T.C.
ALTINBAŞ ÜNİVERSİTESİ

Fen Bilimleri Enstitüsü / Bilişim Teknolojileri

COĞRAFİ BİLGİ SİSTEMLERİ KULLANARAK KONUM TABANLI İŞ VE ELEMAN ARAMA UYGULAMASI

Atilla Gökhan Kartal

Yüksek Lisans

Dr. Öğr. Üyesi Oğuz Ata

İstanbul, 2019

COĞRAFİ BİLGİ SİSTEMLERİ KULLANARAK KONUM TABANLI İŞ VE ELEMAN ARAMA UYGULAMASI

Atilla Gökhan Kartal

Bilişim Teknolojileri

Fen Bilimleri Enstitüsü'ne

Yüksek Lisans

tezi olarak sunulmuştur.

ALTINBAŞ ÜNİVERSİTESİ

2019

Bu çalışma tarafımızca incelenmiş olup, kapsam ve kalite açısından Yüksek Lisans tezi olmaya yeterli bulunmuştur.

Dr. Öğr. Üyesi Oğuz ATA

Danışman

İnceleme Komitesi Üyeleri (İlk isim jüri başkanına, ikinci isim tez danışmanına aittir.)

Prof. Dr. Hasan Hüseyin BALIK	Hava Harp Okulu, Milli Savunma Üniversitesi	
Dr. Öğr. Üyesi Oğuz ATA	Mühendislik ve Doğa Bilimleri Fakültesi, Altınbaş Üniversitesi	
Dr. Öğr. Üyesi Çağatay AYDIN	Mühendislik ve Doğa Bilimleri Fakültesi, Altınbaş Üniversitesi	

Bu çalışma bir Yüksek Lisans tezinin tüm gerekli şartlarını taşımaktadır.

Dr. Öğr. Üyesi Oğuz ATA

Bölüm Başkanı

Prof. Dr. Oğuz BAYAT

Enstitü Müdürü

Fen Bilimleri Enstitüsü onayı: 23 / 05 / 2019

Bu dokümandaki tüm bilgilerin akademik kural ve etiğe bağlı kalınarak yazıldığını ve tez yazım kuralları kapsamında bu çalışmada bulunan ve özgün olmayan bütün bilgi ve materyallerin referanslandırıldığını temin ederim.

Atilla Gökhan Kartal

İTHAF

Almış olduğum eğitimlerin her safhasında maddi ve manevi tüm imkanlarını seferber eden babam İbrahim Kartal'a ve annem Zeliha Kartal'a, yüksek lisans eğitimim boyunca moral ve motivasyon desteğini esirgemeyen kıymetli eşim Meryem Kartal'a, evimin neşe kaynağı çocuklarım Muhammed Furkan Kartal'a ve Ömer Faruk Kartal'a ithaf edilmiştir.

TEŞEKKÜR

Tez çalışmalarımda yardımlarını ve desteklerini esirgemeyen Serkan Altuntaş'a, yöneticim Beslan Çelikcan'a ve tez danışmanım Dr. Öğr. Üyesi Oğuz Ata'ya teşekkürler.

ÖZET

COĞRAFİ BİLGİ SİSTEMLERİ KULLANARAK KONUM TABANLI İŞ VE ELEMAN ARAMA UYGULAMASI

Atilla Gökhan Kartal,

Yüksek Lisans, Bilişim Teknolojileri, Altınbaş Üniversitesi,

Danışman: Dr. Öğr. Üyesi Oğuz Ata

Tarih: 05, 2019

Sayfa Sayısı: 62

Trafik ve ulaşım sorunu, günümüzde insanların çalışma hayatını olumsuz etkileyen faktörlerden biri olarak değerlendirilir. Trafikte geçen zaman, çalışanların iş gücüne de olumsuz yansır ve iş verimini düşürür[1]. Trafik ve ulaşım sorununa bağlı olarak çalışanların iş veriminin düşmemesi için insan kaynakları uygulamalarının bu faktörleri dikkate alması gerekmektedir. Mevcut uygulamalar Coğrafi Bilgi Sistemleri ile etkin bir şekilde entegre çalışmadığı için ilanlar mekânsal ölçütler olmadan veya kısıtlı ölçütlerle, yazılı içerik olarak arama yapmaya olanak sağlar. Bu problemi çözmek için bu tez çalışmasında, iş arayan adayların ikamet adreslerine daha yakın konumlardaki ilanları değerlendirmesi veya işverenlerin işin konumuna daha yakın adayları tercih etmesi için çözüm sunmaya yönelik bir model uygulama geliştirilmiştir. Mevcutta kullanılan uygulamalara ek olarak Coğrafi Bilgi Sistemleri fonksiyonları daha etkin kullanılmaya çalışılmış ve ulaşım sorununu en aza indirgemek için bir çözüm sunmak hedeflenmiştir.

Anahtar kelimeler: CBS, İş Arama, Eleman Arama

ABSTRACT

LOCATION BASED JOB AND CANDIDATE SEARCH APPLICATION USING GEOGRAPHICAL INFORMATION SYSTEMS

Kartal, Atilla Gökhan,

M.S., Information Technology, Altınbaş University,

Supervisor: Asst. Prof. Oğuz Ata

Date: 05, 2019

Pages: 62

The problem of traffic and transportation is considered as one of the factors that negatively affect the working life of people. The time spent in traffic is also negatively reflected in the workforce of the employees and decreases the work efficiency[1]. Human Resources applications need to take into account these factors in order to not reduce the work efficiency of the employees due to traffic and transportation problems. Because existing applications do not work effectively with Geographic Information Systems, job postings allow search without written spatial criteria or with restricted criteria as written content. In order to solve this problem, in this thesis, a model application has been developed in order to provide a solution for job seekers to evaluate job postings in positions closer to their residence addresses or for employers to choose candidates who are closer to the position of the job. In addition to the applications currently used, the functions of Geographical Information Systems are tried to be used more effectively and it is aimed to provide a solution to minimize the transportation problem.

Keywords: GIS, Job Search, Employee Search

İÇİNDEKİLER

Sayfa

ÖZET ..... **vii**

ABSTRACT ..... **viii**

ŞEKİL LİSTESİ ..... **xi**

KISALTMALAR LİSTESİ ..... **xiii**

1. GİRİŞ **1**

1.1 TEZİN AMACI 1

1.2 LİTERATÜR TARAMASI 2

2. MATERYAL **4**

2.1 COĞRAFİ BİLGİ SİSTEMLERİ (CBS) 4

2.2 WEB-CBS 4

2.3 WMS 6

2.4 GEOSERVER 6

2.5 JAVA 7

2.6 APACHE TOMCAT 7

2.7 VAADIN PLATFORMU 7

2.8 ORM PLATFORMU 8

2.8.1 Hibernate 9

2.9 SPRING PLATFORMU 10

2.10 MAVEN 11

2.11 POSTGRESQL 12

2.12 POSTGIS 12

2.13 OPENLAYERS 12

2.14 OPEN STREET MAP (OSM) 13

2.15 KATMANLI MİMARİ 13

3. METOT **15**

3.1 PROJENİN GENEL YAPISI 15
3.1.1 Veri Tabanı 16
3.1.2 Projenin Oluşturulması (Proje:careeronmap) 16
3.1.2.1 Çekirdek modülü (Modül:careeronmapcore) 18
3.1.2.2 Kullanıcı arayüzü modülü (Modül:careeronmapui) 24
4. UYGULAMA 30
5. SONUÇ 46
6. GELECEK ÇALIŞMALAR 47
KAYNAKÇA 48
EK A 52
EK B 53
ÖZGEÇMİŞ 54

ŞEKİL LİSTESİ

Sayfa

Şekil 2.1: Web CBS mimarisi[10] 5

Şekil 2.2: Vaadin uygulama mimarisi[18] 8

Şekil 2.3: Veritabanı varlık sınıfları ve ORM'nin nesne manipülasyonunu SQL sorgularına nasıl dönüştürdüğünün örneği[20] 9

Şekil 2.4: Hibernate mimarisi[22] 10

Şekil 2.5: Spring platformu fonksiyon modülleri[26] 11

Şekil 3.1: Projenin genel yapısı 15

Şekil 3.2: PostGIS eklentisini aktif etmek için çalıştırılacak SQL komutu 16

Şekil 3.3: Dizin oluşturma ve dizine erişim komutları 17

Şekil 3.4: Komut satırından projeyi oluşturma 17

Şekil 3.5: pom.xml dosyasındaki değiştirilecek bölüm 17

Şekil 3.6: Çekirdek modülünü oluşturma 18

Şekil 3.7: Kullanıcı arayüzü modülünü oluşturma 18

Şekil 3.8: Entity sınıfı örneği 19

Şekil 3.9: BaseDao sınıfı 20

Şekil 3.10: Entity özelinde bir veri tabanına erişim işlemini gösteren örnek metot 21

Şekil 3.11: BaseService sınıfı 22

Şekil 3.12: applicationContext.xml dosyasından Hibernate ile ilgili bir bölüm 23

Şekil 3.13: BaseServiceTest sınıfı 24

Şekil 3.14: Admin türünde bir kullanıcıyı veri tabanına ekleme sorgusu..... 25

Şekil 3.15: Admin kullanıcı türü için kullanım durum diyagramı..... 26

Şekil 3.16: İş Arayan ve İşveren kullanıcıların ana sayfa için kullanım durum diyagramı..... 29

Şekil 4.1: Anonim kullanıcı için ana sayfa görünümü..... 31

Şekil 4.2: Kullanıcı giriş ekranı 32

Şekil 4.3: Sistemin gereksinim duyduğu verilerin tanımlandığı ekran..... 33

Şekil 4.4: İşveren kullanıcı üyelik detay ekranı..... 34

Şekil 4.5: İşveren kullanıcı ilan tanım ekranı 35

Şekil 4.6: İş ilanında, İş Arayan kullanıcıdan beklenen bilginin tanımlandığı ekran..... 36

Şekil 4.7: İşveren kullanıcı ana sayfa görünümü 37

Şekil 4.8: İş Arayan kullanıcı üyelik detay ekranı..... 38

Şekil 4.9: İş Arayan kullanıcı ana sayfa görünümü 39

Şekil 4.10: Mesafeye göre renklendirme görünümü..... 41

Şekil 4.11: Çizime göre renklendirme görünümü..... 43

Şekil 4.12: Dış kaynaktan temin edildiği varsayılan emlak ilanları ve detayının görünümü 45

KISALTMALAR LİSTESİ

API	:	Application Programming Interface
ASF	:	Apache Software Foundation
CBS	:	Coğrafi Bilgi Sistemi
DAO	:	Data Access Object
GIF	:	Graphics Interchange Format
GiST	:	Generalized Search Tree
GML	:	Geography Markup Language
GPS	:	Global Positioning System
HTTP	:	HyperText Transfer Protocol
JAVA EE	:	Java Enterprise Edition
JAVA EL	:	Java Expression Language
JPEG	:	Joint Photographic Experts Group
JSP	:	Java Server Pages
OGC	:	Open Geospatial Consortium
ORM	:	Object Relational Mapping
OSM	:	Open Street Map
PNG	:	Portable Network Graphics
POM	:	Project Object Model
SQL	:	Structural Query Language

UCB	:	University of California at Berkeley
UI	:	User Interface
VTYS	:	Veri Tabanı Yönetim Sistemi
WFS	:	Web Feature Service
WMS	:	Web Map Service
XML	:	Extensible Markup Language

1. GİRİŞ

Trafik ve ulaşım sorunu, günümüzde insanların çalışma hayatını olumsuz etkileyen faktörlerden biri olarak değerlendirilir. Özellikle büyükşehirlerde iş ve ikamet yerleri arasındaki ulaşım sorununa bağlı olarak kaybedilen zaman, insanların sosyal hayatlarına da olumsuz anlamda etki eder. Bu da insanların stresli ve gerilimli bir hayat sürmelerine, iş yaşamlarında ciddi performans kayıplarına neden olur[1]. İş veriminin düşmemesi ve stresin çalışanlar üzerinde yaptığı baskıyı azaltmak için bu tür olumsuz faktörleri insan kaynakları yönetiminin dikkate alması gerekmektedir.

Bir insan kaynakları yönetimi fonksiyonu olarak işe alım, bir kurumun performansını ve gelişimini en kritik şekilde etkileyen faaliyetlerden biridir. E-işe alım ise, elektronik kaynakları, özellikle interneti kullanan personel alım sürecidir. Literatürde ayrıca çevrimiçi işe alım, internet işe alım veya siber işe alma olarak da bilinen e-işe alım, çevrimiçi olarak ilanların yayınlanmasına karşılık gelir. Şirketler ve işe alım acentaları işe alım süreçlerinin çoğunu çevrimiçi olarak yürütürler; böylece iş arayan adayların güncel iş fırsatları ile eşleşme hızlarını arttırırlar. İş arayanlar veri tabanı teknolojilerini, çevrimiçi iş ilanlarını ve arama motorlarını kullanarak daha önce mümkün olandan daha kısa sürede başvurularını yapabilirler[2].

Günümüzde de bu tür birçok e-işe alım uygulaması mevcuttur. Bunların kimisi, iş arayanların ve iş verenlerin mekânsal konum bilgileri kullanılmadan tamamen yazılı adres verilerine dayalı olarak arama yapmaya olanak sağlarken, kimisi konum bilgilerini mekânsal olarak harita üzerinde göstererek arama yapmaya olanak sağlar. Ancak ne var ki, son yıllarda çevrimiçi mekânsal emlak ilanları aramadan, akıllı kent sistemlerine kadar birçok farklı sektörde kullanılan Coğrafi Bilgi Sistemleri fonksiyonları insan kaynakları web uygulamalarında aktif olarak kullanılmamıştır.

1.1 TEZİN AMACI

Bu tezde, mevcutta var olan insan kaynakları web uygulamalarına ek olarak Coğrafi Bilgi Sistemleri fonksiyonlarının etkin bir şekilde kullanılması hedeflenmiş ve bunu sağlamak için kullanıcılarının mekânsal ölçütlere göre sorgulamalar da yapmasına olanak sağlayan bir model insan kaynakları web uygulaması geliştirilmiştir.

Bu model uygulamaya göre kullanıcılar iş imkanlarını veya iş arayanları mekânsal olarak harita üzerinde görüntüler ve mekânsal ölçütlere göre arama yaparlar. Ayrıca kullanıcılar arama sonuçlarını bir renklendirme ölçütüne göre harita üzerinde farklı renklerle gruplayarak görüntülerler. Böylelikle iş arayan kullanıcılar; tercih ettikleri renklendirme ölçütüne göre iş ilanlarının, ikamet adreslerine olan yakınlığını veya uzaklığını net bir şekilde ayırt ederler. Aynı şekilde iş veren kullanıcılar da iş arayan kullanıcıları sorgularlarken bu ölçütlerden faydalanırlar.

Kimi zaman iş imkanları mevcut ikamet adresine yakın olmaz ve adres değişikliği düşünülebilir. Ancak iş nedeniyle ikamet adresini değiştirmek maddi olarak ek bir maliyet de çıkarabilir. Çözüm olarak; iş imkanının olduğu bölgedeki emlak ilanlarını iş arayan kullanıcılara göstermek ve kullanıcının karar vermesine destek olmak amaçlanmıştır. Bunun için de dış kaynak olarak temin edileceği düşünülen emlak ilanları harita üzerinde gösterilmiştir.

1.2 LİTERATÜR TARAMASI

Ülkemizde ve dünyada Coğrafi Bilgi Sistemleri farklı alanlarda farklı amaçlarla kullanılmaktadır.

Yalçın İ., yapmış olduğu tez çalışmasında; üniversitesinin coğrafi uygulamalarına teknolojik altyapı sağlaması için, web üzerinden hizmet veren bir Coğrafi Bilgi Sistemi'nin geliştirilmesinde ihtiyaç duyulabilecek bileşenleri ve yazılımları incelemiş ve detaylı olarak tanıtmıştır. Ayrıca incelenen bileşen ve yazılımların bazılarının da kullanıldığı örnek bir sistem tasarlamıştır. Bu sistemde, kullanıcılar bir mobil uygulama üzerinden deprem anında hissettikleri sarsıntı derecesini, konum bilgisini, kullanıcı bilgilerini vb. sisteme kaydeder. Bir web uygulaması arayüzü üzerinden de girilen bu bilgiler görselleştirilir. Sonuç olarak, teknolojik altyapı çalışması ve geliştirilen sistem tasarımı ile farklı uygulamaların geliştirilmesi için bir altyapı oluşturulmuştur[3].

Dinç O., tez çalışması kapsamında; kampüs sınırları içerisindeki insanların, kampüslerdeki yaşamlarını kolaylaştırmak için örnek bir Web CBS uygulaması geliştirmiştir. Üniversiteye bağlı kampüsleri, kampüsler içerisindeki fakülte, kütüphane, spor tesisi vb. gibi önemli mekânsal verileri bir program aracılığı ile oluşturup, veri tabanında depolamıştır. Web CBS bileşenlerini kullandığı bir web uygulaması ile de bu mekânsal verileri son kullanıcıya farklı sembollerle

göstermiştir. Böylelikle kampüs içerisindeki önemli mekânsal veriler web ortamında yayınlanarak, geniş bir kullanıcı kitlesi için daha az maliyetle erişilebilir hale gelmiştir[4].

Amaneddine N. ve Chmeit N., yayınlamış oldukları makalede; Lübnan'da emlak aramak ve araştırmak için var olan yaklaşımların bazı eksiklikleri olduğunu ifade etmişlerdir. CBS ile bütünleşik olmayan bu yaklaşımlarda, mülklerin grafiksel gösteriminin sunulamadığından, coğrafi alanların mesafe ve yükseklik hesaplamaları ile ilgili hizmetleri kapsamamasından, trafik bilgilerinin modellenmemesinden, popülasyon oranlarının olmayışından vb. bahsetmişlerdir. Bu eksikliklere çözüm olarak, Lübnan çevresindeki farklı tipte emlak ve mülklerin grafiksel gösterimini sağlayarak kullanıcıların araştırmasını, keşfetmesini, mülklerin görünümü ve çevresi hakkında fikir edinmelerini sağlayacak bir uygulama geliştirmişlerdir. Böylelikle grafiksel gösterimin olduğu, kullanıcıların daha kolay arama yapabileceği bir ortam oluşturulmuştur[5].

Sezer A., Deniz M. ve Topuz M., yapmış oldukları çalışmada; çeşitli kurumlardan temin ettikleri verilere göre CBS bileşenlerini kullanarak, devlet okullarının mekânsal erişilebilirlik durumlarını incelemiş, öğrenci, öğretmen, derslik vb. sayılarına göre analiz etmişlerdir. Çalışmanın neticesi olarak; lise düzeyindeki okullarda sorun görülmediği, bazı bölgelerdeki anaokullarının yetersiz kaldığı, ilkokulların şehrin merkezinde kümelendiği gibi bulgular tespit etmişlerdir[6].

2. MATERYAL

Bu bölümde, tez çalışması kapsamında geliştirilen model uygulamanın ihtiyaçları doğrultusunda kullanılan teknolojiler hakkında bilgiler verilmiştir.

