sistemin bu anlamdaki tek dezavantajı tahliye işlemi esnasında kullanıcının bir RFID okuyucuya sahip olma zorunluluğudur. Fakat günümüzde mobil cihazların NFC (Near Field Communication) teknolojisine sahip olduğu gibi, yakın zamanda RFID teknolojisine de sahip olacağı varsayılırsa, sistemin bu dezavantajının da ortadan kalkacağı ön görülmektedir. Böylece geliştirilen sistem, yaygın kullanıma sahip bir tahliye sistemi olmaya aday olacaktır.

Yangın durumunda binada birçok kişinin olacağı gerçeğini de göz önüne alarak geliştirilen tahliye sistemi, aynı anda birçok kullanıcıya hizmet verebilecek şekilde tasarlanmaktadır. Kullanıcılar bu çalışma kapsamında geliştirilen mobil uygulama üzerinden sistemden bireysel tahliye hizmeti alabileceklerdir. Sisteme bağlanabilecek kullanıcı sayısı tamamıyla kullanılan sunucunun özelliğine bağlıdır. Sisteme bağlanan her bir kullanıcı için yeni bir ağ modeli oluşturulacağı düşünülürse, tahliye sisteminin sunucuya yüklediği yük miktarı kullanıcı sayısı × ağ modelinin boyutudur. Bu bağlamda, yapılan testlerde 1000 düğüm içeren bir bina için oluşturulan ağ modeli yaklaşık sunucuda 1MB bellek alanına ihtiyaç duymaktadır. Bu veriler de 1 GB gibi küçük boyutlu bir ana belleğe sahip sunucuda bile yaklaşık 1000 kullanıcının sistemden aynı anda hizmet alabileceğini göstermektedir.

Ayrıca binaların mimari tasarımı aşamasında bu tahliye sistemi gerçek bir yangın simülasyonu ile entegre edilerek ve binada bulunması beklenen insan sayısı da göz önüne alınarak binada bir yangın testi yapılabilir. Böylece binadaki bağlantıların, merdivenlerin ve çıkışların muhtemel bir yangın durumunda binadaki insanların tahliyesi için yeterli olup olmadığı ortaya çıkarılabilir. Simulasyonda elde edilen sonuçlara göre bina tasarımında, merdivenlerin ve çıkışların konumlarında ve sayılarında değişiklik yapılabilir. Böylece yangında daha etkili bir tahliye imkânı sunan bina tasarımları ortaya çıkarılmış olur.

Geliştirilen akıllı tahliye sistemine sadece Android işletim sistemine sahip mobil cihazlardan ulaşılabilmektedir. Gelecekte bu sistemin diğer mobil platformlar olan iOS ve Windows Mobile işletim sitemlerini göre uyarlanması planlanmaktadır. Böylece geliştirilen sistemin daha yangın bir kullanım alanına kavuşması ve daha fazla insan kitlesine hizmet vermesi sağlanmış olacaktır.

KAYNAKLAR

1. Khoury, H. M. and Kamat, V. R., "Evaluation of position tracking technologies for user localization in indoor construction environment", ***Automation in Construction,*** 18 (4): 444-457 (2009).

2. Koyuncu, H. and Yang, S. H., "A survey of indoor positioning and object locating systems", ***IJCSNS International Journal of Computer Science and Network Security***, 10 (5): 121-128 (2010).

3. Demiral, E., Karaş, I. R. and Turan, M. K., "RFID Sistemleri ile Konum Belirleme Uygulamaları", **TMMOB Harita ve Kadastro Mühendisleri Odası**, 14: 14-17 (2013).

4. Choi, B. S., et al., "A hierarchical algorithm for indoor mobile robot localization using RFID sensor fusion", ***Industrial Electronics IEEE Transactions on***, 58(6): 2226-2235 (2011).

5. Lee, H. J., et al., "Localization of mobile robot based on radio frequency identification devices", ***Journal of Institute of Control, Robotics and Systems***, 12 (1): 41-46 (2006).

6. Hahnel, D., et al. "Mapping and localization with RFID technology", ***in Robotics and Automation Proceedings ICRA'04 IEEE International Conference on,*** (2004).

7. Hightower, J., Want R. and Borriello G., "SpotON: An indoor 3D location sensing technology based on RF signal strength", ***UW CSE 00-02-02, University of Washington, Department of Computer Science and Engineering***, Seattle, WA (2000).

8. Bechteler, T. F. and Yenigün H., "2-D localization and identification based on SAW ID-tags at 2.5 GHz.", ***Microwave Theory and Techniques, IEEE Transactions on***, 51 (5): 1584-1590 (2003).

9. Stelzer, A., Pourvoyeur, K., and Fischer, A., "Concept and application of LPM-a novel 3-D local position measurement system", ***Microwave Theory and Techniques, IEEE Transactions on***, 52 (12): 2664-2669 (2004).

10. Ni, L. M., et al., "LANDMARC: indoor location sensing using active RFID ", ***Wireless networks***, 10 (6): 701-710, (2004).

11. Jiang, X., Liu, Y. and Wang, X., "An enhanced approach of indoor location sensing using active RFID. in Information Engineering", ***ICIE'09. WASE International Conference on,*** (2009).

12. Jain, M., Rahul, R. C., and Tolety, S., "A study on Indoor navigation techniques using smartphones. in Advances in Computing", ***International Conference on Communications and Informatics (ICACCI),*** (2013).

13. Chung, J., Pagnini, F., and Langer, E., "Mindful navigation for pedestrians: Improving engagement with augmented reality", ***Technology in Society***, 45: 29-33 (2016).

14. Khalifa, I. H., Kamel A. E. and Barfety, B. "Real time indoor intelligent navigation system inside hypermarkets", ***in Large Scale Complex Systems Theory and Applications***, (2010).

15. Blattner, A., Vasilev, Y. and Harriehausen-Mühlbauer, B., "Mobile Indoor Navigation Assistance for Mobility Impaired People", ***Procedia Manufacturing***, 3: 51-58 (2015).

16. Sriharee, G., "A Symbolic-based Indoor Navigation System with Direction-based Navigation Instruction", ***Procedia Computer Science***, 52: 647-653 (2015).

17. Czogalla, O., "Smart phone based indoor navigation for guidance in public transport facilities", ***IFAC-PapersOnLine***, 48 (10): 233-239 (2015).

18. Koch, C., et al., "Natural markers for augmented reality-based indoor navigation and facility maintenance", ***Automation in Construction***, 48: 18-30 (2014).

19. Barberis, C., et al., "Experiencing indoor navigation on mobile devices", ***It Professional***, 16 (1): 50-57 (2014).

20. Neges, M., et al.,"Combining visual natural markers and IMU for improved AR based indoor navigation", ***Advanced Engineering Informatics***, (2015).

21. Serrão, M., et al., "Indoor localization and navigation for blind persons using visual landmarks and a GIS", ***Procedia Computer Science***, 14: 65-73 (2012).

22. Bellini, P., et al., "Maintenance and emergency management with an integrated indoor/outdoor navigation support", ***Journal of Visual Languages & Computing***, 25 (6): 637-649 (2014).

23. Tsirmpas, C., et al., "An indoor navigation system for visually impaired and elderly people based on Radio Frequency Identification (RFID)", ***Information Sciences,*** 320: 288-305 (2015).

24. Han, S. and Wang, J., "Integrated GPS/INS navigation system with dual-rate Kalman Filter", ***GPS solutions***, 16 (3): 389-404 (2012).

25. Goshen-Meskin, D. and Bar-Itzhack, I.Y., "Unified approach to inertial navigation system error modeling", ***Journal of Guidance Control and Dynamics***, 15 (3): 648-653 (1992).

26. Cheng, J., et al., "Seamless outdoor/indoor navigation with WIFI/GPS aided low cost Inertial Navigation System", ***Physical Communication***, PA, 13: (2014).

27. Boysen, M., et al., "Constructing indoor navigation systems from digital building information", ***IEEE 30th International Conference on Data Engineering***, (2014).

28. Park, I., et al., "*Time-dependent optimal routing in micro-scale emergency situation*", ***Mobile Data Management: Systems, Services and Middleware MDM'09 Tenth International Conference***, IEEE (2009).

29. Jeo-Lee, S.Z., "3D Data Model for Representing Topological Relations of Urban Areas", Geospatial Information Technology for Emergency Response, J.L. Sisi Zlatanova (Editor), ***Taylor&Francis Group***, London,UK, 143-168 (2008).

30. Pu, S. and Zlatanova S., "Evacuation route calculation of inner buildings", Geo-information for disaster management, ***Springer***, 1143-1161 (2005).

31. Kwan, M.-P. and Lee J., "Emergency response after 9/11: the potential of real-time 3D GIS for quick emergency response in micro-spatial environments", ***Environment and Urban Systems***, 29 (2): 93-113 (2005).

32. Inoue, Y., et al., "Indoor emergency evacuation service on autonomous navigation system using mobile phone", ***Universal Communication ISUC'08. Second International Symposium,*** IEEE, (2008).