2.1 COĞRAFİ BİLGİ SİSTEMLERİ (CBS)

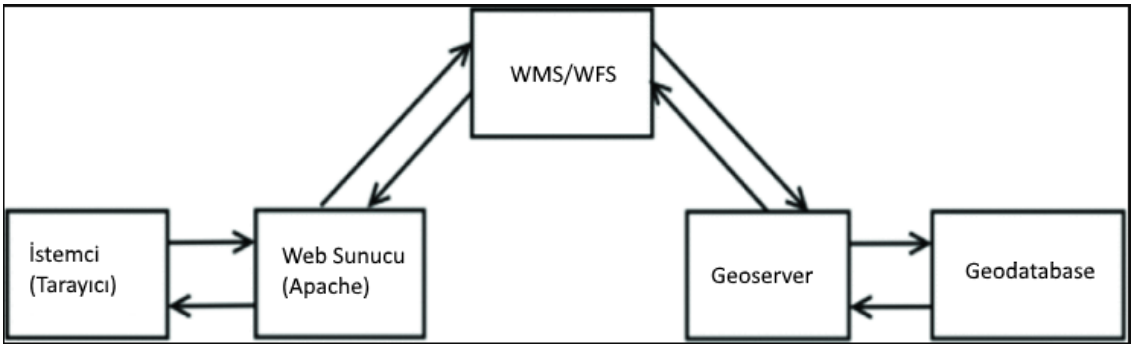
Coğrafya Bilgi Sistemi (CBS), sıklıkla mekânsal veri olarak adlandırılan coğrafi verilerin toplanmasını, saklanmasını, yönetilmesini ve yorumlanmasını amaçlayan bir sistemdir. Jeoloji, jeo-teknikler, navigasyonlar, ölçme, evrensel konumlandırma sistemi (GPS), uzaktan algılama ve fotogrametri dâhil olmak üzere birçok farklı aracı kapsar[7]. CBS başlangıçta bilişim sistemlerine bağlı kantitatif haritalama öğelerinin yalnızca bir birleşimiydi. Altmışlı yılların başlarında bilgilerini ve ihtiyaçlarını teknolojiye uyarlamaya çalışan haritacılar ve coğrafyacılar tarafından kullanılan bir alandı. Yetmişlerin başında, gelişme disiplini ve bu disiplinin istikrara kavuşması, bilişim temelli coğrafi bilgi sistemleri ile sonuçlandı. 1987 yılında Uluslararası Coğrafi Bilgi Sistemleri Dergisi doğdu. Bir yıl sonra, CBS'ye adanmış ilk çevrimiçi dağıtım listesi, Buffalo'daki New York Eyalet Üniversitesi'nde kuruldu ve bu alan için periyodik yayın başladı. Bu aracın popülerleşmesi, kişisel bilgisayarların da daha yaygın hale gelmesi ile birlikte CBS'yi tüm araştırmacılar için çeşitli alanlarda kullanışlı hale getirmiştir. Kullanıcıların her tür coğrafi veriye erişmelerine izin veren web tabanlı Maps gibi Google hizmetlerinin ve birçok uygulamanın ortaya çıkması buna bir örnektir. CBS, deneyimsiz kullanıcıların, yeteneklerini kullanmasına ve yararlanmasına olanak tanıyan yararlı bir sistemdir[8]. CBS, insan, yöntem, yazılım, teçhizat ve verileri içeren beş ana bileşene ayrılmıştır. İnsan, veri tabanını yöneten, sistemi analiz eden ve programcı olarak hareket eden kişiyi ifade eder. Yöntem, belli verilerin girilme, saklanma, analiz edilme ve sisteme yüklenme yoludur. Diğer taraftan yazılım ise veriyi giriş için paketleyen, saklayan, yöneten ve analiz eden bileşendir. Bunun dışında, CBS'yi uygulayabilecek bir bilgisayar sistemi gibi teçhizatlar teknik açıdan gereklidir[9].

2.2 WEB-CBS

Başlangıçta CBS yazılımları masaüstü uygulamaları olarak mevcuttu, ancak internet teknolojisinin gelişimi coğrafi bilgilerin web üzerinden yayılmasını mümkün kıldı. CBS ve web teknolojisinin bu birleşimi, yaygın olarak web tabanlı CBS veya kısaca WEB-CBS olarak bilinir.

Bir WEB-CBS uygulaması genellikle konumla ilgili bilgileri, kullanıcının yakınlaştırıp uzaklaştırabildiği, nesneleri arayabildiği ve tanımlayabileceği etkileşimli dijital çevrimiçi bir harita olarak gösterir. WEB-CBS, orman alanı, madencilik bölgesi, turistik noktaları ve daha pek çok konum bilgisini yayınlamak için kullanılmıştır. WEB-CBS uygulamalarının çoğu, etkileşimli çevrimiçi harita üzerinden sadece bilgi görüntüler. Bazı WEB-CBS uygulamaları, kullanıcıların coğrafi verilerle ilişkili ilgili mekânsal olmayan bileşenler eklemelerine veya düzenlemelerine izin verir. İnternet teknolojisinin ve coğrafi bilgi sistemlerinin gelişmesi ile coğrafi verilerin mekânsal bileşenini de web üzerinden düzenlemek mümkün olmuştur. Bu özellik ilk olarak ticari CBS yazılımlarında mevcuttu, ancak günümüzde coğrafi verilerin web üzerinden düzenlemesi açık kaynaklı CBS yazılımları kullanılarak da yapılabilir.

Şekil 2.1, Hyper Text Transfer Protocol (HTTP) olarak adlandırılan web protokolü yoluyla transfer edilen verinin istemci ve sunucu arasındaki etkileşimini göstermektedir. İstemci mekânsal veri talep ettiğinde, web sunucusu gelen istekler doğrultusunda internetten Web Harita Hizmeti (WMS) veya Web Özellik Hizmeti (WFS) olarak adlandırılan biçimlerde coğrafi veri hizmetleri alır. Bu tür web hizmetleri bir mekânsal veri sunucusu tarafından yönetilir ve yayınlanır, örneğin mekânsal veri tabanları ile çalışan GeoServer'dan.



Şekil 2.1: Web CBS mimarisi[10]

WMS ve WFS, coğrafi veri hizmetlerinin en yaygın biçimleridir. Coğrafi veriler WMS olarak yayınlandığında, JPEG, GIF veya PNG formatında imajlar olarak tüketilebilirler. WMS, coğrafi verileri çevrimiçi bir harita biçiminde görüntülemek için idealdir, ancak coğrafi verilerin düzenlenmesine izin vermez. Öte yandan, mekânsal veriler WFS olarak yayınlanırsa, GML veya GeoJSON gibi vektörel veri olarak tüketilebilirler. Bu tür coğrafi veri hizmetleri, coğrafi verinin yanı sıra geometrik bilgisini (yani, şekli oluşturan köşelerin şekilleri ve koordinatlarını) içerir.

Bu tür coğrafi veri hizmetleri bu web haritalarını görüntülemek için daha fazla bellek ve zaman gerektirse de, kullanıcıların coğrafi verileri web üzerinden düzenlemelerine izin verir[10].

2.3 WMS

Web Harita Servisi (WMS), mekânsal içerikli haritaları coğrafi bilgilerden dinamik olarak sunan bir OGC standardıdır. WMS, coğrafi bilgi kümelerinin bir bilgisayar ekranında görüntülenmeye uygun dijital görüntü dosyalarını içerir[11]. WMS, coğrafi verileri harita olarak gerçek coğrafi verilerden dinamik oluşturulan PNG, GIF veya JPEG gibi mekânsal olarak referans verilen bir resim olarak sunar. Belirli bir WMS erişimi için iş akışı şu şekilde tanımlanır: Sistem ilk önce *GetCapabilities* isteği ile yayınlanan katmanları ve bunların kapsamı, stili ve sorgulama yetenekleri hakkındaki bilgileri keşfeder. Yayınlanan katmanlar bir ağaç görünümü olarak düzenlenir ve görüntülenir. *GetCapabilities* sonucunu inceledikten sonra, bir sonraki adım kullanıcının seçtiği ve kutu ile sınırlandırdığı katmana bağlı olarak *GetMap* isteğini yapılandırmaktır. Uzak veri sunucusundan yanıt alındıktan sonra, yeni bir görüntü katmanı oluşturulur ve görüntülemek için harita görüntüleyicisine eklenir. Bu adım tamamlandıktan sonra sistem, *GetFeatureInfo* isteği ile seçilen bir coğrafi nesnenin öznitelik verisine bakmak için harita katmanını kullanır[12].

2.4 GEOSERVER

GeoServer, farklı dosya biçimlerinden ve veri tabanlarından gelen verilere dayanarak Geo Web Hizmetlerini sağlamak için kullanılan Java tabanlı bir uygulamadır[13]. Başka bir ifadeyle GeoServer, kullanıcıların mekansal verileri minimum çabayla yayınlamasını ve paylaşmasını sağlayan bir harita sunucusudur. GeoServer ile geliştiriciler hızlı bir şekilde harita servisleri kurabilir ve böylece tek bir kod satırı olmadan başkalarının verilerini web sitelerine ve uygulamalarına eklemelerine olanak sağlar. WFS, WMS vb. gibi OGC standartlarına da uygundur. Bu coğrafi nesneler GeoServer'a harita oluşturma ve veri paylaşımında büyük esneklik sağlar[14]. GeoServer PostgreSQL veri tabanlarından gelen vektör verileri için yerleşik bir destekle birlikte gelir[13]. Shapefiles, MapInfo, MySql ve Oracle'ın yanı sıra diğer formatları ve yüzlerce projeksiyon türünü içeren çoklu giriş ve çıkış formatlarını destekler[15]. Dahası, GeoServer ücretsiz ve güçlüdür, bu nedenle ticari harita yazılımları yerine artan bir şekilde kullanılmaya başlanmıştır[14].

2.5 JAVA

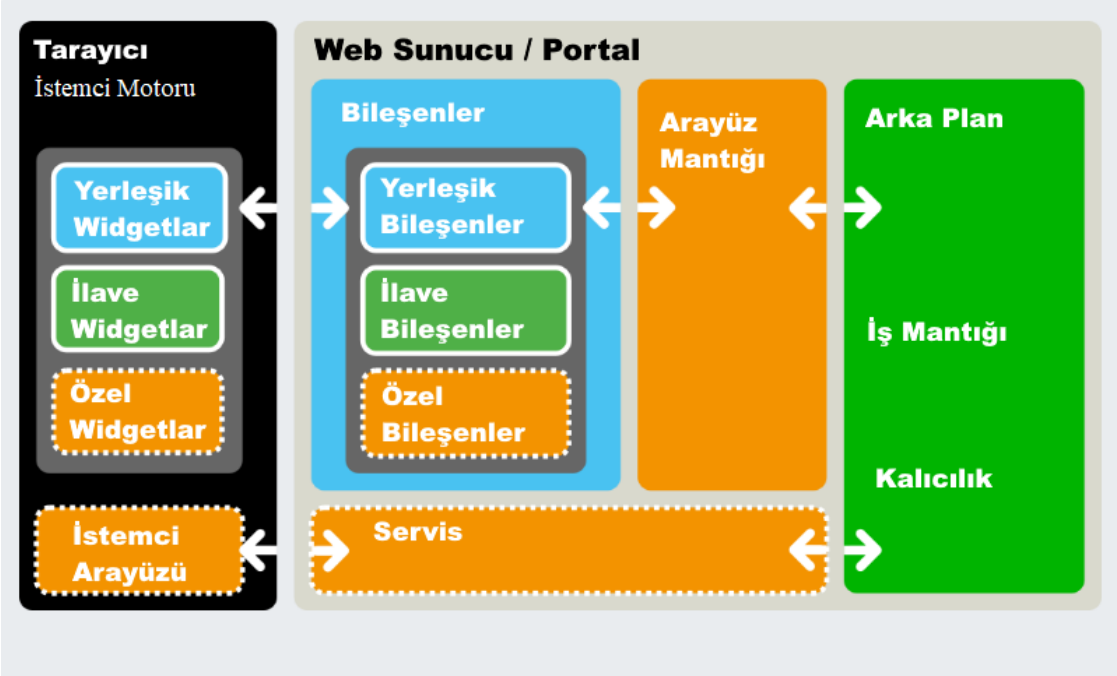
Java en popüler programlama dillerinden biridir. Java dilinde yazılmış programlar bir Java sanal makinesi tarafından yorumlanan Java bayt koduna çevrilir. Bu yaklaşım, dile, bellek sızıntısı, arabellek taşması ve benzeri türdeki kusurları azaltan veya ortadan kaldıran otomatik bellek yönetimi gibi bir takım elverişli özelliği eklememize olanak sağlar. Ancak, derlenmiş dillerle yazılmış olan iyi tasarımlı bir analog programa kıyasla uygulama performansını önemli ölçüde azaltabilir[16].

2.6 APACHE TOMCAT

Apache Tomcat, Apache Yazılım Vakfı (ASF) tarafından geliştirilen açık kaynaklı bir web sunucusu ve servlet sağlayıcısıdır. Tomcat, Java Servlet, Java Sunucu Sayfaları (JSP), Java EL, Web Soket ve içinde Java kodu çalışabilmesi için saf Java HTTP web sunucu ortamı da dahil olmak üzere birçok JAVA EE spesifikasyonunu sağlar[17].

2.7 VAADIN PLATFORMU

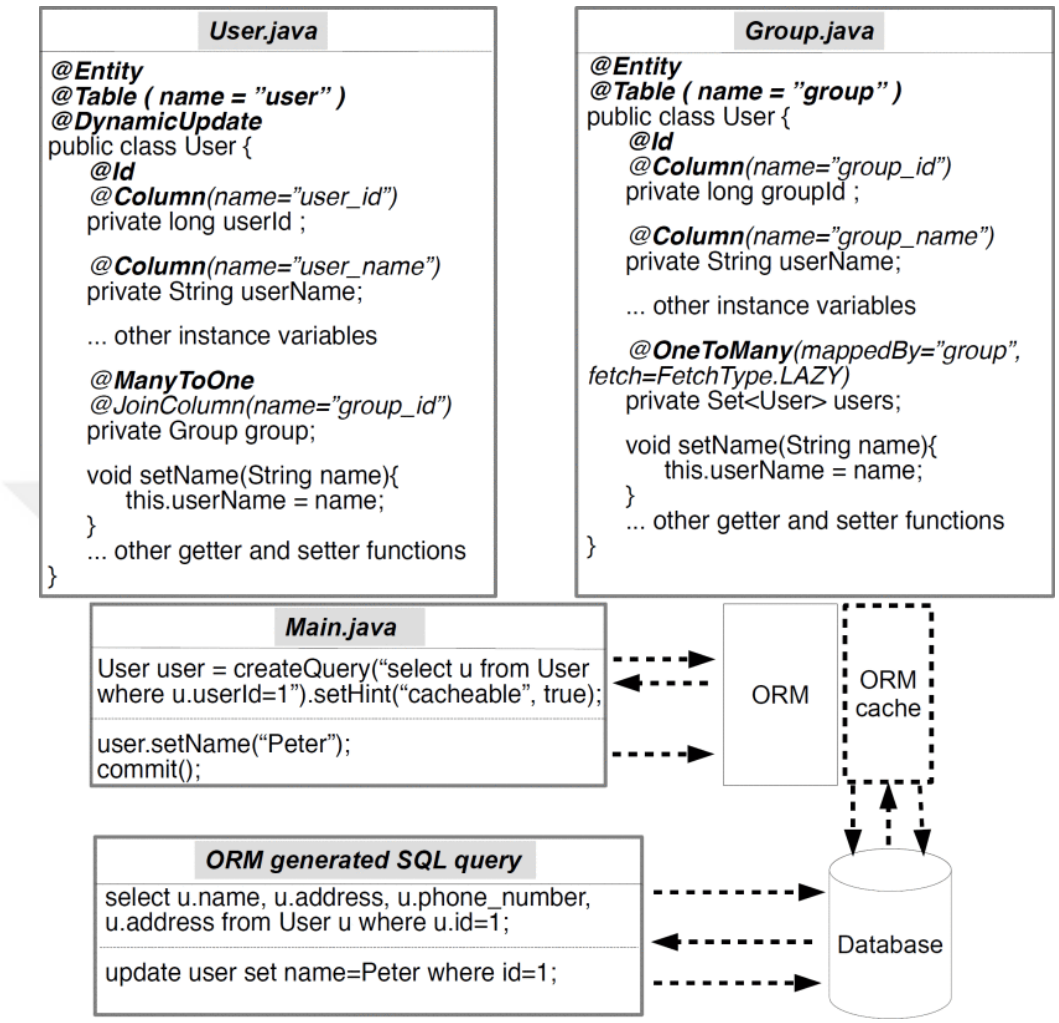
Vaadin platformu, yüksek kaliteli web tabanlı kullanıcı arayüzlerinin oluşturulmasını ve bakımını kolaylaştırmak için tasarlanmış bir Java web uygulama geliştirme platformudur. Bir masa üstü uygulaması geliştirir gibi web uygulaması geliştirmenize olanak verir[18]. Web uygulamalarının geliştirilmesi için bir API, görünümü kontrol etmek için birçok kullanıcı arayüzü bileşeni ve teması, bileşenleri doğrudan verilerle sınırlandırmanıza izin veren veri modelleri içerir. Bunun arkasında, tarayıcıdan istek almak ve yanıtları karşılık geldiği sayfalarda dağıtmak için bir "Terminal Bağdaştırıcısı" kullanır. Şekil 2.2'de görüldüğü gibi Vaadin kullanan bir uygulama HTTP isteklerine yanıt vermek için bir Java web sunucusunda servlet gibi çalışır. "Terminal Bağdaştırıcısı", sunucu uygulamasındaki servlet API'si aracılığıyla istemciden istekleri alır ve bunları çerezlerde depolanan belirli bir oturum için kullanıcı olayları olarak yorumlar. Olaylar, arayüzün bileşenleri ile ilişkilendirilir ve dinleyiciler aracılığıyla yönetilecek uygulamaya teslim edilir[19].



Şekil 2.2: Vaadin uygulama mimarisi[18]

2.8 ORM PLATFORMU

Veri tabanları, modern yazılım sistemlerinde en önemli bileşenlerden biri haline gelmiştir. Örneğin, web servisleri, bulut bilişim sistemleri ve çevrimiçi işlem işleme sistemlerinin tümü büyük ölçüde veri tabanlarına dayanır. Bu veri tabanlarına erişimin karmaşıklığını soyutlamak için, geliştiriciler ORM platformlarını kullanırlar. ORM platformları, uygulama mantığı ve arka plandaki veri tabanı arasında bir soyutlama katmanı sağlar. Bu soyutlama katmanı, nesne yönelimli dillerdeki nesneleri otomatik olarak veri tabanı kayıtlarına eşler ve bu da yazılması gereken kodun miktarını önemli ölçüde azaltır[20]. Diğer bir ifade ile ORM, veri çoğaltma, bakım maliyeti ve hata duyarlılığını ortadan kaldırarak nesne yönelimli ve ilişkisel dünya arasındaki köprüyü otomatikleştirmeyi amaçlayan bir teknolojidir[21]. Şekil 2.3'de veritabanı varlık sınıflarını ve ORM'nin nesne manipülasyonunu SQL sorgularına nasıl dönüştürdüğünün örneği gösterilmiştir.

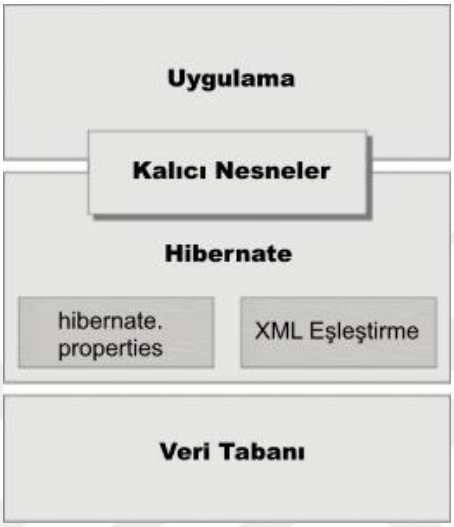


Şekil 2.3: Veritabanı varlık sınıfları ve ORM'nin nesne manipülasyonunu SQL sorgularına nasıl dönüştürdüğünün örneği[20]

2.8.1 Hibernate

Hibernate, hemen hemen tüm Java programcıları tarafından kullanılan popüler bir Java ORM platformudur. Hibernate, geliştiriciyi genel veri işlemleri ile ilgili programlama görevlerinin yüzde 95'inden kurtarmaya çalışır. Geliştirici Hibernate'i kullanarak, nesne sınıfı ve tablo arasındaki ilişkiyi kolayca yönetebilir. Hibernate, yalnızca Java sınıflarından veri tabanı tablolarına (ve Java veri türlerinden SQL veri türlerine) eşleme işlemine özen göstermekle kalmaz, aynı zamanda veri sorgulama ve geri alma olanakları sağlar ve SQL ve JDBC'de manuel veri işleme ile harcanan geliştirme süresini önemli ölçüde azaltabilir. Hibernate mimarisi Şekil 2.4'deki gibi tanımlanabilir[22]. hibernate.cfg.xml ve XML eşleştirmeleri gibi bazı yapılandırma

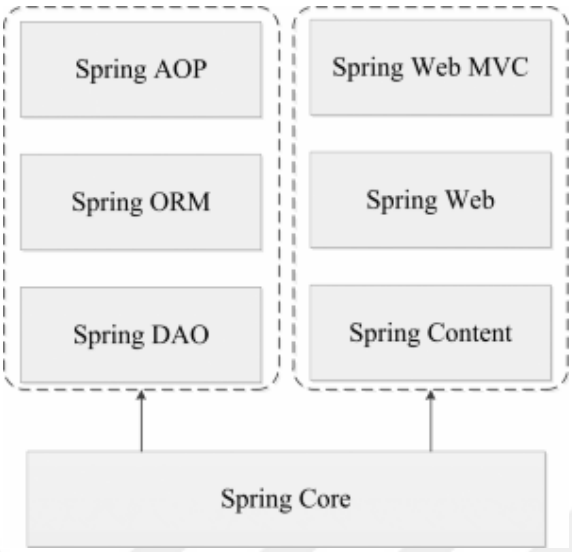
dosyalarını kullanarak veri tabanında uygulama programları için veri tutma hizmeti de sağlar. Böylece, programcılar, birbirleri ile ilişkili işlemleri gerçekleştirmek için veri tabanı arabirimlerini normal nesneler olarak çağırabilir, böylece hizmet mantığına ağırlık verebilirler[23].



Şekil 2.4: Hibernate mimarisi[22]

2.9 SPRING PLATFORMU

Spring, Hibernate gibi diğer birçok platforma destek sağlayan hafif ve güçlü bir Java uygulama platformudur[24]. İlk olarak Rod Johnson tarafından yazılmış ve 2003 yılında Apache Yazılım Vakfı tarafından piyasaya sürülmüştür[25]. Spring platformunun temel avantajlarından biri, yedi iyi tanımlanmış modülden oluşan katmanlı mimarisidir. Spring, beanlerin oluşturulması, konfigürasyonu ve yönetimi gibi çeşitli fonksiyonlara sahip bir çekirdek konteyner üzerine inşa edilmiştir. Spring platformu fonksiyon modülleri Şekil 2.5'de verilmiştir[26].



Şekil 2.5: Spring platformu fonksiyon modülleri[26]

Spring hafiftir, yükü ihmal edilebilir bir işlemci kullanımına sahiptir ve Java Bean'lerini kullanır. Spring tabanlı uygulamalar gevşek bir şekilde bağlanmıştır, bağımlılıkları yapılandırma dosyalarında listelenir ve platform tarafından enjekte edilir[27]. JDBC, Hibernate, JPA için önceden tanımlanmış taslaklar sunar. Uygulamanın test edilmesi kolaylaşır çünkü uygulamayı çalıştırmak için sunucuya ihtiyaç duyulmaz. Önbelleğe alma mekanizması, işlem yönetimi, doğrulama ve biçimlendirme için tanımlanabilir destek sağlar[24].

2.10 MAVEN

Maven bir yazılım proje yönetim aracıdır, projenin yapısını, raporlamasını ve dokümantasyon tabanlı proje nesnesi modelini (POM) yönetir. POM, Maven'in özüdür ve genel bilgilerini, derleme ayarlarını, derleme ortamını ve projenin POM ilişkilerini içeren bir XML dosyasıdır. Kullanımının kolay oluşu, platformdan bağımsız olması, projelerin otomatik test ve otomatik dağıtım ihtiyaçlarını karşılaması Maven'in başlıca özellikleridir. Yaşam döngüsü boyunca, yazılım oluşturma, test etme, sürüm çıkarma işlemlerinin tümü, manuel işlemi büyük ölçüde azaltan Maven tarafından yönetilir ve hata olasılığını önler[28]. Maven ile bir proje derlendiğinde, her biri öncekine bağlı olarak altı derleme fazı gerçekleştirilir. Bu fazlar varsayılan Maven yaşam döngüsünü oluşturur. Validate, projeyi oluşturmak için gereken tüm ayarların geçerli bir şekilde belirtilip belirtilmediğini kontrol eder. Compile, projenin kaynak kodunu ham dağıtımlara (örneğin, sınıf dosyalarına) çevirir. Test, hataları kontrol etmek için test

süitlerini derler ve yürütür. Packaging, işlenmemiş çıktıları hedef çıktı biçiminde (örneğin, jar, war, ear) birleştirir. Install, çıktıları lokal sistemdeki hedef konumlarına yerleştirir. Deploy, lokal kurulumu üretim ortamlarına iletir[29].

2.11 POSTGRESQL

PostgreSQL, yetenekli, açık kaynaklı bir veri tabanı sistemidir. Dinamik gelişimi, güvenilirliği, bilgi doğruluğu ve sağlamlığı ile ün kazanmış bir tasarıma sahiptir.

Tamamen gönüllülerden oluşan bir grup tarafından oluşturulan açık kaynaklı bir veri tabanı yönetim sistemidir (VTYS). PostgreSQL, herhangi bir şirket veya başka bir özel kuruluş tarafından kontrol edilmez.

Başlangıçta Postgres adı verilen PostgreSQL, UCB'de Michael Stonebraker adlı bir yazılım mühendisliği öğretmeni tarafından üretilmiştir. Stonebraker Postgres'e, Computer Associates tarafından sahiplenilen Ingres'in devamı olarak başlamıştır[30].

2.12 POSTGIS

PostgreSQL ile onun mekânsal eklentisi olan PostGIS, coğrafi verileri depolamak ve hesaplamak için yaygın olarak kullanılan açık kaynaklı ilişkisel veri tabanıdır. Coğrafi sorguların hızını arttırmak veya mekânsal işlemler yapmak için PostGIS, GiST endeksini (Genelleştirilmiş Arama Ağacı) kullanır. GiST ağaç yapılı bir indekstir, rastgele indeksleme şemalarını uygulamak için bir şablon gibi davranır. Coğrafi veriler için bu indeksleme şemalarından biri R-tree'ye dayanmaktadır. Tüm geometriler için sınırlayıcı kutular oluşturur ve sınırlayıcı kutular dizine bağlanır. Bu iyi bilinen kavramın, ilk önce geometrinin sınırlayıcı kutusunun R-tree indeksi kullanılarak belirlendiği ve ardından geometrinin yüklendiği sorgular içinde kullanıldığı anlamına gelir. Bu yaklaşım performansı büyük ölçüde artırır. Ayrıca PostGIS 3000'den fazla mekânsal referans sistemiyle çalışabilir[13].

2.13 OPENLAYERS

OpenLayers, harita verilerini web tarayıcılarında görüntülemek için coğrafi nesneler sağlayan açık kaynaklı bir JavaScript kütüphanesidir. Ayrıca, web tabanlı coğrafi uygulamalar oluşturmak için bir API sağlar. Ayrıca API, haritalar, katmanlar veya kontroller gibi muazzam bir bileşen

kümesi sunar. OpenLayers, birçok farklı veri formatını kullanarak çok sayıda veri kaynağına erişim sunar ve OGC'nin birçok standardını uygular[31]. OpenLayers'ın en büyük avantajı, diğer açık kaynaklı projelerde olduğu gibi, kullanımı tamamen ücretsiz olmasıdır. Proje ayrıca JavaScript tabanlı olduğu için standart web programlama pratikleri ile Google ve Bing haritalarının temiz kod avantajlarını OpenLayers kullanarak entegre edilmesini sağlar. OpenLayers da sürekli olarak geliştirilmektedir ve bir kuruluşun sahip olabileceği mevcut GIS verilerini görüntülemek için kullanılabilir[32].

2.14 OPEN STREET MAP (OSM)

Open Street Map (OSM), vatandaşlardan vatandaşlara ücretsiz coğrafi veriler sağlayan tanınmış bir uluslararası iş birliği projesidir. OSM projesi 2004 yılında başlatılmış olup, asıl amacı herkese ücretsiz coğrafi veri oluşturmak ve sağlamaktır. OSM karayolları, demiryolları, binalar, arazi kullanımı ve dünyadaki diğer birçok bilgi türüne katkıda bulunan ve bakımını yapan gönüllüler tarafından geliştirilmiştir[33]. OSM, OSM projesini destekleyen uluslararası ve kâr amacı gütmeyen bir kuruluş olan OSM vakfı tarafından işletilmektedir. OSM'nin coğrafi verileri, GPS, dijital fotoğraflar veya uydu verileri halinde gönüllüler tarafından toplanarak işlenir ve ardından OSM platformu üzerinden erişilebilir hale getirilmiş olur. Gönüllüler, verilerini açıklamak için gerekirse kendi isteğe bağlı etiketlerini ekleme özgürlüğüne sahiptir. Bu nedenle, OSM topluluğu OSM veri tabanındaki mekânsal nesneler için varlıklar üretebilir. OSM'nin en büyük yararı, yalnızca herhangi bir zaman dilimi için tüm veri tabanı erişimine sahip olmak değil, aynı zamanda, başkalarının yanlış veya hatalı katkılarını düzeltmeye niyetli bir topluluk ağı oluşturarak verinin kalitesini arttırmayı sağlamaktır. Böylece sistematik bir yöntemle ayarlanan verilerin tüm kalitesini iyileştirir[34].

2.15 KATMANLI MİMARİ

Bir yazılım sistemi için uygun bir mimari çok önemlidir. Çok katmanlı mimari, “böl ve yönet” stratejisini temsil eden ve modülerliği, yeniden kullanılabilirliği, ölçeklenebilirliği, sadeliği ve sürdürülebilirliği etkin bir şekilde artırabilen klasik, yaygın kullanılan bir mimari kalıptır[35].

Katmanlı mimari stili, hiyerarşik olarak düzenlenebilen benzersiz sınıfları ve servisleri içeren uygulamalar için en uygun seçimdir. Her katman işlevsel olarak diğerleriyle ortak bir rol veya

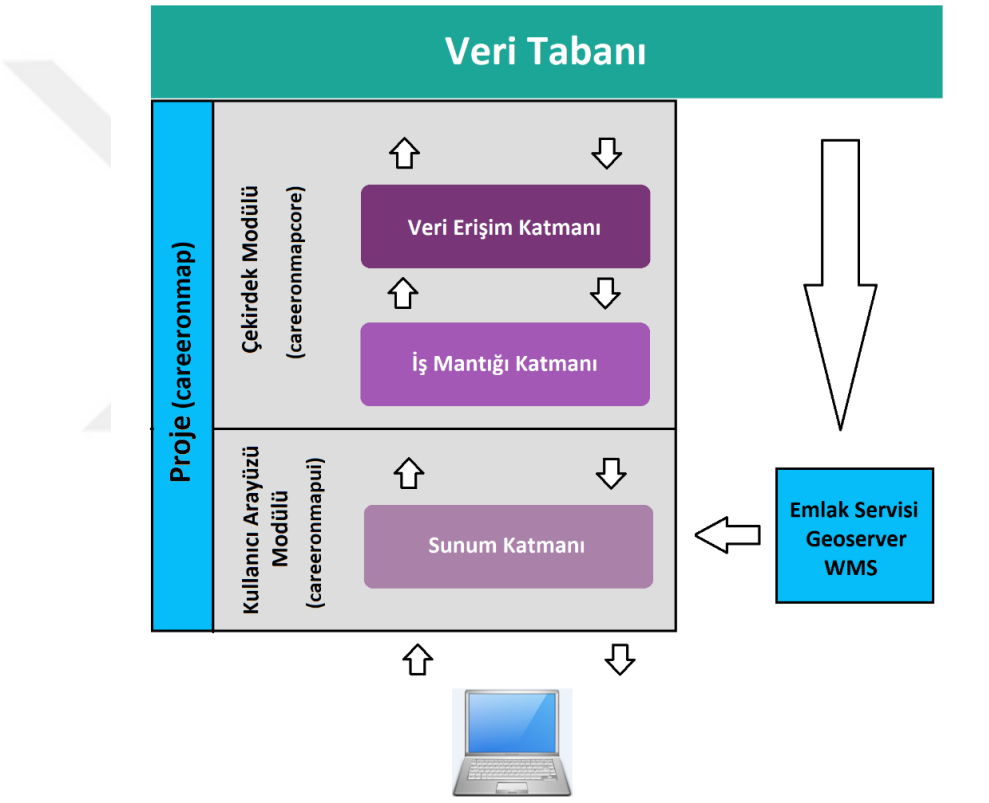
sorumluluk ile ilişkilidir. Katmanlar arasındaki iletişim açık ve gevşek bir şekilde birleştirilmiştir. Katmanlı mimari stil, her katmanın doğrudan altındaki katın sorumluluklarını ve soyutlamalarını topladığı ters bir piramit gibidir. Katmanlı mimari stilin tasarım ilkeleri soyutlama, yeniden kullanılabilirlik, açıkça tanımlanmış fonksiyonel katmanlar, gevşek bağlanma, yüksek bağlılık ve kapsüllemedir. Katmanlı uygulamaların örnekleri arasında web tabanlı uygulamalar, web siteleri ve dağıtılmış sistemler bulunur. Katmanlı mimari stil, karmaşık ve yapılandırılabilir sistem problemleri için de uygundur[36].

3. METOT

Bu bölümde, teze konu olan model projenin geliştirilmesinde izlenen adımlar açıklanmıştır.

3.1 PROJENİN GENEL YAPISI

Proje modüler ve katmanlı mimariye uygun olarak tasarlanmış olup genel mimari yapı Şekil 3.1'de gösterildiği gibidir.



Şekil 3.1: Projenin genel yapısı

Proje, iş ve eleman ilanlarına ek olarak emlak ilanlarını da kullanıcıların görüntülemesini amaçlar. Söz konusu emlak ilanlarının dış kaynaktan temin edilmesi planlanmıştır. Dış kaynağın örneklenebilmesi için emlak ilanlarını yayınlayan ek bir uygulama projeye dâhil edilmiştir. Bu uygulama da proje ile aynı veri tabanını kullanmaktadır.

3.1.1 Veri Tabanı

Projede coğrafi verileri depolamak ve coğrafi hesaplamalar yapabilmek için yaygın olarak kullanılan PostgreSQL ve onun mekânsal eklentisi olan PostGIS tercih edilmiştir. PostGIS ile mekânsal veriler depolanabilmekte ve farklı şekillerde sorgulanabilmektedir. Örneğin verilen bir kapalı alan sınırları içerisinde kalan iş ilanları bu eklenti ile mekansal olarak veri tabanından sorgulanabilir.

PostGIS eklentisinin aktif olabilmesi için Şekil 3.2'de belirtilen SQL komutu çalıştırılmalıdır. Komut çalıştırıldıktan sonra şema içerisinde PostGIS eklentisine özgü tablo ve view gibi veri tabanı nesneleri oluşur. Bu nesneler EK-A olarak eklerde gösterilmiştir.

```
CREATE EXTENSION postgis;
```

Şekil 3.2: PostGIS eklentisini aktif etmek için çalıştırılacak SQL komutu

PostgreSQL veri tabanındaki tablolar Hibernate platformunun desteği ile otomatik olarak üretilir. Java sınıfları olarak oluşturulan Entity modelleri, Java sınıfları içerisindeki anotasyonlara göre tabloları ve içerisindeki alanları otomatik olarak oluşturur. Bu tabloların otomatik olarak oluşabilmesi için Hibernate platformunun yapılandırma ayarlarında bulunan *hibernate.hbm2ddl.auto* parametresinin değeri *update* olarak tanımlanmalıdır.

Dış kaynaktan temin edilmesi planlanan emlak ilanlarının örneklemine ait veriler de bu veri tabanından okunarak *GeoServer* harita sunucu uygulaması aracılığı ile yayınlanır. Yayınlanan bu emlak verilerinin okunması ve görüntülenmesi için Şekil 3.1'de görüldüğü üzere WMS servisleri aracılığı ile haberleşilir.

Veri tabanı diyagramı EK-A olarak eklerde gösterilmiştir.

3.1.2 Projenin Oluşturulması (Proje:careeronmap)

Bu aşamada *Maven* komutları ile projenin genel iskeleti oluşturulur. *Maven* komutlarını çalıştırabilmek için her işletim sisteminde farklı isimlerle adlandırılan, Windows işletim sistemindeki adı "Komut İstemi" olan uygulama açılır. Projenin oluşturulmak istendiği dizin Şekil 3.3'de ifade edildiği gibi sırayla çalıştırılarak oluşturulur ve dizine erişim sağlanır.

```
mkdir <ProjeDizini>
cd <ProjeDizini>
```

Şekil 3.3: Dizin oluşturma ve dizine erişim komutları

Şekil 3.4'de belirtilen maven komutu çalıştırılır. Komut, *careeronmap* adında maven mimarisinde basit bir proje oluşturur.

```
mvn archetype:generate -DarchetypeGroupId=org.apache.maven.archetypes -DarchetypeArtifactId=maven-archetype-quickstart -DarchetypeVersion=1.1 -DgroupId=edu.altinbas -DartifactId=careeronmap -Dversion=1.0.0-SNAPSHOT
```

Şekil 3.4: Komut satırından projeyi oluşturma

Oluşan bu proje içerisindeki *pom.xml* yapılandırma dosyası haricindeki diğer dizinler ve dosyalar silinir. Amaç, ana projeye bağlı alt modülleri oluşturmaktır. Burada bırakılan *pom.xml* dosyası, alt modüllerde kullanılan kütüphanelerin sürüm bilgilerini tutmak ve genel yönetimsel ayarları yapabilmek için silinmez.