33. Chu, L. and Wu, S. J. "A real-time decision support with cloud computing based fire evacuation system", ***Nano Information Technology and Reliability (NASNIT) 15th North-East Asia Symposium, (***IEEE) 2011.

34. Pang, G.K., et al., "Adaptive route selection for dynamic route guidance system based on fuzzy-neural approaches", ***IEEE Transactions on Vehicular Technology***, 48 (6): 2028-2041 (1999).

35. Resmi Gazete, "Binaların yangından korunması hakkında yönetmelik", ***Cumhuriyeti*** , 47 (5) (2007).

36. Şahin, M., "Üst Düzey Programlama", (2015).

37. İnternet: Stefan, Z., "Servlet Essentials", **http://www.novocode.com/doc/servlet-essentials/chapter1.html** (2015)

38. İnternet: "Servlet Tutorials", **https://www.tutorialspoint.com/servlets/index.htm** (2015)

39. İnternet: Marty, H. and B. Larry, "Servlet Basics", **http://www.pdf.coreservlets.com** (2015).

40. İnternet: Mughal, K. A., "Core Servlets", **http://www.ii.uib.no/~khalid/atij/atij-core-servlets-web/atij-core-servlets-2x1.pdf** (2015).

41. Jakob, J., "Java Servlets", **http://tutorials.jenkov.com/java-servlets/index.html** (2015).

42. Martin, Z. and W. Rafael., "Web Database Access with Apache/Tomcat." **http://www.akadia.com/download/soug/tomcat/html/tomcat_apache.html#Tomcat** (2015).

43. Graham, E., "Tomcat Overview", ***Well Hosuse Consultants***, United Kingdom. 345-348.

44. Heffelfinger, D. R., "Java EE 5 Development Using GlassFish Application Server", ***Packt Publishing Ltd.***, (2007).

45. İnternet: "GlassFish Server Application Deployment Guide", **https://docs.oracle.com**/ (2016).

46. İnternet: "Netbeans Documentation", **https://netbeans.org/kb/index.html** (2016).

47. İnternet: David, M. "JDBC Connection Pooling Best Practices", **http://www.javaranch.com/journal/200601/JDBCConnectionPooling.html** (2016).

48. Purushothaman, J., "Configuring JNDI Data Source connecting to Oracle DB in GlassFish Server", ***Binaries*** (2015).

49. İnternet: "Interface ServletContext", **https://docs.oracle.com/javaee/6/api/javax/servlet/ServletContext.html** (2016).

50. İnternet: *"Interface ServletContextListener"*, **https://docs.oracle.com/javaee/6/api/javax/servlet/ServletContextListener.html** (2016).

51. Lee, J. K. and Lee, J. Y., "Android programming techniques for improving performance", ***3rd International Conference on Awareness Science and Technology (iCAST)***, IEEE (2011).

52. Sonuç, E., Ortakci, Y. and Elen, A., "Karabük Üniversitesi Bilgi Sistemi Android Uygulaması", (2013).

53. Bing, H. "Analysis and research of system security based on android", ***Intelligent Computation Technology and Automation (ICICTA) Fifth International Conference***, IEEE (2012).

54. DiMarzio, J.F., "Android", ***Tata McGraw-Hill Education*** (2008).

55. Peng, B., Yue, J. and Tianzhou, C., "The Android Application Development College Challenge", ***High Performance Computing and Communication & IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS)*** (2012).

56. Kayande, D. and Shrawankar, U., "Performance analysis for improved RAM utilization for Android applications". ***Software Engineering (CONSEG) CSI Sixth International Conference,*** IEEE (2012).

57. İnternet: "Android (Operating System)″, **https://en.wikipedia.org/wiki/Android_(operating_system)** (2016).

58. İnternet: Android Devolopers, "Dashboards″, **https://developer.android.com/about/dashboards/index.html** (2017).

59. Milette, G. and Stroud, A., "Professional Android sensor programming", ***John Wiley & Sons*** (2012).

60. İnternet: Android Devolopers, "Handler", **https://developer.android.com/reference/android/os/Handler.html** (2016).

61. İnternet: Android Devolopers, "Timer", **http://developer.android.com/reference/java/util/Timer.html** (2016).

62. İnternet: Android Devolopers, "AsyncTask", **http://developer.android.com/reference/android/os/AsyncTask.html** (2016).

63. Atila, U., " 3B CBS kapsamında çok katlı bina yangınlarına yönelik akıllı bireysel tahliye modelinin yapay sinir ağı ile tasarlanması", Doktora Tezi, , Karabük, 217 (2013).

64. Kothuri, R., Godfrind , A. and Beinat, E., "Pro Oracle Spatial For Oracle Database 11g", ***Dreamtech Press*** (2008).

65. Döner, F. and Bıyık, C., "Üç boyutlu nesnelerin konumsal veritabanında yönetimi″, ***HKM Jeodezi*** , 1(100): 27-33 (2009).

66. Kazar, B. M., et al., "On valid and invalid three-dimensional geometries", ***Advances in 3D Geoinformation Systems,*** Springer, 19-46 (2008).

67. Murray, C., "Oracle Spatial Developer's Guide 6 Coordinate Systems″, ***Spatial Reference Systems***, (2009).

68. ECMA, "The JSON Data Interchange Format", ***ECMA International*** (2013).

69. Internet: JSON, "Introducing JSON", **http://www.json.org** (2015).

70. Demiral, E., "Üç boyutlu (3B) coğrafi bilgi sistemleri kapsamında iç mekânlara yönelik RFID tabanlı konum belirleme sistemi tasarımı ", Yüksek Lisans Tezi, ,
Karabük, 103 (2014).

71. Khong, G. and White, S., "Moving right along: Using RFID for collection management at the parliamentary library", ***Austl. L. Library***, 14: 29 (2006).

72. Masse, D., "RFID handbook: fundamentals and applications in contactless smart cards and identification second edition", ***Microwave Journal***, 47(10): 168-169 (2004).

73. Bhuptani, M. and Moradpour, S., "RFID Field Guide: Deploying Radio Frequency İdentification Systems", ***Prentice Hall PTR***, (2005).

74. İnternet: Android Devolopers, "Bluetooth", **https://developer.android.com/guide/topics/connectivity/bluetooth.html** (2015).

75. İnternet: Android Devolopers, "Wi-Fi Peer-to-Peer", **https://developer.android.com/guide/topics/connectivity/wifip2p.html** (2015).

76. Gavin, H., "The Levenberg–Marquadt Method for Nonlinear Least Squares Curve-Fitting Problems", ***Durham, NC: Duke University*** (2011).

77. Igel, C. and Hüsken M., "Empirical evaluation of the improved Rprop learning algorithms", ***Neurocomputing***, 50: 105-123 (2003).

78. Lewis, A.S. and Overton, M.L., "Nonsmooth optimization via BFGS. submitted to SIAM", ***J. Optimization***, 1-35 (2009).

79. Şerbetçi, M. and Atasoy, V., "Jeodezik Hesap", ***Trabzon Karadeniz Teknik*** (1990).

80. Karaş, İ.R., "Objelerin topolojik ilişkilerinin 3B CBS ve ağ analizi kapsamında değerlendirilmesi", Doktora Tezi,

, Yıldız Teknik Üniversitesi (2007).

ÖZGEÇMİŞ

Yasin ORTAKCI 1981 yılında Karabük'te doğdu; ilk ve orta öğrenimine aynı şehirde devam etti. 1996 yılında Karabük Demir Çelik Yabancı Dil Ağırlıklı Lisesi'nde başladığı lise öğrenimini 1999 yılında tamamladı. Aynı yıl Sakarya Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü'nde lisans öğrenime başladı ve 2003 yılında buradan mezun oldu. 2003-2004 yılları arasında asteğmen olarak askerlik görevini yerine getirdi. 2005 yılında 6 ay kadar süren özel sektör deneyiminden sonra yine 2005 yılında Zonguldak Karaelmas Üniversitesi Safranbolu Meslek Yüksekokulunda okutman olarak göreve başladı. 2007 yılında Zonguldak Karaelmas Üniversitesinden ayrılan Karabük Üniversitesi bünyesinde okutmanlığa devam etti. 2008 yılında Karabük Üniversitesi Bilgisayar Mühendisliği bölümünde yüksek lisans eğitimine başladı Yine aynı yılda halen devam etmekte olduğu Karabük Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü Yazılım Anabilim Dalında okutman olarak göreve başladı. 2011 yılı haziran ayında yüksek lisanstan mezun olarak Bilgisayar Mühendisliği Anabilim Dalında doktora eğitimine başladı. Doktora eğitimini 2017 yılında tamamlayan Yasin ORTAKCI, evli ve biri kız biri erkek olmaz üzere iki çocuk babasıdır.

ADRES BİLGİLERİ

Adres : Karabük Üniversitesi Mühendislik Fakültesi
Bilgisayar Mühendisliği Bölümü,
Demir Çelik Kampüsü, 78050, KARABÜK

Tel : +90 555 487 83 63

E-posta : yasinortakci@karabuk.edu.tr