Bu *pom.xml* dosyasına "Parent POM" adı verilir. Ancak bu dosyanın Parent POM olmasını sağlamak için *pom.xml* dosyasında bir değer değişikliği yapılmalıdır. Şekil 3.5'deki *jar* değeri *packaging* olarak değiştirilmelidir.

```
<packaging>jar</packaging>
```

Şekil 3.5: pom.xml dosyasındaki değiştirilecek bölüm

Komut satırından Şekil 3.6'da belirtilen komutlar çalıştırılarak yeni oluşturulan projenin bulunduğu dizine erişim sağlanır ve önce çekirdek modülü oluşturulur. Bu modül bir kütüphane gibi kullanılabilir olması için paketleme türü *jar* olarak oluşturulur.

```
cd <ProjeDizini>\careeronmap

mvn archetype:generate -DarchetypeGroupId=org.apache.maven.archetypes -DarchetypeArtifactId=maven-archetype-quickstart -DarchetypeVersion=1.1 -DgroupId=edu.altinbas -DartifactId=careeronmapcore -Dversion=1.0.0-SNAPSHOT -Dpackaging=jar
```

Şekil 3.6: Çekirdek modülünü oluşturma

Projede kullanıcı arayüzlerinin *Vaadin* platformu ile oluşturulması planlanmıştır. Bu nedenle çalıştırılacak komut *Vaadin* mimari tipine göre oluşturulur. Kullanıcı ile etkileşimi sağlayan bu modülün oluşturulması için Şekil 3.7'deki komut çalıştırılır. Bu modül *Apache Tomcat* uygulama sunucusunda yayınlanacaktır. Bu nedenle paketleme türü *war* olarak oluşturulur.

```
mvn archetype:generate -DarchetypeGroupId=com.vaadin -DarchetypeArtifactId=vaadin-archetype-application -DarchetypeVersion=7.5.10 -DgroupId=edu.altinbas -DartifactId=careeronmapui -Dversion=1.0.0-SNAPSHOT -Dpackaging=war
```

Şekil 3.7: Kullanıcı arayüzü modülünü oluşturma

3.1.2.1 Çekirdek modülü (Modül:careeronmapcore)

Proje, yönetimi daha kolay ve bakım maliyeti daha az olduğu için katmanlı mimariye uygun olarak tasarlanmıştır. Bu yapıya göre, veri tabanından veri okuma ve yazma işlemleri için Veri Erişim Katmanı ve verinin nasıl yorumlanacağı, hangi veriye kimin erişeceği gibi iş kurallarını yönetmek için İş Mantığı Katmanı bu modülde değerlendirilecektir.

Veri Erişim Katmanı:
Kaydetme, silme, güncelleme, veri sorgulama gibi veri tabanından veri okuma ve veri tabanına yazma işlemleri bu katmanın sorumluluğundadır.

Projede bu işlemleri daha az kod yazarak yapabilmek için *Hibernate* platformundan faydalanılmıştır. *Hibernate* e göre her bir entity sınıfı veri tabanındaki bir tabloya karşılık gelir. Entity sınıfı içerisindeki alanlar ise o tablodaki kolonları ifade eder. Her entity aynı zamanda bir

Java sınıfıdır. Böylelikle veri tabanındaki her bir tablo Java tarafında nesne olarak kullanılabilir. Şekil 3.8'de veri tabanında *IL* tablosuna karşılık gelen *Il* entity sınıfı gösterilmiştir.

```
@Entity
@Table(name = "IL")
public class Il extends BaseEntity {

    @SequenceGenerator(name = "generator", sequenceName = "IL_ID_SEQ")
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY, generator = "generator")
    @Column
    private Long id;

    @Size(max = 50)
    @Column(length = 50)
    private String ilAdi;

}
```

Şekil 3.8: Entity sınıfı örneği

Entity sınıf diyagramı EK-B olarak eklerde gösterilmiştir.

Proje içerisinde kimi zaman aynı iki entity nesnesi karşılaştırılmak istenebilir. Bu karşılaştırma işleminde entity nesnesinin tüm alanlarının karşılaştırılmasına gerek yoktur. Her bir entity için ortak olan karşılaştırma özelliği *BaseEntity* adında bir üst sınıfta *equals* ve *hashCode* metotlarının özelleştirilmesi ile yapılır. Bunu sağlamak için, tüm entity sınıflarında ortak olan *Id* bilgisinin alınabilmesi gerektiği için *getId* metodu abstract bir metot olarak tanımlanır ve *BaseEntity*'den türetilmiş tüm sınıflarda tanımlanması zorunlu kılınır.

Oluşturulan entity sınıflarını kullanarak veri tabanından okuma ve veri tabanına yazma işlemlerini yapabilmek için DAO (Veri Erişim Nesnesi) sınıfları oluşturulur. Her bir entity ile ilişkili ayrı bir DAO sınıfı tanımlanır. Bu DAO sınıfları, ortak olan kaydetme – güncelleme, silme, tablodaki tüm veriyi entity olarak okuma, tablodaki veriyi *Id* bilgisine göre okuma, tablodaki veriyi bir sorgulama ölçütüne göre okuma gibi metotlar içeren *BaseDao* sınıfından türetilir. *BaseDao* sınıfı Şekil 3.9'da gösterildiği gibidir.

```
@Transactional(rollbackFor = Exception.class)
public abstract class BaseDao<T> {

    @Autowired
    SessionFactory sessionFactory;

    private final Class<T> clazz;

    public BaseDao(Class<T> clazz) {
        this.clazz = clazz;
    }

    @SuppressWarnings("unchecked")
    public T save(T entity) {
        return (T) getCurrentSession().merge(entity);
    }

    @SuppressWarnings("unchecked")
    public void delete(T entity) {
        getCurrentSession().delete(entity);
    }

    @SuppressWarnings("unchecked")
    @Transactional(readOnly = true, rollbackFor = Exception.class)
    public List<T> findAll() {
        return getCurrentSession().createCriteria(clazz).list();
    }

    @SuppressWarnings("unchecked")
    @Transactional(readOnly = true, rollbackFor = Exception.class)
    public List<T> findAllByQueryFilterDto(QueryBuilder queryFilterDto) {
        if (QueryBuilder.class.isAssignableFrom(queryFilterDto.getClass()))
{
            Query query =  ((QueryBuilder)
queryFilterDto).buildQuery(getCurrentSession());
            return query.list();
        }
        return null;
    }

    public T findById(Serializable id) {
        return (T) getCurrentSession().get(clazz, id);
    }

    public Session getCurrentSession() {
        return sessionFactory.getCurrentSession();
    }
}
```

Şekil 3.9: BaseDao sınıfı

BaseDao sınıfında olmayan, entity özelinde bir veri tabanına erişim işlemi varsa bu işi yapan metod kendi DAO sınıfında tanımlanır. Örneğin Şekil 3.10'da gösterilen metot *IlDao* sınıfında tanımlanır.

```
public List<Ilce> findAllByIlId(Long ilId) {
   String hql = "Select ilce From Ilce ilce Where ilce.il.id = :ilId";
   Query query = getCurrentSession().createQuery(hql);
   query.setParameter("ilId", ilId);
   return query.list();
}
```

Şekil 3.10: Entity özelinde bir veri tabanına erişim işlemini gösteren örnek metot

İş Mantığı Katmanı:
Veri tabanından ham halde gelen veriyi geliştiricinin belirlediği kurallara göre işleyen veya kullanıcı arayüzlerinden gelen veriyi yine geliştiricinin belirlediği kurallara göre işleyip veri tabanına yazmak için sunum katmanı ile veri erişim katmanı arasındaki iletişimi sağlayan katmandır.

Bu katmanda bir transaction mekanizması bulunur. Transaction mekanizması birden fazla işlemin tek bir bütün halinde yapılmasını sağlar. Bütün olarak kullanmanın faydası herhangi bir parçada meydana gelen bir hatada diğerlerinin otomatik olarak geri alınmasını varsayılan olarak sağlıyor olmasıdır. Bu sebepten bu metotları birer birer kullanmak yerine transactionlar tercih edilir.

İş mantıkları Service sınıflarında oluşturulur. Her bir DAO sınıfına karşılık gelen bir Service sınıfı tanımlanır. Tüm Service sınıflarında ortak olan metotlar da *BaseService* sınıfında tanımlanır. DAO sınıflarında ortak olmayan metotlar ise kendisine karşılık gelen Service sınıfı içerisinde iş mantığı ile beraber oluşturulur. *BaseService* sınıfı Şekil 3.11'de gösterilmiştir.

```
public abstract class BaseService<E extends BaseEntity, D extends BaseDao<E>> {

    @Autowired
    private ApplicationContext applicationContext;

    @Autowired
    private D dao;

    private Class<D> daoClass;

    public BaseService(Class<D> clazz) {
        daoClass = clazz;
    }

    public E findById(long id) {
        return dao.findById(id);
    }

    public List<E> findAll() {
        return dao.findAll();
    }

    public E save(E e) {
        return dao.save(e);
    }

    public void delete(E e) {
        dao.delete(e);
    }

    public D getDao() {
        return dao;
    }
}
```

Şekil 3.11: BaseService sınıfı

Bu katmanda test senaryolarına göre test metotları da yazılır. Bunu yapmak için Hibernate platformu ve Spring platformunun yapılandırma ayarları yapılmalıdır.

Hibernate platformunda entity nesnesinin veri tabanındaki tablo ile olan ilişkisinin tanımı ve veri tabanına erişim bilgileri *hibernate.cfg.xml* dosyası içerisinde tanımlanır. Spring platformunun desteği ile bu tanımlar *applicationContext.xml* dosyası içerisinde yapılır ve *hibernate.cfg.xml* dosyası oluşturulmaz. Spring platformunun paket tarama özelliği sayesinde de tüm entity

nesneleri için ayrı ayrı ilişki tanımı yapmaya gerek kalmaz. Şekil 3.12'de *applicationContext.xml* dosyasındaki Hibernate ile ilgili bir bölüm gösterilmiştir. Ayrıca bu dosyada tabloların otomatik olarak oluşabilmesi için Hibernate platformunun yapılandırma ayarlarında bulunan *hibernate.hbm2ddl.auto* parametresinin değeri *update* olarak tanımlanmalıdır.

```
<!-- Hibernate session factory -->
<bean id="sessionFactory"
class="org.springframework.orm.hibernate4.LocalSessionFactoryBean">

   <!-- DataSource.xml PostgreSQL hakkında olan bilgiler eklendi. -->
   <property name="dataSource" ref="dataSource"/>

   <!-- Entity kümesini hangi pakette arayacağı eklendi. -->
   <property name="packagesToScan" value="edu.altinbas.careeronmap.core"/>

   <!-- Hibernate ile alakalı tüm özellikler eklendi. -->
   <property name="hibernateProperties">
      <props>
         <prop
key="hibernate.dialect">org.hibernate.spatial.dialect.postgis.PostgisDialect</prop>
         <prop key="hibernate.show_sql">true</prop>
         <prop key="hibernate.format_sql">true</prop>
         <prop key="hibernate.default_schema">careeronmap</prop>
         <prop key="hibernate.hbm2ddl.auto">update</prop>
      </props>
   </property>
</bean>
```

Şekil 3.12: applicationContext.xml dosyasından Hibernate ile ilgili bir bölüm

Tüm test sınıflarında ortak olan anotasyonlar *BaseServiceTest* sınıfı içerisinde tanımlanmıştır. Bu anotasyonlar, *applicationContext.xml* dosyasındaki konfigürasyon bilgilerine göre Spring platformunun çalışmasını, Hibernate platformunun veri tabanına erişmesini ve transactional yapının oluşmasını sağlar. Diğer tüm test servisleri de bu sınıftan türetilir. *BaseServiceTest* sınıfı Şekil 3.13'de gösterilmiştir.

```
@Transactional(rollbackFor = Exception.class)
@RunWith(SpringJUnit4ClassRunner.class)
@TransactionConfiguration(transactionManager = "transactionManager",
defaultRollback = false)
@ContextConfiguration("classpath:applicationContext.xml")
public class BaseServiceTest {
}
```

Şekil 3.13: BaseServiceTest sınıfı

3.1.2.2 Kullanıcı arayüzü modülü (Modül:careeronmapui)

Bu modülde veri tabanındaki veri kullanıcıya arayüz bileşenleri aracılığı ile gösterilir veya kullanıcıdan alınan veriler iş katmanına iletilerek işlenir ve veri erişim katmanı aracılığıyla veri tabanına yazılması sağlanır.

Katmanlı mimarideki, arayüzler ile iş mantığı katmanının etkileşimini sağlayan Sunum Katmanı bu modül içerisindedir.

Sunum Katmanı:
Projenin dış dünyaya açılan kullanıcı arayüzlerinin olduğu katmandır. Bu katmanın geliştirilmesinde Vaadin platformu tercih edilmiştir. Sadece Java kodu yazarak masa üstü bir uygulama geliştirir gibi web uygulaması geliştirilmesi planlanmıştır. Vaadin platformunun içerisinde hazır olarak gelen birçok arayüz bileşeni bulunur. Bu bileşenler arayüzlerin geliştirilmesinde hız ve kolaylık sağlar.

Vaadin platformu ile hazır gelen bileşenlerin farklı durumlarda farklı renklerde, boyutlarda veya farklı özelliklerde görünebilmesi için özelleştirilmiş bileşenler oluşturulur. Örneğin *null* verilerde metin bileşeninde *null* yazmaması için Vaadin platformunda var olan *TextField* bileşeninden türetilen bir *ComTextField* bileşeni oluşturulur ve içerisinde *null* veri geldiğinde boşluk görünmesi için *setNullRepresentation("")* metodu çağrılır. Ayrıca bu bileşenlerdeki stil özelleştirmeleri için de Vaadin platformunun kendi içerisindeki *ValoTheme* sınıfı, *ValoTheme* sınıfından türetilmiş *MaterialTheme* sınıfı veya *CareerOnMapTheme.scss* dosyasındaki geliştirici tanımlı stilleri çağıran *ComTheme* sınıfı kullanılır.

Projede, sistemi kullanabilen 4 farklı kullanıcı türü mevcuttur. Bunlardan 3 tanesi veri tabanına kaydedilir ve sisteme giriş yapar. Diğer kullanıcı türü ise veri tabanına kaydedilmez ve sistemi

kısıtlı olarak kullanır. Bu kullanıcı türleri ve yapabilecekleri işlemler aşağıda detaylı olarak verilmiştir.

a. Anonim Kullanıcı

Kullanıcı girişi yapmadan sistemi kullanabilen kullanıcılardır.

Anonim kullanıcı:

- İş İlanlarını Görüntüler
- Mekânsal Filtrelemeler Yapar
- Başvuru Yapamaz
- Mesafeye Göre Renklendirme Filtresini Kullanamaz. Çünkü bunu yapabilmek için kullanıcının mevcut konum bilgisine ihtiyaç vardır.

b. Yönetici Kullanıcı (Admin)

Kullanıcı girişi yaparak projedeki temel bilgileri arayüzler aracılığı ile sisteme ekleyip, çıkaran kullanıcılardır. Bu kullanıcılar veri tabanına Şekil 3.14'deki gibi manuel sorgu yazarak oluşturulur. Bundan sonraki bölümlerde Admin olarak isimlendirileceklerdir.

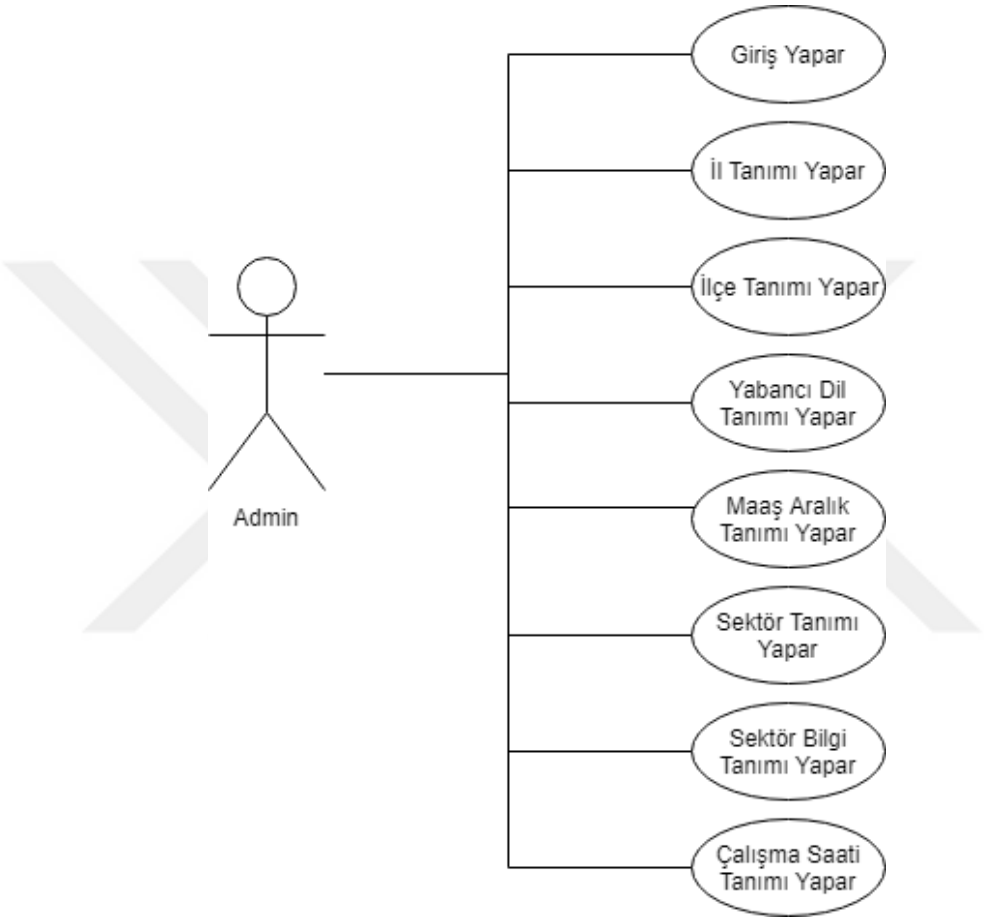
```
insert into careeronmap.kullanici
(ad, soyad, kullaniciadi, sifre, admin)
values
('Atilla Gökhan', 'KARTAL', 'agkartal', 'agk123', 'EVET');
```

Şekil 3.14: Admin türünde bir kullanıcıyı veri tabanına ekleme sorgusu

Admin kullanıcıları, *BaseTanimView*'den türetilmiş olan kaydetme – güncelleme, silme ve listeleme işlemlerinin yapılmasına olanak sağlayan sınıfların oluşturduğu arayüzleri kullanırlar. Bu sınıflar uygulamayı kullanan diğer kullanıcılar için veri üretmeyi sağlayan arayüzleri oluştururlar. Örneğin bir ilanın sektör bilgisinin ilan arayüzünde seçilebilmesi için Admin kullanıcısı tarafından "*Sektör Tanım*" arayüzünden tanımlanmış olması gerekir. Temelde hepsi aynı işlevleri barındırdığı için *BaseTanimView*'den türetilmişlerdir ve kendi içlerinde

özelleşmişlerdir. Tüm bu özelleşmiş arayüzler *AdminContainerView* arayüzü üzerinden Admin kullanıcılarına gösterilir.

Admin kullanıcıların kullanım durumu diyagramı Şekil 3.15'de gösterilmiştir.



Şekil 3.15: Admin kullanıcı türü için kullanım durum diyagramı

c. İşveren Kullanıcı

Kullanıcı girişi yaparak mekânsal iş ilanları yayınlayan veya İş Arayan kullanıcıları bulmak için mekânsal filtrelemeler yapan kullanıcılardır. Bu kullanıcılar sisteme üyelik arayüzleri üzerinden kayıt olurlar.

İşveren Kullanıcı:

- Üyelik Bilgilerini Ekler/Çıkarır

- İlan Yayınlar
- İlana Detay Ekler/Çıkarır
- İlana Başvuranları Görüntüler
- Mekânsal Filtrelemeler ile Eleman Arar

d. İş Arayan Kullanıcı

Kullanıcı girişi yaparak iş ilanlarını mekânsal olarak filtreleyen ve uygun gördükleri ilanlara başvuru yapan kullanıcılardır. Bu kullanıcılar da sisteme üyelik arayüzleri üzerinden kayıt olurlar.

İş Arayan Kullanıcı:

- Üyelik Bilgilerini Ekler/Çıkarır
- Aldığı Eğitim Bilgilerini Ekler/Çıkarır
- Yabancı Dil Bilgilerini Ekler/Çıkarır
- Deneyimlerini Ekler/Çıkarır
- Referanslarını Ekler/Çıkarır
- Mekânsal Filtrelemeler ile İlan Arar
- İş İlanlarına Başvuru Yapar

Projenin ana sayfasında arayüzün tamamını kaplayan bir harita bulunur. Projede mekânsal filtrelemelere vurgu yapıldığı için bu şekilde bir tasarım uygun görülmüştür.

Harita arayüzü ve bu harita arayüzü üzerinde yapılan mekânsal işlemler *OpenLayers* eklentisi ile yapılmaktadır. Arayüzde görüntülenen harita ise OSM haritasıdır.

Ana sayfanın sol tarafında açılır kapanır bir panel bulunur. Panelde, haritada görüntülenen iş ilanı veya eleman konumunu ifade eden noktaların renklerini farklı ölçütlere göre tekrardan düzenlemek için "*Renklendirme Türü*" seçim bileşeni bulunur. Renklendirme türü sektör olarak

seçilmişse, Admin kullanıcısının Sektör Tanım arayüzünde tanımlamış olduğu renkte görünür. Diğer bilgilerden biri seçilirse rastgele üretilen bir renk koduna göre görüntülenir. Haritanın üst kısmında üyelik oluşturma veya sisteme giriş yapılması için istemciyi yönlendirecek bileşenler bulunur. Sisteme giriş yapıldıktan sonra bu bileşenler değişir. Sisteme giriş yapan kullanıcının ad – soyad veya unvan bilgisi görüntülenir.

İş Arayan kullanıcılar, "*İş Arayan Üyelik*" ekranından bilgilerini girerek üyelik oluşturur. Her üye kendi ikamet adresine ait koordinat bilgisini harita arayüzünü kullanarak seçer. Üyelik oluştururken girdiği bilgilerin bazıları ana sayfadaki harita arayüzünde kullanılan ölçütlerdendir. Örneğin "*Maaş Aralık*" bilgisi harita ekranındaki "*Renklendirme Türü*" seçiminde İşveren kullanıcıları için bir görüntüleme ölçütüdür.

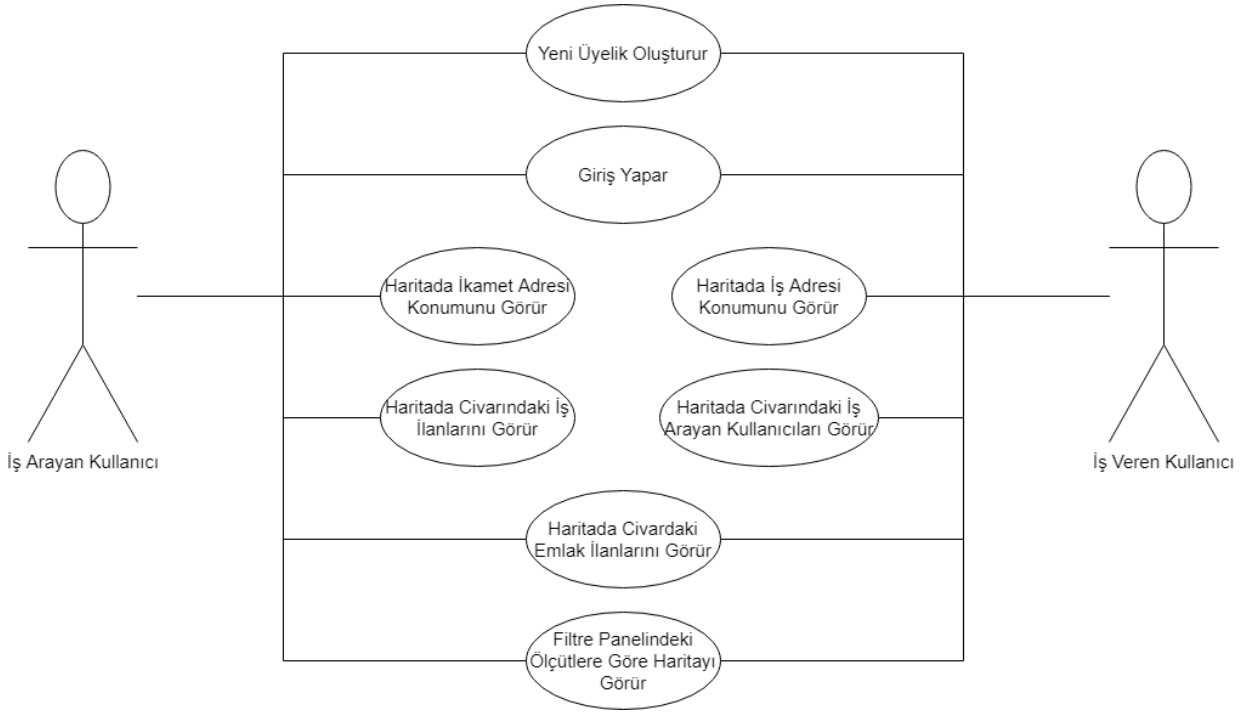
İş Arayan kullanıcı sisteme giriş yaptıktan sonra sol taraftaki paneli kullanarak kendi istediği ölçütlere göre görüntüleme yapabilir. İkamet adresine göre kendi istediği mesafeye göre oluşan alandaki ilanları görüntülemek isterse "*Mesafe*" alanına kilometre cinsinden yarıçap bilgisi girerek görüntüleme yapar. Bu işlem PostGIS eklentisinin veri tabanında coğrafi sorgulama yapması ile sağlanır. PostGIS, mesafeye göre oluşan alana göre içine düşen ilanları getiren bir sorgulama işlemi çalıştırır. İş Arayan kullanıcı dilerse kendisinin çizdiği bir alandaki iş ilanlarını görüntülemek isteyebilir. Bunun için ise "*Çiz*" bileşenini kullanır. Bu bileşen ile kullanıcı bir çokgen çizebilir ve bu çokgen içerisindeki iş ilanlarını görüntüleyebilir.

Beğenilen bir iş ilanı ikamet edilen yerden uzak yerlerde olabilir. İkamet edilen yer ile iş ilanının arasındaki mesafenin uzak olması kimi zaman kullanıcıyı ikamet adresini değiştirmeye itebilir. Bunun için, kullanıcı başvuru yapmak istediği iş ilanının çevresindeki emlak ilanlarını da görüntülemek isteyebilir. Civardaki emlak ilanlarının fiyatlarına göre bu ilan hakkındaki kararını gözden geçirebilir. Bu ilanlar, emlak ilanı yayınlayan diğer uygulamalardan istek yapılarak görüntülenir. Bu örneklemi sağlayabilmek ve emlak ilanlarını yayınlamak için GeoServer kullanılır. Bu veriler başka bir uygulama üzerinden geliyormuş gibi bir ortam oluşturulur. GeoServer bir Apache Tomcat üzerinden yayın yapar ve sadece emlak ilanlarını yayınlar.

İşveren kullanıcılar ise, "*İş Veren Üyelik*" ekranından bilgilerini girerek üyelik oluşturur. Her İşveren kullanıcı, firmanın konumuna ait koordinat bilgisini harita arayüzü üzerinde seçer. Üyelik oluşturulduktan sonra eğer bir ilan yayınlanacaksa ilan tanımı ayrıca yapılır ve her ilan

için ayrı bir konum bilgisi girilebilmesi için ayrı bir harita arayüzü kullanılır. İşveren kullanıcılar da tıpkı İş Arayan kullanıcılar gibi sol paneldeki ölçütleri kullanabilir.

İş Arayan ve İşveren kullanıcıların ana sayfa arayüzündeki yapabileceklerini gösteren durum diyagramı Şekil 3.16'deki gibidir.

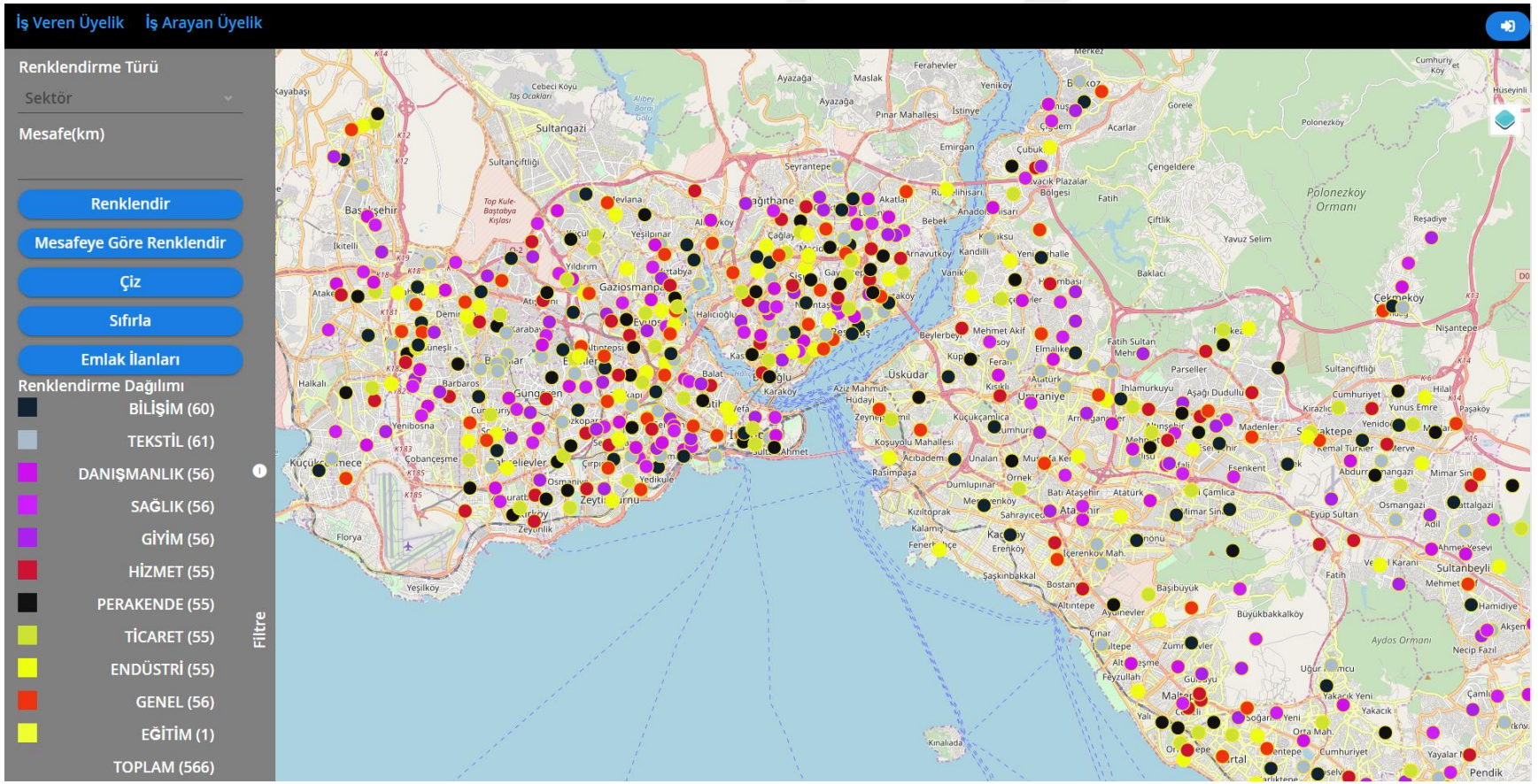


Şekil 3.16: İş Arayan ve İşveren kullanıcıların ana sayfa için kullanım durum diyagramı

4. UYGULAMA

Bu bölümde, proje arayüzlerinin nasıl kullanıldığı ile ilgili bilgiler, ekran görüntüleriyle anlatılarak detaylandırılmıştır.

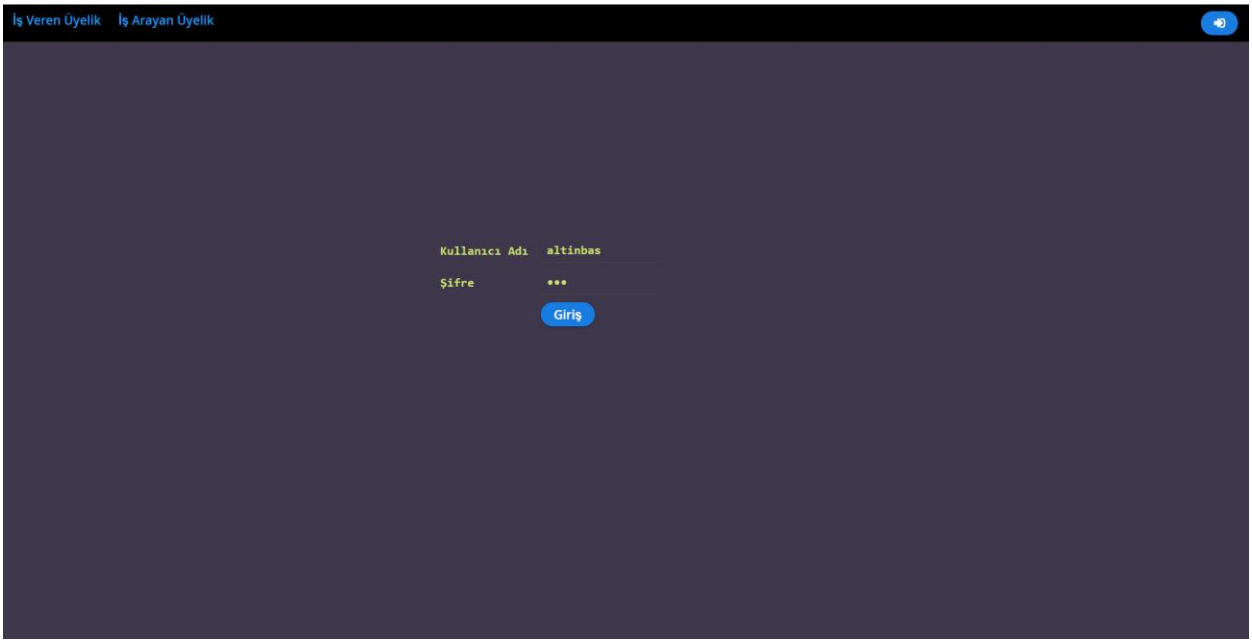
Kullanıcı sistemi web tarayıcıdan açtığında ilk olarak ana sayfa ekranını görüntüler. Bu esnada kullanıcı, anonim kullanıcıdır ve sistemdeki tüm iş ilanlarını sektöre göre renklendirilmiş olarak Şekil 4.1'de olduğu gibi görüntüler.



Şekil 4.1: Anonim kullanıcı için ana sayfa görünümü

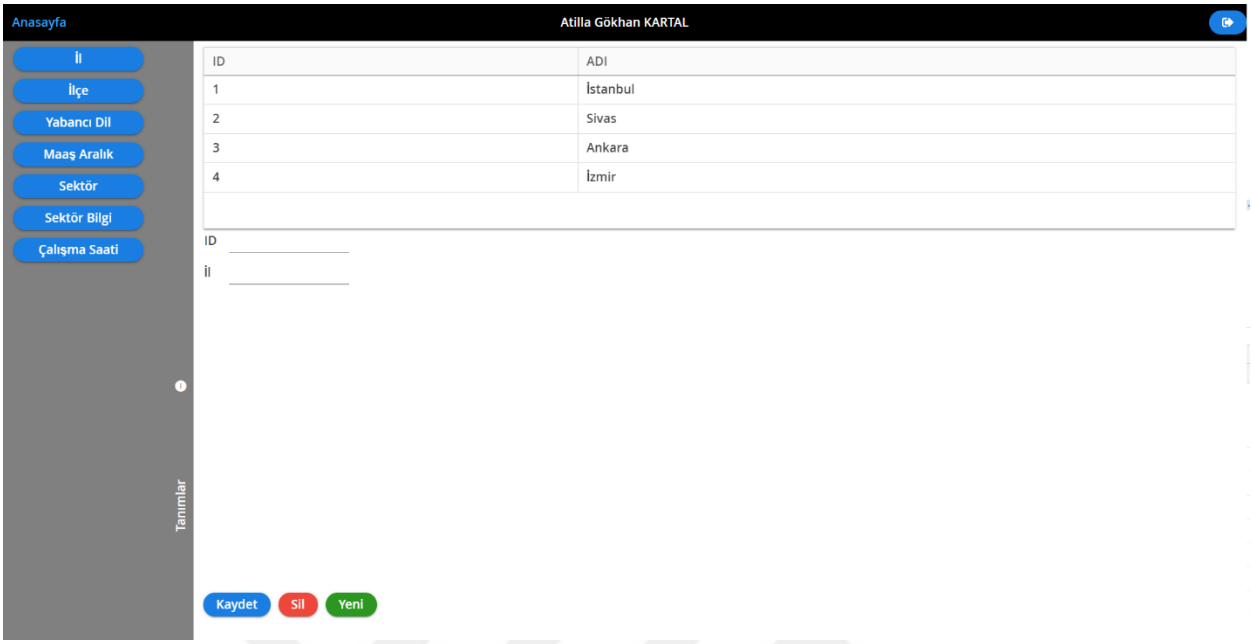
Sistemdeki iş ilanları veya iş arayan kullanıcıların konum bilgileri sol tarafta bulunan "Renklendirme Türüne" göre farklı renklerde görüntülenir. Renklendirme türü sektöre göre seçilirse, Admin kullanıcısının sektör tanımı yaparken belirlemiş olduğu renk koduna göre ilanlar haritada görüntülenir. Diğer renklendirme türleri seçilirse sistem otomatik renk üretir.

Kullanıcı Şekil 4.2'deki arayüzden sisteme giriş yaptıktan sonra kullanıcı türüne göre işlemlerine devam eder.



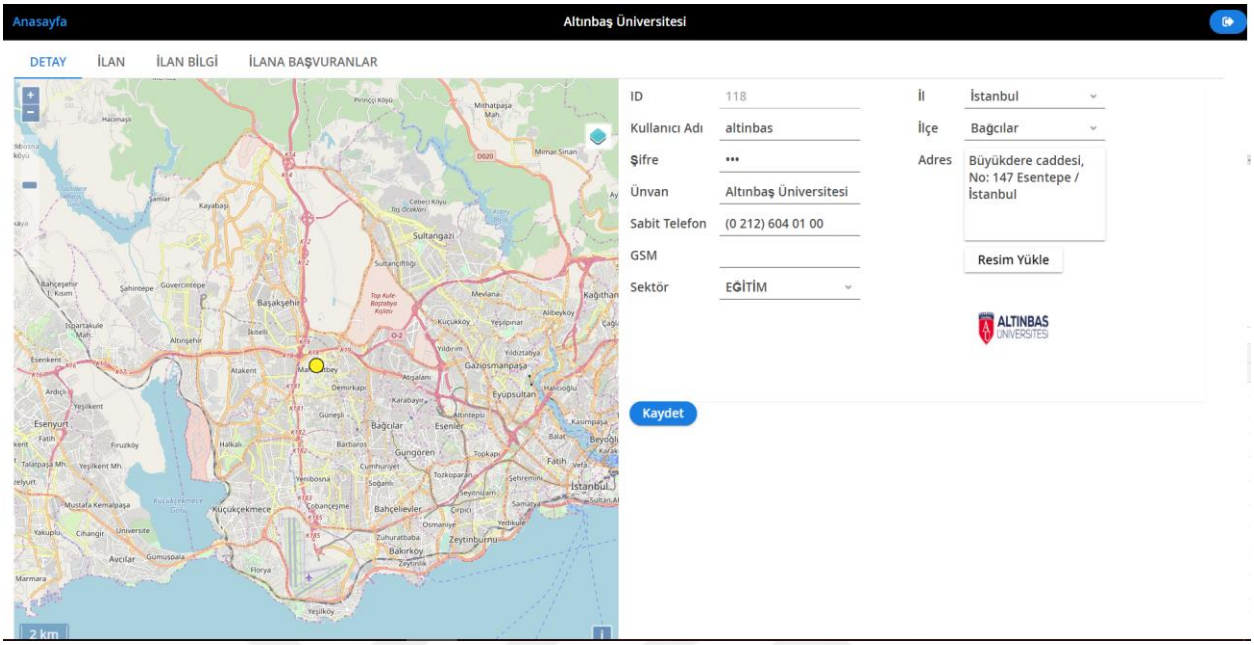
Şekil 4.2: Kullanıcı giriş ekranı

Kullanıcı Admin türünde ise sistemi kullanan kullanıcıların gereksinim duyduğu verileri Şekil 4.3'deki ekrandan sisteme ekler veya sistemden çıkarır. Kullandığı arayüzleri sadece Admin türündeki diğer kullanıcılar görebilir. Admin ana sayfaya giriş yaptığında ise İş Arayan kullanıcı gibi davranır.



Şekil 4.3: Sistemin gereksinim duyduğu verilerin tanımlandığı ekran

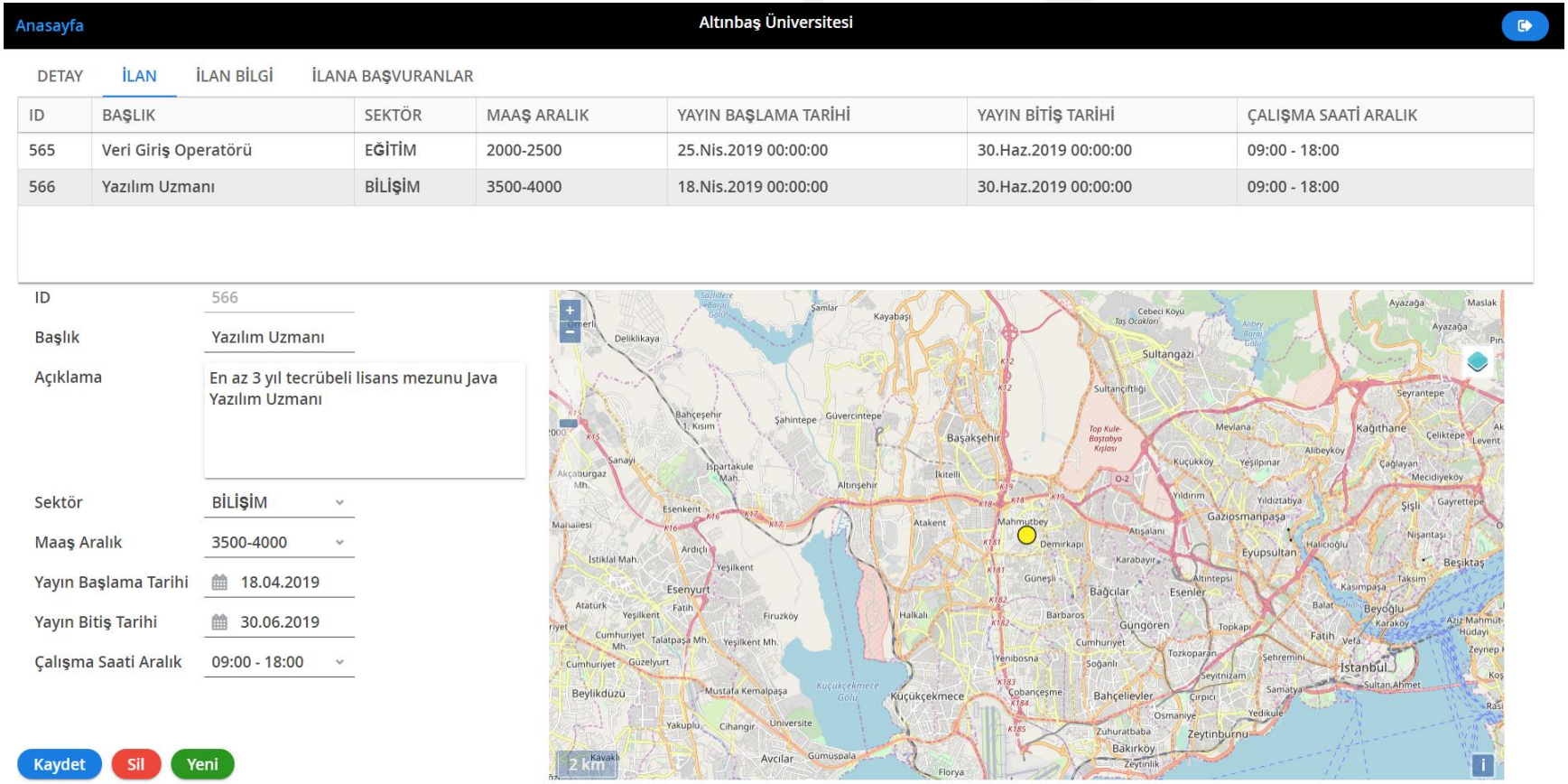
Sisteme giriş yapan kullanıcı İşveren türünde ise sadece kendi sektöründeki elemanları görüntüler. İşveren türündeki kullanıcılar sisteme Şekil 4.4'deki gibi bir üyelik arayüzünde kendisine özel bilgilerini girerek kayıt olurlar. Bu arayüz üzerinde kullanıcının konum bilgisini işaretlemesi için bir harita bulunur. Buradaki konum bilgisi İşveren kullanıcının merkez konumunu ifade eder.



Şekil 4.4: İşveren kullanıcı üyelik detay ekranı

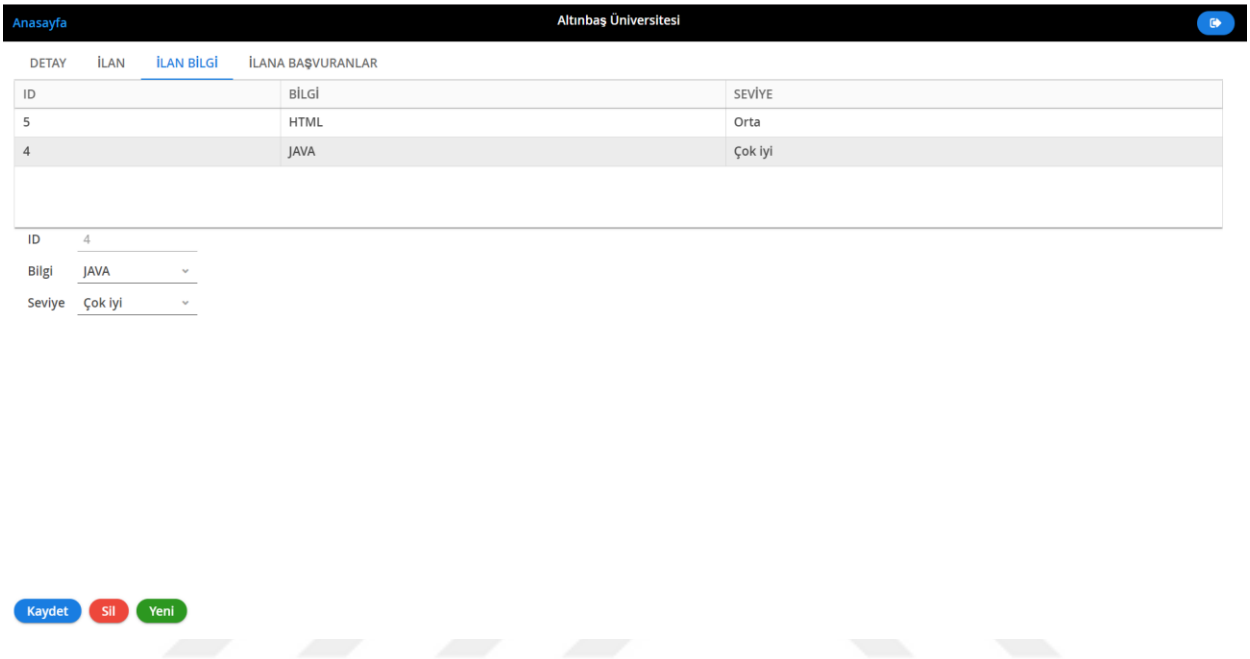
Kullanıcı kayıt işlemini tamamladıktan sonra ilanlarını, ilanlarına ait detaylı bilgilerini veya ilanlarına başvuran adaylarını görebileceği sekmeleri görüntüleyebilir.

İlan sekmesinde İşveren kullanıcı Şekil 4.5'de görüldüğü gibi yeni ilanlar yayınlayabilir. Bu ilanlar İş Arayan kullanıcılar tarafından görüntülenebilir. Bir İşveren kullanıcının birçok farklı konumda eleman istihdam etme ihtiyacı olabileceği için her ilanına farklı konum girmek isteyebilir. Bu nedenle ilan sekmesinde de harita bulunur. İlan bilgilerini giren kullanıcı konumunu da belirtir. Bu konum bilgisi İş Arayan kullanıcı için ve sistem için değerlidir. Çünkü ana sayfada bu konum bilgilerine göre filtrelemeler yapılır.



Şekil 4.5: İşveren kullanıcı ilan tanım ekranı

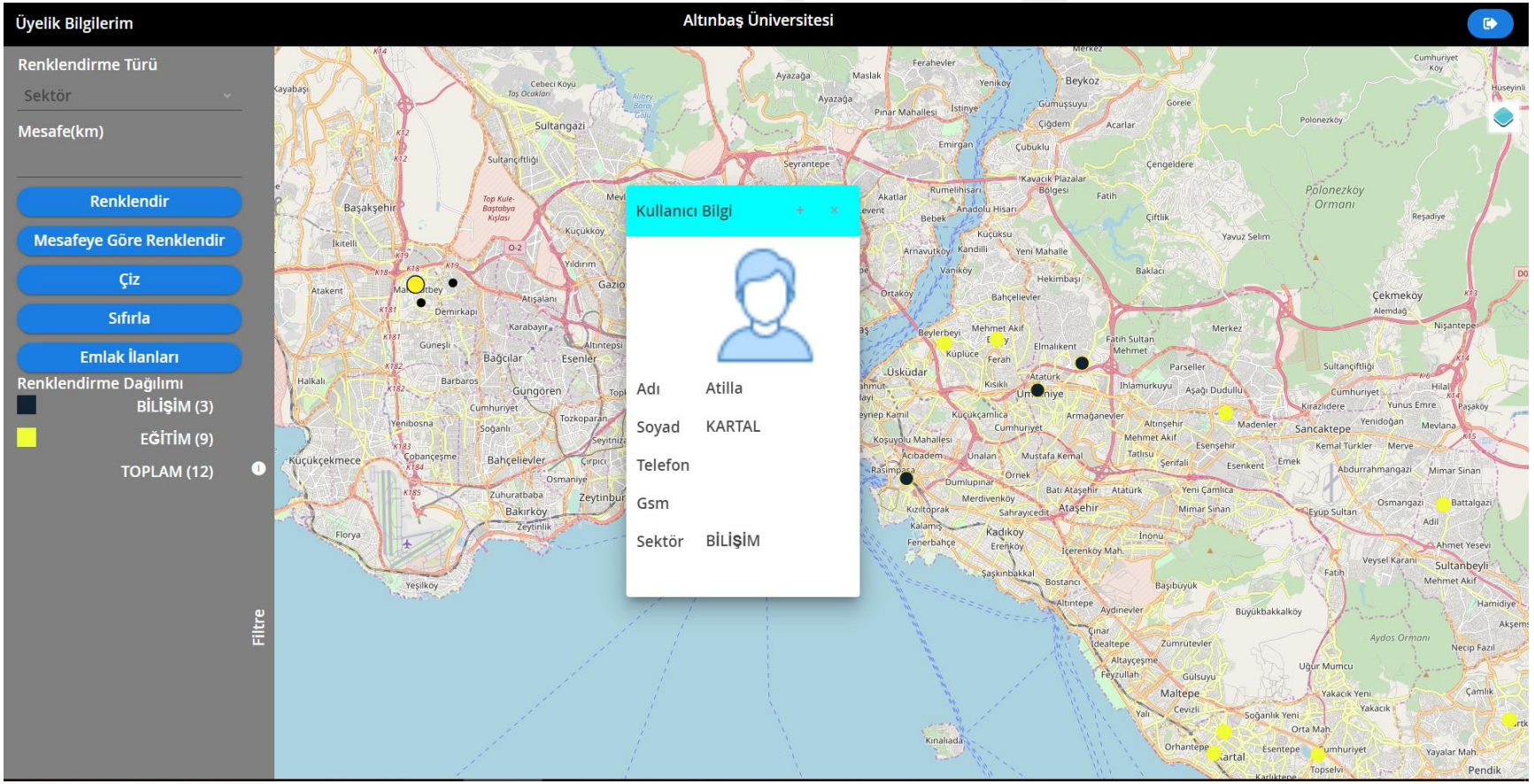
İlan bilgi sekmesinde, İşveren kullanıcının ilan sekmesinde girdiği ilana ait İş Arayan kullanıcıdan beklediği bilgi ve bu bilginin seviyesi Şekil 4.6'daki arayüzden tanımlanır. Son sekmede ise İşveren kullanıcı, yayınlamış olduğu bu ilana başvuran kullanıcıları görüntüler.



Şekil 4.6: İş ilanında, İş Arayan kullanıcıdan beklenen bilginin tanımlandığı ekran

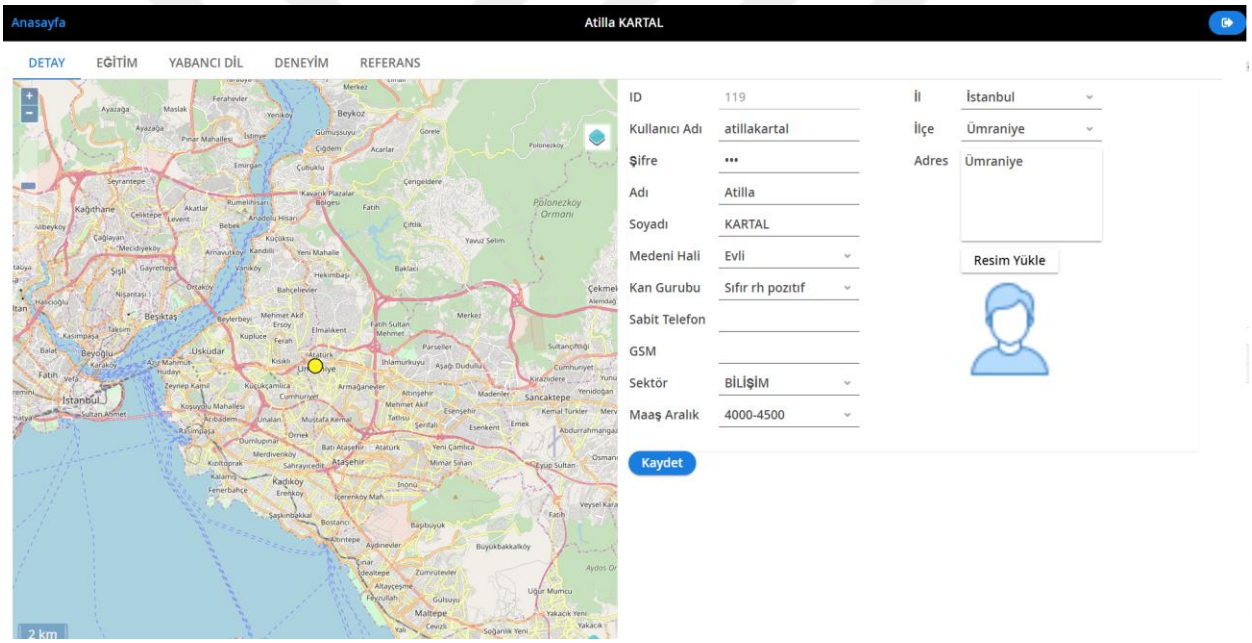
İşveren kullanıcı, ana sayfada firmanın merkez konumunu, yayınlamış olduğu aktif iş ilanlarının konumlarını ve iş ilanları ile aynı sektörde olan İş Arayan kullanıcıların konumlarını görüntüler. İşveren kullanıcının merkez konumu diğer noktalardan biraz daha büyük ve sarı ile görüntülenirken, yayınlanan iş ilanlarının konumları ise biraz daha küçük ve siyah renkli olarak görüntülenir.

Ayrıca İşveren kullanıcı, sol taraftaki filtre panelinde bulunan "Renklendirme Türü" ölçütüne göre İş Arayan kullanıcıların konumlarını harita üzerinde renklendirebilir. Örneğin Şekil 4.7'de sisteme giriş yapan "Altınbaş Üniversitesi" kullanıcısı, yayınladığı farklı sektörlerdeki iş ilanları için hangi İş Arayan kullanıcının hangi sektörde olduğunu, "Renklendirme Türü" ölçütünü "Sektör" seçerek farklı renklerle görüntüler. Böylelikle hangi sektörde kaç tane İş Arayan kullanıcı var net bir şekilde görür. Ek olarak, İş Arayan kullanıcıların yayınlanan iş ilanlarının konumlarına yakınlığını da harita desteği sayesinde kolaylıkla gözlemler. İş Arayan kullanıcıyı seçtiğinde de detay bilgisini görüntüler.



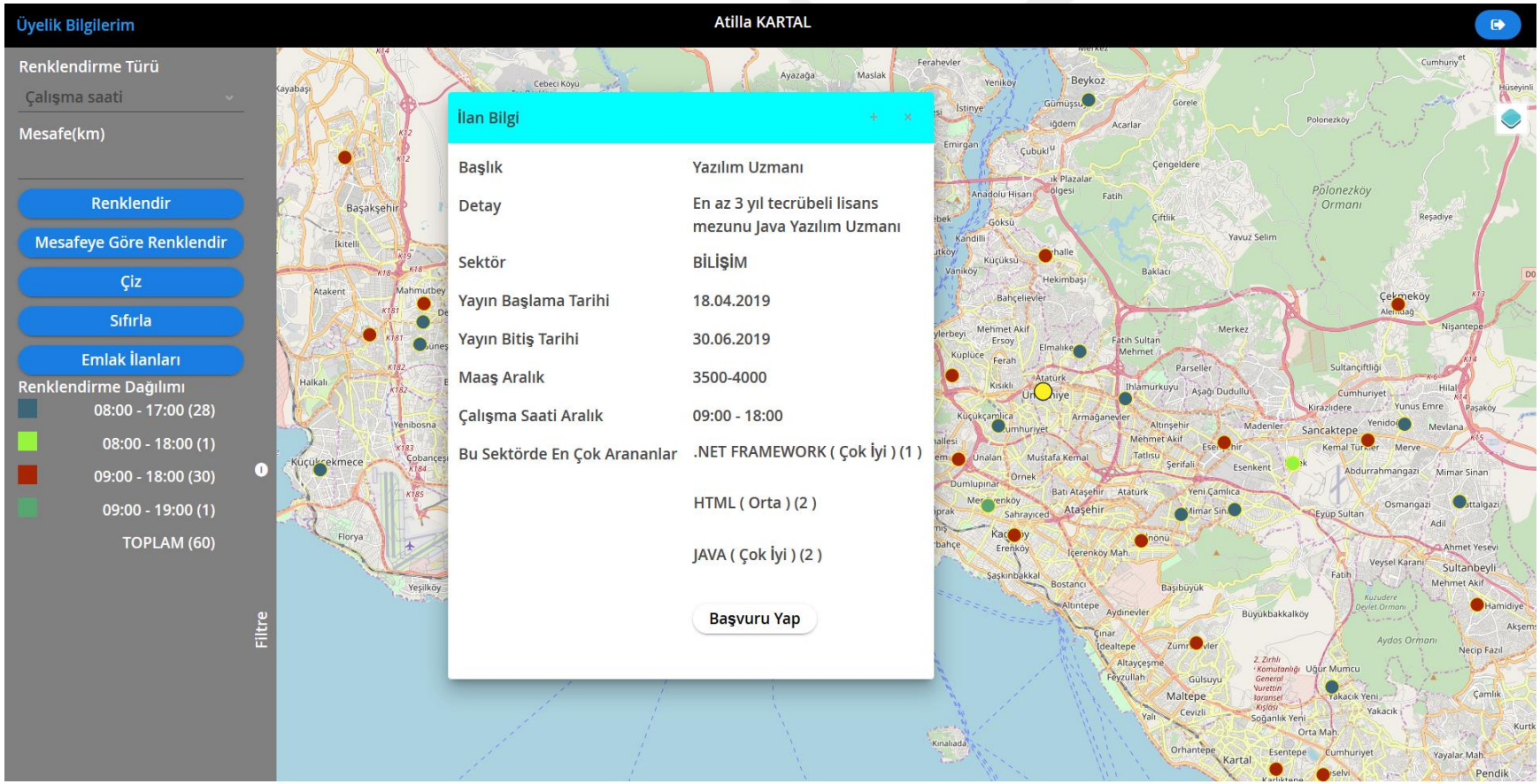
Şekil 4.7: İşveren kullanıcı ana sayfa görünümü

İş Arayan kullanıcılar da sisteme tıpkı İşveren kullanıcılar gibi bir üyelik arayüzünden kendine özel bilgilerini girerek kayıt olurlar ve Şekil 4.8'de olduğu gibi sekmeler halindeki ekranlardan detaylı bilgi girişi yaparlar. Mezun olduğu okul veya eğitimini almış olduğu sertifika bilgilerini eğitim sekmesinde, varsa yabancı dil bilgisini seviyeleri ile yabancı dil sekmesinde, daha önce çalışmış olduğu veya halen çalışmakta olduğu bir kurum varsa deneyim sekmesinde, kendisine referans olabilecek kişileri de referans sekmesinde sisteme kaydeder. İkamet ettikleri konum bilgilerini de bu arayüz üzerindeki haritada işaretlerler. Konum bilgisi tanımı İşveren kullanıcı ve sistemin verimli çalışması için zorunludur.



Şekil 4.8: İş Arayan kullanıcı üyelik detay ekranı

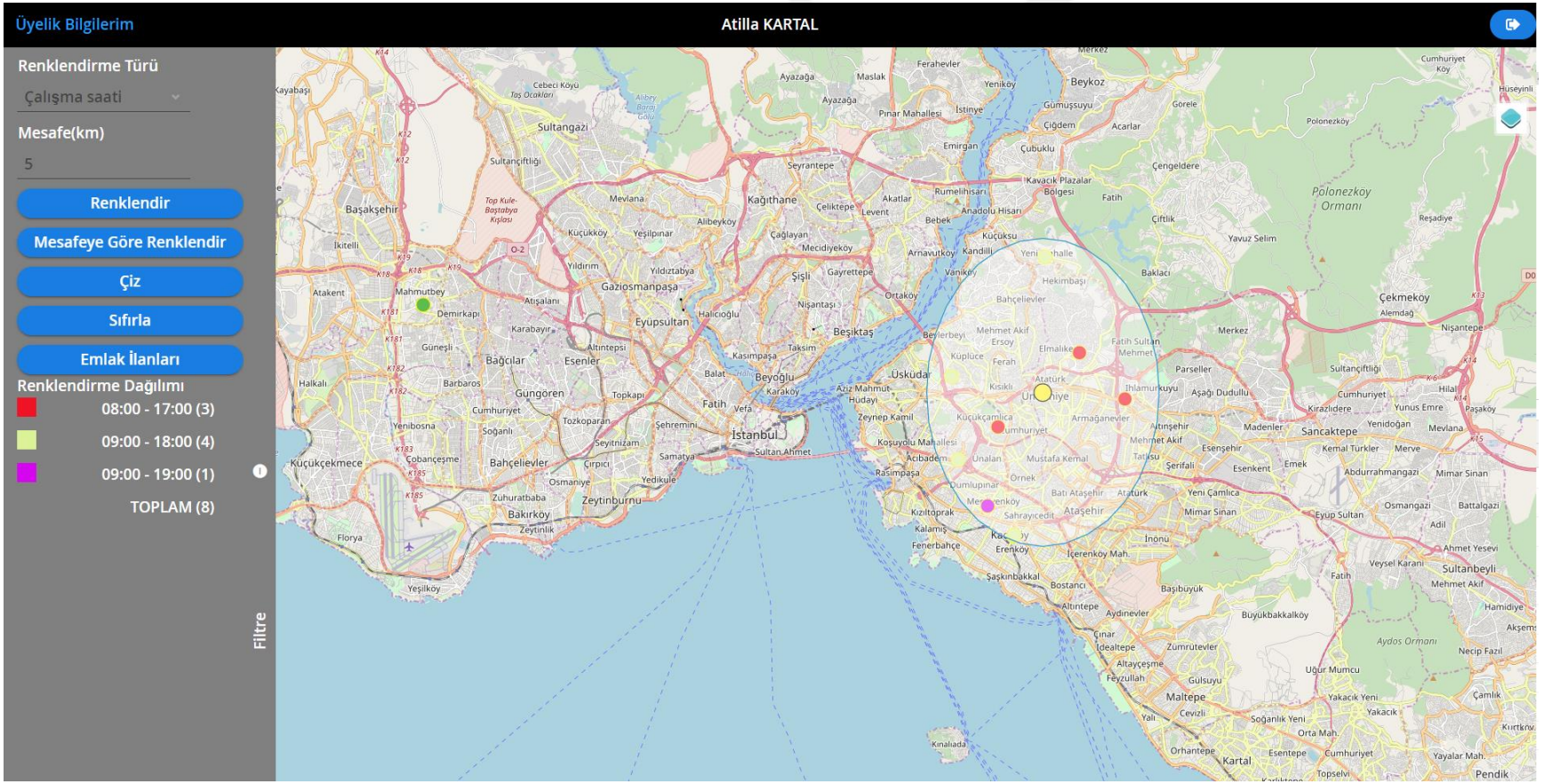
Sisteme giriş yapan İş Arayan kullanıcı ana sayfada, görüntüleyeceği iş ilanlarına ait konumları farklı filtrelere göre renklendirir. Kendi ikamet ettiği konumu diğer noktalardan biraz daha büyük sarı bir nokta olarak görüntüler. Filtresine göre uygun gördüğü ilanın detayını Şekil 4.9'daki gibi görüntüler ve dilerse başvuru yapar. İlana ait bilgilerin görüntülendiği panelde İş Arayan kullanıcının kendi sektöründeki en çok aranan özelliklerin olduğu bir bölüm bulunur. Bu bölüm İş Arayan kullanıcıya, kendi eksiklerini görme imkânı sağlar. Kullanıcı bu panelde başvuru yaptığında, İşveren kullanıcının ilana başvuranlar sekmesine yeni bir kayıt gelir.



Şekil 4.9: İş Arayan kullanıcı ana sayfa görünümü

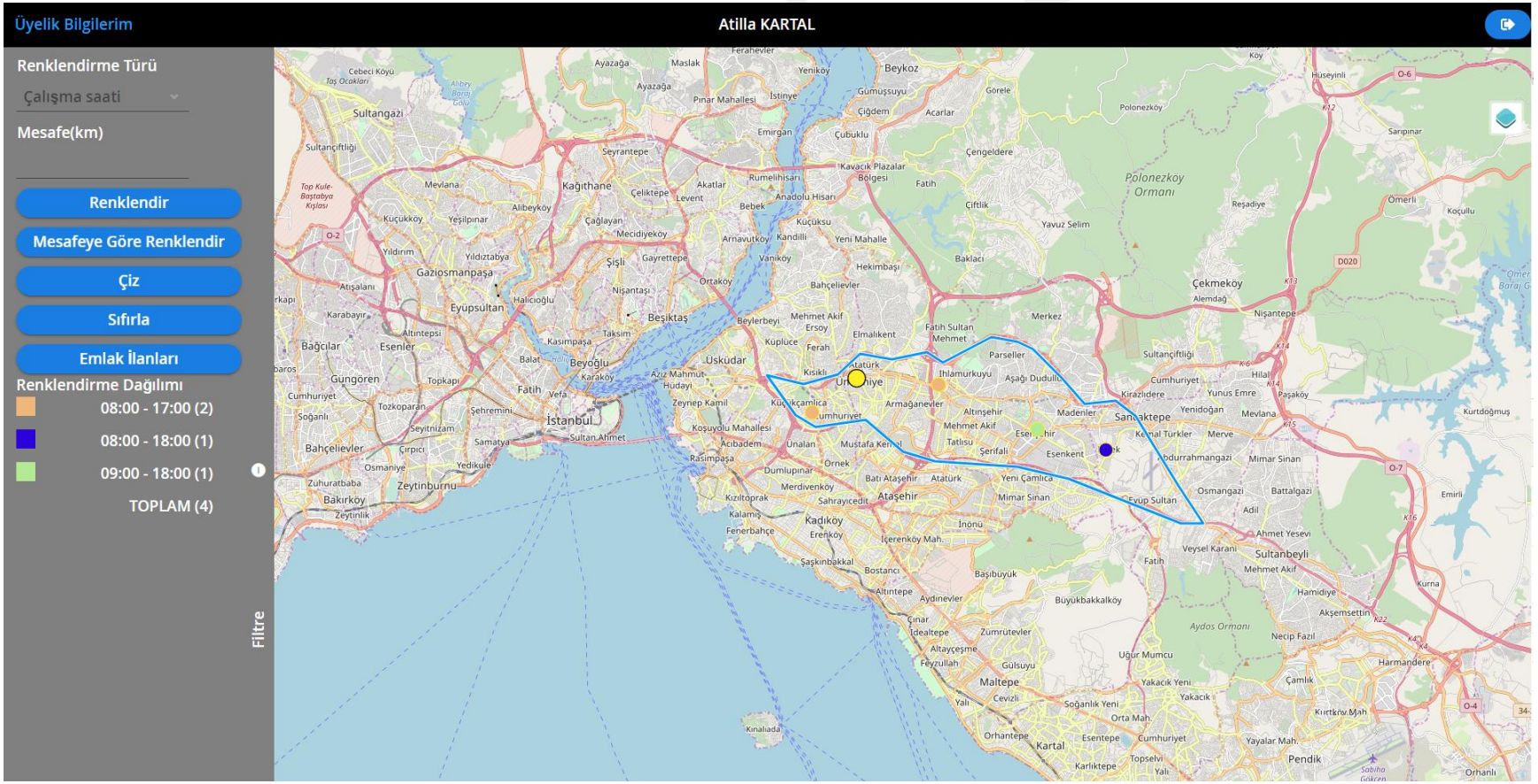
Ana sayfada bulunan filtreleme panelinde kullanıcıların karar vermesine yardımcı olabilecek bileşenler ve bir renklendirme dağılımı alanı mevcuttur. Renklendir butonuna her basıldığında harita üzerindeki noktalar farklı renklerle yeniden üretilir. Haritada gösterilen renkli noktaların renk dağılımı da bu paneldeki bir bölümde görüntülenir.

Kullanıcı kendi konumuna belirli mesafedeki iş ilanlarını görüntülemek istediğinde kilometre cinsinden mesafe girerek “Mesafeye Göre Renklendir” butonuna tıklar. Haritada, kullanıcının konumu orta nokta olarak kabul edilen bir alan çizilir. Girilen mesafe yarıçap uzunluğu olur. Girilen mesafeye göre oluşan alan ve bu alan içerisinde kalan noktalar haritada renklendirme ölçütüne göre Şekil 4.10’daki gibi görüntülenir.



Şekil 4.10: Mesafeye göre renklendirme görünümü

İlçe sınırları içerisine ulaşım başka ilçeye gitmekten zor olabilmektedir. Bu tür senaryolarda "Çiz" bileşeni Şekil 4.11'daki gibi çizim yaparak renklendirmek için kullanılabilir. Kullanıcı burada kendi çizdiği sınırlar içerisindeki noktaları çalışma saati bilgisine göre renklendirmiştir.

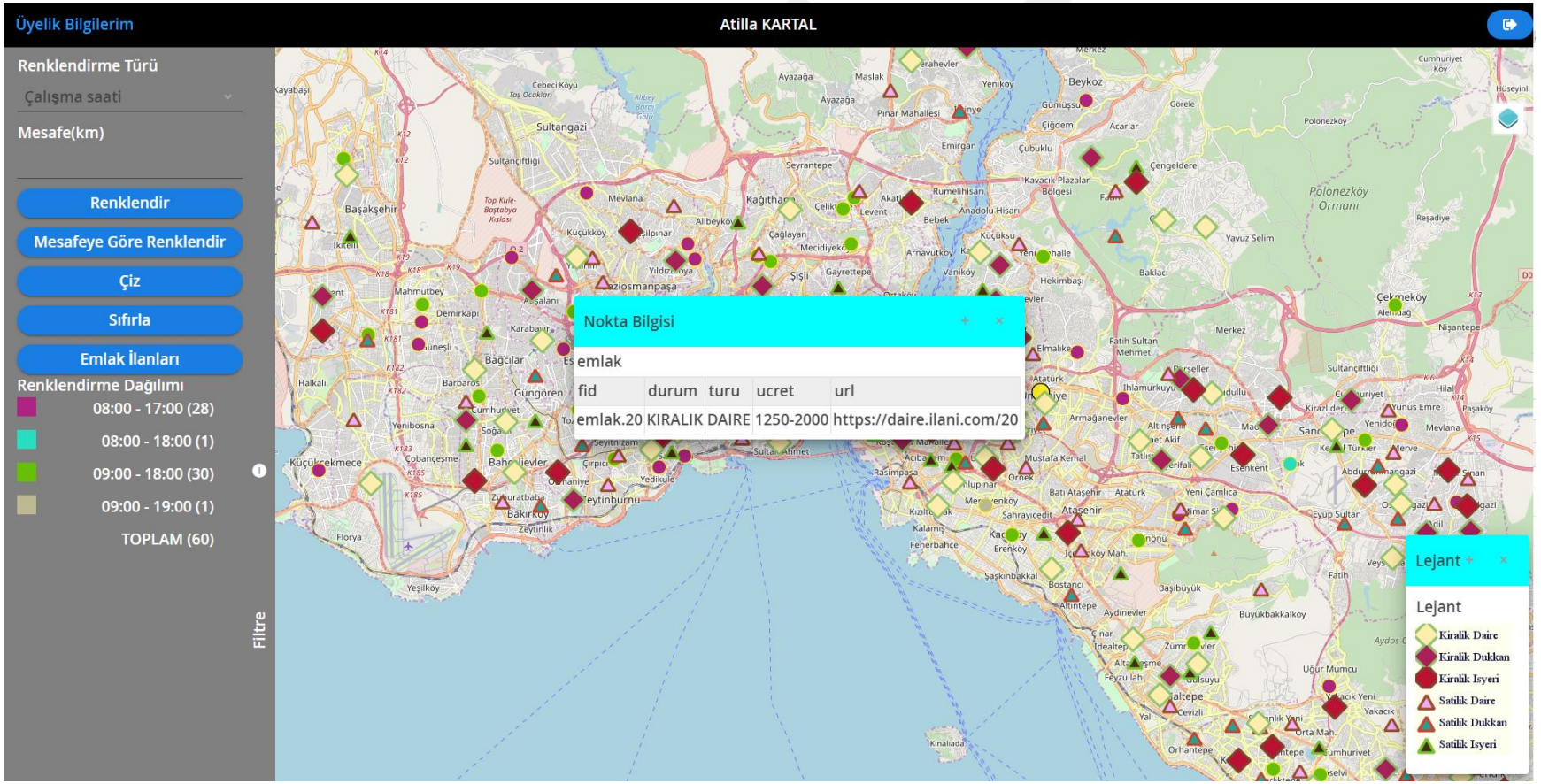


Şekil 4.11: Çizime göre renklendirme görünümü

Kullanıcı yapmış olduğu filtrelemeleri “Sıfırla” bileşenine basarak temizler.

Kimi zaman iş ilanları ikamet edilen konumdan çok uzak olabilir. Bu senaryoya göre İş Arayan kullanıcılar ilanın bulunduğu adrese taşınmayı planlayabilir. Bunun için dış kaynak bir uygulamadan emlak verileri gösterilirse, İş Arayan kullanıcının o ilanı tercihi konusunda karar almasını kolaylaştırır.

GeoServer uygulaması üzerinden emlak ilanları dış kaynaktan temin edildiği varsayılarak yayınlanmıştır. İş Arayan kullanıcı, başvurmak istediği bölgedeki emlak ilanlarını ve detayını Şekil 4.12’deki gibi görüntüler.



Şekil 4.12: Dış kaynaktan temin edildiği varsayılan emlak ilanları ve detayının görünümü

5. SONUÇ

Bu projede, Coğrafi Bilgi Sistemleri'nin bileşenleri aktif olarak kullanılarak konum tabanlı ilan ve eleman arama sistemi oluşturulmuştur. Oluşturulan bu sistemde şu çözümler sunulmuştur:

- İş Arayan kullanıcıların, iş ilanlarını harita üzerinde görüntülemesi sağlanmıştır. Böylelikle iş arayanlar, işe ulaşım için daha az zaman ve çaba harcayacak ve motivasyonu yüksek olarak işe başlayacaktır.

- İşveren kullanıcıların, İş Arayan kullanıcıları harita üzerinde görüntülemesi sağlanmıştır. Böylelikle işverene ek bir yük olan personel ulaşım maliyeti azaltılmıştır. Aynı zamanda trafikte yorulmayan motivasyonu yüksek personellerle çalışma imkânına sahip olmuştur.

- Haritada iş ilanları veya elemanlar seçilen ölçütlere göre renklendirilmiştir. Bu sayede kullanıcılar hem konum tabanlı hem de seçilen ölçüte göre renklendirilmiş bir görünüm elde etmişlerdir.

- Büyük şehirlerdeki trafik sorununa da kısmi olarak çözüm üretilmiştir. İşe gitmek için ne kadar az ulaşım aracı kullanılırsa trafikteki araç sayısı da o kadar azalacaktır. Buna bağlı olarak da trafik yoğunluğunda bir azalma olacaktır.

- Emlak servisi entegrasyonu ile elemanlara iş ilanına yakın bölgelerdeki emlak durumları ve fiyatları gösterilmiştir. Emlak durumları ve özellikle de emlak fiyatları kullanıcıların, konum olarak bu bölgeyi tercih edip etmeyeceğine karar verebilmesini kolaylaştırır.

6. GELECEK ÇALIŞMALAR

Bu bölümde projede daha sonraki zamanlarda ne tür eklemeler yapılabilir konusuna değinilmiştir.

- Renklendirme dağılımının olduğu paneldeki detaylar seçimli yapılabilir. Böylelikle kullanıcı sadece istediği bilgileri haritada görüntüler.

- Sosyal faaliyet alanları, okul, kreş gibi ilgi alanları farklı bir katmanda gösterilerek, İş Arayan kullanıcıların çalışmak istediği kurum konumuna yakınlığı görüntülenebilir. Böylelikle İş Arayan kullanıcılar, işine yakın konumdaki bir kreşe çocuğunu bırakıp, akşam iş çıkışı kreşten çocuğunu alabilir. İş çıkışı bir alışveriş merkezine ya da spor salonuna gidebilir.

- Projeye trafik yoğunluğu ayrı bir katman olarak eklenebilir. Kullanıcılar trafik yoğunluğuna göre ilanı veya elemanı tercih etme konusunda kararını gözden geçirebilir. Trafik yoğunluğu bilgisi, farklı kurumlarla servis entegrasyonu yapılarak elde edilebileceği gibi Admin kullanıcı tarafından günlük, aylık veya belirli periyotlar halinde sisteme tanımlanarak da temin edilebilir.

- Ulaşımda aracını kullanan kullanıcılar için yol tarifi yapan servis sağlayıcılar ile entegrasyon yapılabilir.

- İlanlarda aranan ölçütler ile veri madenciliği yapılarak, kullanıcının eksiklerini görmesi sağlanabilir. Yoğunlaşılması gereken konularda kullanıcı yönlendirilebilir.

- Tercih edilen bölgede bir ilan yayınlandığında veya bir eleman iş aradığında otomatik bilgilendirme yapması için “bölge çizerek otomatik bilgilendir” özelliği eklenebilir. Kullanıcı çizim yaptığı bölgede kendisi ile ilgili bir ilan yayınlandığında haberdar olur.

- Sistemi kullanan kullanıcıların kendilerini daha iyi ifade edebilmeleri için kendi sayfalarını özelleştirmelerine imkân verilebilir.

KAYNAKÇA

[1] M. Kabul, "Çalışanların Örgütsel Stres Kaynakları ve Stresle Başa Çıkma Yöntemleri," 2016.

[2] M. O. Fred and U. M. Kinange, "Effectiveness of E-Recruitment in Organization Development," *Manag. Econ. J.*, vol. 2, no. 4, pp. 294–301, 2018.

[3] İ. Yalçın, "Açık Kaynaklı Web Tabanlı Coğrafi Bilgi Sistemi Geliştirilmesi," Hacettepe Üniversitesi, 2018.

[4] O. Dinç, "Web Cbs Ve Açık Kaynak Kodlu Kampüs Bilgi Sistemi Uygulaması," İstanbul Teknik Üniversitesi, 2018.

[5] N. Amaneddine and N. Chmeit, "Modeling Real-Estate search service using GIS technology," in *2009 17th International Conference on Geoinformatics*, 2009, vol. 62, no. 2.

[6] A. Sezer, M. Deniz, and M. Topuz, "Uşak Şehrinde Okullara Erişebilirliğin Coğrafi Bilgi Sistemleri (CBS) İle Analizi / Analysis of Accessibility of Schools in Uşak City via Geographical Information Systems (GIS)," *J. Hist. Cult. Art Res.*, vol. 7, no. 5, p. 470, 2019.

[7] A. R. Putra and F. Ramdani, "Development of WEB-GIS Based Customer Complaint Management Information System," *Int. Symp. Geoinformatics 2017*, pp. 48–54, 2017.

[8] E. Sosa-Terrazas, M. Parada-Gonzalez, and U. Martinez-Contreras, "Geographic information systems: Proposal for the relocation of a central distribution point through the centroid method," *2018 Syst. Inf. Eng. Des. Symp.*, pp. 209–213, 2018.

[9] M. H. Selamat, M. S. Othman, N. H. M. Shamsuddin, N. I. M. Zukepli, and A. F. Hassan, "A Review on Open Source Architecture in Geographical Information Systems . final outcome and on the vector data Open jump can read and," *2012 Int. Conf. Comput. Inf. Sci.*, vol. 2, pp. 962–966, 2012.

[10] N. Nizamuddin, I. Meutia, and A. Ardiansyah, "Development of WebGIS Application for

Updating Spatial Data of Paddy Fields," *2018 Int. Conf. Electr. Eng. Informatics*, pp. 139–144, 2018.

[11] A. Pinheiro *et al.*, "Implementing interoperability in a Brazilian seismic event database," *2015 10th Iber. Conf. Inf. Syst. Technol. Cist. 2015*, 2015.

[12] M. Huang and Y. Tian, "RIAs and OGC web services based geospatial data searching and visualization," *2nd Int. Conf. Inf. Eng. Comput. Sci. - Proceedings, ICIECS 2010*, pp. 1–4, 2010.

[13] S. Schmid, E. Galicz, and W. Reinhardt, "WMS performance of selected SQL and NoSQL databases," *ICMT 2015 - Int. Conf. Mil. Technol. 2015*, pp. 1–6, 2015.

[14] Y. He, Y. Zheng, J. Deng, and H. Pan, "Design and implementation of a POI collection and management system based on public map service," *4th Int. Conf. Ubiquitous Positioning, Indoor Navig. Locat. Serv. - Proc. IEEE UPINLBS 2016*, pp. 197–200, 2016.

[15] G. Li, "Research on Building the Information Sharing Platform Based on the OGC Web Services," *2013 6th Int. Conf. Inf. Manag. Innov. Manag. Ind. Eng.*, vol. 2, pp. 200–203, 2013.

[16] S. Vartanov, "Dynamic symbolic execution of Java programs using JNI," *11th Int. Conf. Comput. Sci. Inf. Technol. CSIT 2017*, vol. 2017–March, pp. 83–86, 2017.

[17] M. U. H. Al Rasyid, A. Sayfudin, A. Basofi, and A. Sudarsono, "Development of semantic sensor web for monitoring environment conditions," *Proceeding - 2016 Int. Semin. Intell. Technol. Its Appl. ISITIA 2016 Recent Trends Intell. Comput. Technol. Sustain. Energy*, vol. 3, pp. 607–612, 2016.

[18] "Vaadin Overview," 2019. [Online]. Available: https://vaadin.com/docs/v7/framework/introduction/intro-overview.html. [Accessed: 17-Apr-2019].

[19] Y. G. Gutierrez, "Kelluntekun : Rich Internet Application For Collaborative Snippet Management , Using Oows2 . 0 And Vaadin," *2014 33rd Int. Conf. Chil. Comput. Sci. Soc.*, pp. 107–115, 2014.

[20] T.-H. Chen *et al.*, “An empirical study on the practice of maintaining object-relational mapping code in Java systems,” *Proc. 13th Int. Work. Min. Softw. Repos. - MSR ’16*, pp. 165–176, 2016.

[21] J. Armas, P. Navas, T. Mayorga, P. Rengifo, and B. Arévalo, “Optimization of code lines and time of access to information through object-relational mapping (ORM) using alternative tools of connection to database management systems (DBMS),” *2017 2nd Int. Conf. Syst. Reliab. Safety, ICSRS 2017*, vol. 2017–Janua, pp. 500–504, 2017.

[22] Kisman and S. M. Isa, “Hibernate ORM query simplication using hibernate criteria extension (HCE),” *NICS 2016 - Proc. 2016 3rd Natl. Found. Sci. Technol. Dev. Conf. Inf. Comput. Sci.*, pp. 23–28, 2016.

[23] X. Hao and H. Tang, “Struts Spring Hibernate integrated framework and its use in log accounting and analyzing system,” *Proc. - 2010 2nd Int. Conf. Multimed. Inf. Netw. Secur. MINES 2010*, pp. 936–939, 2010.

[24] H. Pandey, H. Rastogi, C. Gupta, and Anuja, “Web-based network management system implemented using Hibernate, JBoss and spring framework,” *2nd Int. Conf. Telecommun. Networks, TEL-NET 2017*, vol. 2017–Janua, pp. 1–5, 2017.

[25] A. Singh, P. Chawla, K. Singh, and A. K. Singh, “Formulating an MVC Framework for Web Development in Java,” *Proc. 2nd Int. Conf. Trends Electron. Informatics, ICOEI 2018*, no. Icoei, pp. 926–929, 2018.

[26] Y. Xiaokun, “Design and Implementation of Student Status Management System Based on SSH2 Framework Technology,” *Proc. - 2016 Int. Conf. Smart City Syst. Eng. ICSCSE 2016*, pp. 222–225, 2016.

[27] A. Mukherjee, Z. Tari, and P. Bertok, “A spring based framework for verification of service composition,” *Proc. - 2011 IEEE Int. Conf. Serv. Comput. SCC 2011*, pp. 258–265, 2011.

[28] Z. H. Xiong and Y. Z. Yang, “Automatic updating method based on Maven,” *Proc. 9th Int. Conf. Comput. Sci. Educ. ICCCSE 2014*, no. Iccse, pp. 1074–1077, 2014.

[29] C. Désarmeaux, A. Pecatikov, and S. McIntosh, “The dispersion of build maintenance activity across maven lifecycle phases,” *Proc. 13th Int. Work. Min. Softw. Repos. - MSR ’16*, pp. 492–495, 2016.

[30] S. Sultana and S. Dixit, “Indexes in PostgreSQL,” *IEEE Int. Conf. Innov. Mech. Ind. Appl. ICIMIA 2017 - Proc.*, no. Icimia, pp. 512–515, 2017.

[31] C. Ledur, D. Griebler, I. Manssour, and L. G. Fernandes, “Towards a Domain-Specific Language for geospatial data visualization maps with Big Data sets,” *Proc. IEEE/ACS Int. Conf. Comput. Syst. Appl. AICCSA*, vol. 2016–July, pp. 1–8, 2016.

[32] P. A. Campbell, J. P. Havlicek, A. R. Stevenson, and R. D. Barnes, “Realization of ITS applications through mapping technologies: A survey of advanced traveler information systems,” *IEEE Conf. Intell. Transp. Syst. Proceedings, ITSC*, pp. 788–795, 2012.

[33] P. Lopes, C. Fonte, L. See, and B. Bechtel, “Using OpenStreetMap data to assist in the creation of LCZ maps,” *2017 Jt. Urban Remote Sens. Event, JURSE 2017*, 2017.

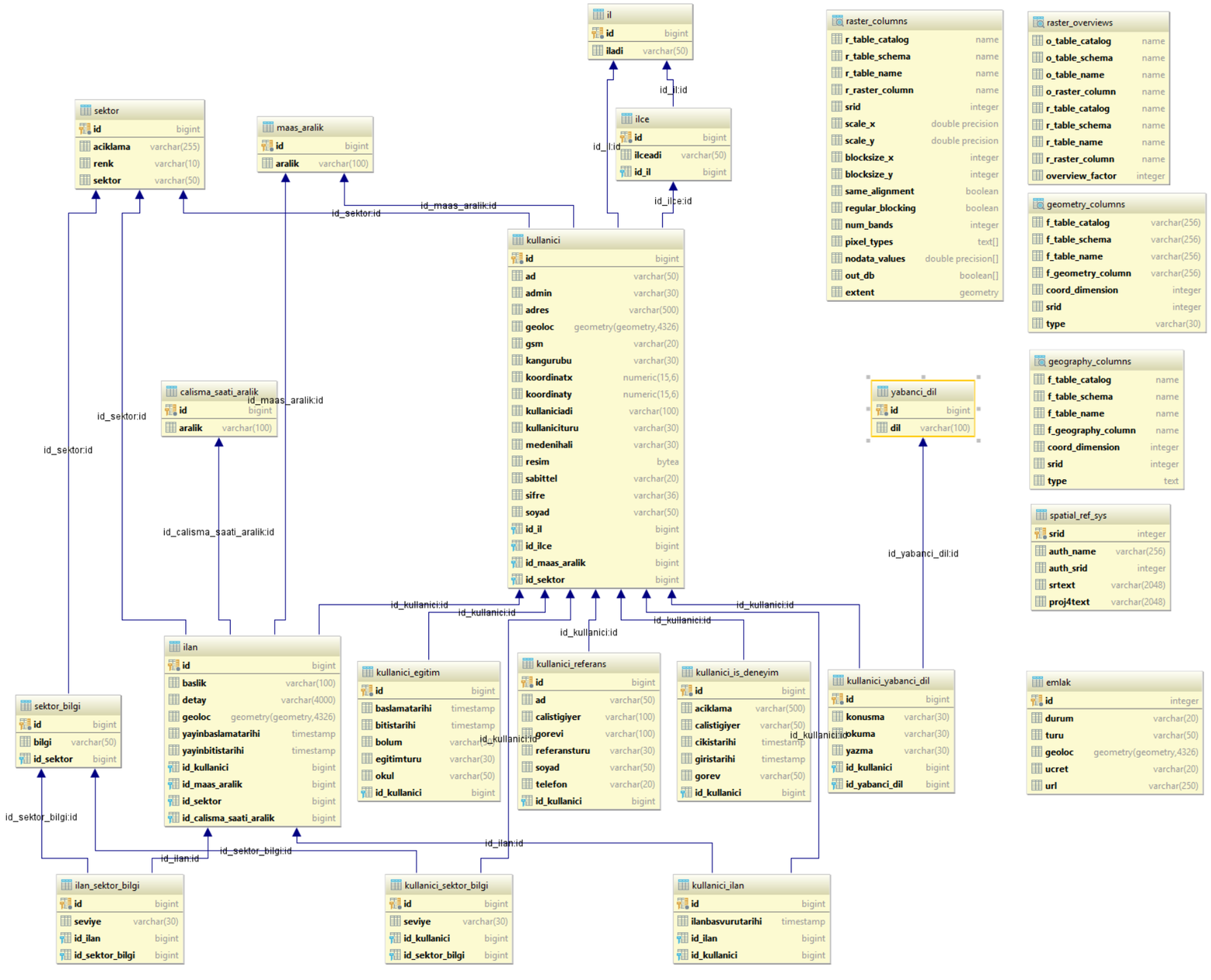
[34] C. L. Yang, S. J. H. Shiau, C. Y. Huang, and J. N. Juang, “Adoption of OpenStreetMap in Taiwan testing SERVQUAL-based on MCDM methods,” *2016 Int. Conf. Appl. Syst. Innov. IEEE ICASI 2016*, vol. 6, pp. 7–10, 2016.

[35] Y. Sun and Z. Xu, “Design solution of Scientific Research Achievements Management System based on .NET multi-layer architecture,” *2010 Int. Conf. Intell. Comput. Technol. Autom. ICICTA 2010*, vol. 2, pp. 68–70, 2010.

[36] P. U. Chavan, M. Murugan, and P. P. Chavan, “A review on software architecture styles with layered robotic software architecture,” *Proc. - 1st Int. Conf. Comput. Commun. Control Autom. ICCUBEA 2015*, pp. 827–831, 2015.

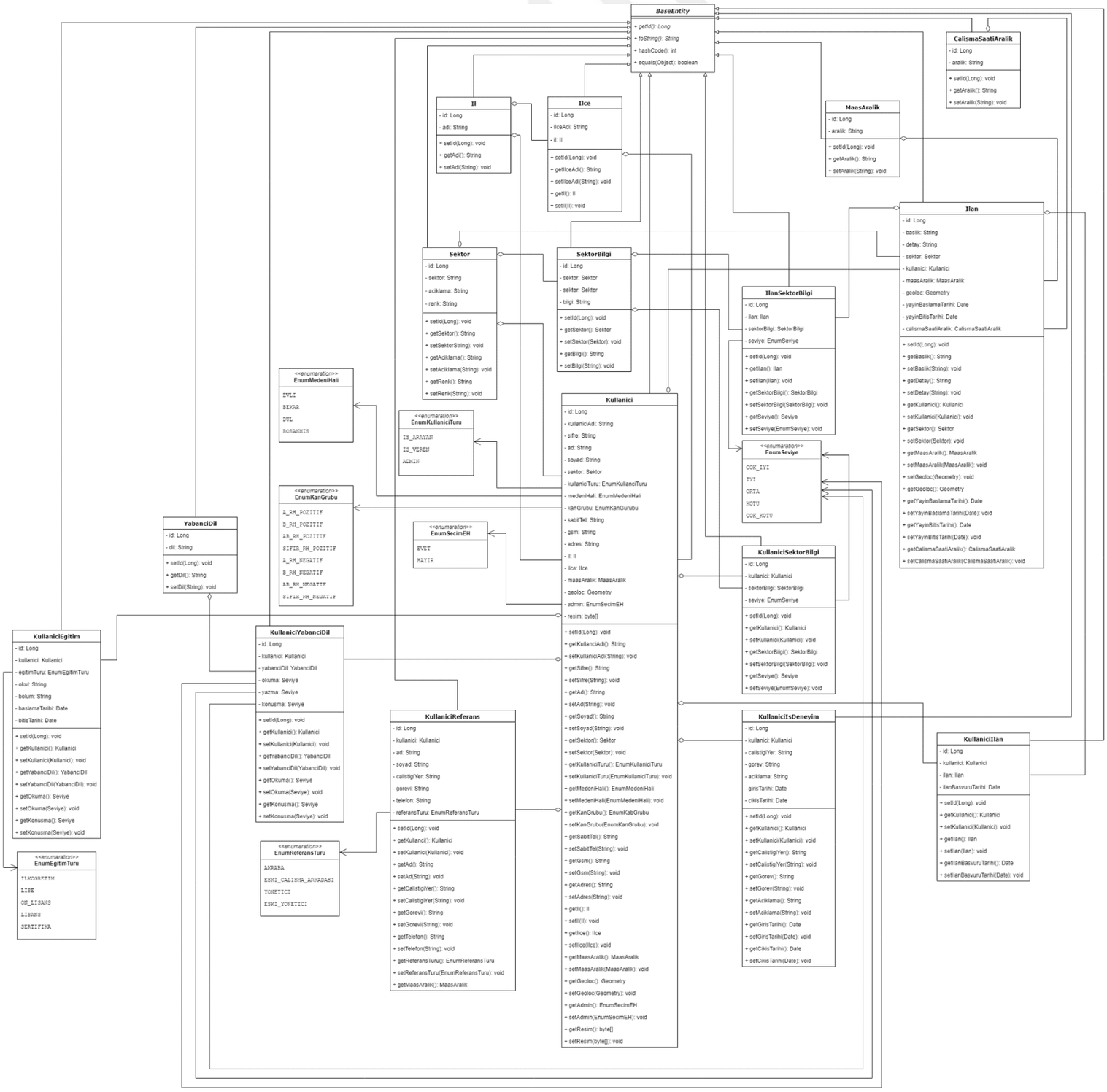
EK A

VERİ TABANI DİYAGRAMI



EK B

ENTITY SINIF DİYAGRAMI



ÖZGEÇMİŞ

Atilla Gökhan Kartal, 28.03.1986 İstanbul doğumludur. Aslen Sivaslıdır. Evli ve 2 çocuk babasıdır. 1993 yılında Burak Reis İlkokulu'nda başlayan ilkokul öğrenimini, 1997 yılında Kemerdere İsmihan İsmet Süzer İlkokulu'nda tamamlamıştır. Orta okulu 2000 yılında Şehit Öğretmen Nurgül Kale Ortaokulu'nda, liseyi de 2004 yılında Ümraniye Lisesi'nde (Yabancı Dil Ağırlıklı) tamamladıktan sonra 2006 yılında başladığı Selahattin Karasoy Meslek Yüksek Okulu'ndaki Bilişim Teknolojileri ve Programlama ön lisans eğitimini 2008 yılında tamamlamıştır. 2008 yılında ön lisans programdan mezun olduktan sonra Açık Öğretim Fakültesi İşletme Bölümü'ne geçiş yaparak 2011 yılında da İşletme Bölümü mezunu olmuştur. 2013 yılında eski adı Kemerburgaz Üniversitesi olan, şimdiki adıyla Altınbaş Üniversitesi'nde Bilişim Teknolojileri Yüksek Lisans programına başvurmuştur ve eğitimine devam etmektedir.

Meslek hayatına 2009 yılında Geovision Group firmasında yazılım uzmanı olarak başlamıştır. Burada 4 yıla yakın bir süre çalıştıktan sonra 2013 yılında şu an halen devam etmekte olduğu Universal Yazılım A.Ş. firmasında yazılım uzmanı olarak göreve başlamıştır.