İSTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

SUNUCUSUZ YAZILIM MİMARİSİYLE COĞRAFİ BİLGİ SİSTEMİ TASARIMI VE UYGULAMASI

DOKTORA TEZİ

Mete Ercan PAKDİL

Bilişim Uygulamaları Anabilim Dalı

Coğrafi Bilgi Teknolojileri Programı

HAZİRAN 2022

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

SUNUCUSUZ YAZILIM MİMARİSİYLE COĞRAFİ BİLGİ SİSTEMİ TASARIMI VE UYGULAMASI

DOKTORA TEZİ

Mete Ercan PAKDİL
(706142003)

Bilişim Uygulamaları Anabilim Dalı

Coğrafi Bilgi Teknolojileri Programı

Tez Danışmanı: Prof. Dr. Rahmi Nurhan ÇELİK

HAZİRAN 2022

İTÜ, Lisansüstü Eğitim Enstitüsü'nün 706142003 numaralı Doktora Öğrencisi Mete Ercan PAKDİL, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı "SUNUCUSUZ YAZILIM MİMARİSİYLE COĞRAFİ BİLGİ SİSTEMİ TASARIMI VE UYGULAMASI" başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı :	**Prof. Dr. Rahmi Nurhan ÇELİK** İstanbul Teknik Üniversitesi	.....
Jüri Üyeleri :	**Prof. Dr. Nesibe Necla ULUĞTEKİN** İstanbul Teknik Üniversitesi	.....
	Prof. Dr. Ali Melih BAŞARANER Yıldız Teknik Üniversitesi	.....
	Doç. Dr. Caner GÜNEY İstanbul Teknik Üniversitesi	.....
	Doç. Dr. Taylan ÖCALAN Yıldız Teknik Üniversitesi	.....

Teslim Tarihi : 14 Mayıs 2022
Savunma Tarihi : 16 Haziran 2022

Akıl, bilim ve vicdana,

ÖNSÖZ

Üniversiteye başladığım ilk yıldan bu zamana kadar bana yalnızca akademik çalışmalarımda değil her konuda rehberlik eden ve desteğini hiç esirgemeyen Sayın Prof. Dr. Rahmi Nurhan ÇELİK'e,

Ayrıca doktora sürecime yaptıkları katkılar ile doktora konusunun olgunlaşmasını sağlayan Sayın Prof. Dr. Nesibe Necla ULUĞTEKİN'e, Sayın Prof. Dr. A. Melih BAŞARANER'e ve Sayın Doç.Dr. Caner GÜNEY'e,

Doktora sürecimde desteklerini ve yardımlarını esirgemeyen Sayın Prof. Dr. Abdullah FERİKOĞLU'na,

Tez çalışması boyunca bana her türlü teknik konuda katkı sağlayan ve tez çalışması sürecinde beni sürekli cesaretlendiren başta Thomas JOSEPH ve tüm Mott MacDonald çalışma arkadaşlarıma,

Beni yetiştiren ve daima yanımda olan Sevgili Annem'e, Babam'a, Kardeşim'e ve tabii ki tüm bu zorlu süreçte bana her zaman destek olan Sevgili Eşim'e,

Saygı ve Teşekkürlerimle.

Mayıs 2022

Mete Ercan PAKDİL
(Geomatik Yüksek Mühendisi)

İÇİNDEKİLER

Sayfa

ÖNSÖZ ..... **vii**
İÇİNDEKİLER ..... **ix**
KISALTMALAR ..... **xi**
ÇİZELGE LİSTESİ ..... **xiii**
ŞEKİL LİSTESİ ..... **xv**
ÖZET ..... **xvii**
SUMMARY ..... **xxi**
1. GİRİŞ **1**
1.1 Tezin Amacı ve Kapsamı 2
1.2 Tez İçeriği ve Kurgu 3
1.3 Literatür İncelemesi 5
2. SUNUCUSUZ KAVRAMI VE SUNUCUSUZ HİZMET TÜRLERİ **9**
2.1 Veri Depolama Hizmetleri 11
2.1.1 Yapısal ve ilişkisel veri depolama hizmetleri 12
2.1.2 Yapısal olmayan veri depolama hizmetleri 14
2.2 Hesaplama Hizmetleri 17
2.2.1 Hizmet olarak fonksiyon 19
2.2.2 Hizmet olarak konteyner 21
3. BULUT BİLİŞİM SAĞLAYICILARININ SUNUCUSUZ HİZMETLERİ **25**
3.1 Amazon Web Services (AWS) Tarafından Sunulan Sunucusuz Hizmetler ve Mekânsal Bilişim Özellikleri 26
3.1.1 Sunucusuz veri depolama hizmetleri 26
3.1.2 Sunucusuz hesaplama hizmetleri 27
3.1.3 Ekonomik model 29
3.2 Microsoft Azure Tarafından Sunulan Sunucusuz Hizmetler ve Mekânsal Bilişim Özellikleri 29
3.2.1 Sunucusuz veri depolama hizmetleri 29
3.2.2 Sunucusuz hesaplama hizmetleri 31
3.2.3 Ekonomik model 33
3.3 AWS ve Microsoft Azure Sunucusuz Hizmetlerinin Karşılaştırılması 34
4. SUNUCUSUZ COĞRAFİ BİLGİ SİSTEMİ YAZILIMI TASARIMLARI VE UYGULAMALARI **37**
4.1 Mimari Tasarımların Değerlendirmesinde Kullanılan Yöntemler 37
4.1.1 12 Faktör yöntemi 38
4.1.2 Bulut bilişim sağlayıcılarının mimari değerlendirme ölçütleri 45
4.2 Sunucusuz Vektör Karo Harita Servisi Tasarımı, Uygulaması ve Değerlendirmesi 49
4.2.1 Tasarım 49
4.2.2 Uygulama 54
4.2.3 Değerlendirme 55

4.3 Sunucusuz Raster Karo Harita Servisi Tasarımı, Uygulaması ve Değerlendirmesi 58
4.3.1 Tasarım 59
4.3.2 Uygulama 62
4.3.3 Değerlendirme 64
4.4 Sunucusuz Mekânsal Zekâ Sistemi Tasarımı, Uygulaması ve Değerlendirmesi 66
4.4.1 Tasarım 67
4.4.2 Uygulama 73
4.4.3 Değerlendirme 76
4.5 Sunucusuz Olay Güdümlü Mekânsal Veri İşleme Servisi Tasarımı, Uygulaması ve Değerlendirmesi: Kandilli Deprem Habercisi Örneği 78
4.5.1 Tasarım 79
4.5.2 Uygulama 83
4.5.3 Değerlendirme 86
4.6 Sunucusuz Mekânsal Analiz İş Akışı Sistemi Tasarımı, Uygulaması ve Değerlendirmesi 88
4.6.1 Tasarım 89
4.6.2 Uygulama 103
4.6.3 Değerlendirme 109
5. SONUÇ VE ÖNERİLER 113
KAYNAKLAR 119
ÖZGEÇMİŞ 127

KISALTMALAR

ACI : Azure Container Instances (Azure Konteyner Örnekleri)

API : Application Programming Interface (Uygulama Geliştirme Arayüzü)

AWS : Amazon Web Services (Amazon Web Hizmetleri)

BLOB : Binary Long Objects (İkili Büyük Objeler)

CBS : Coğrafi Bilgi Sistemi (Geographic Information System)

CAD : Computer-aided Design (Bilgisayar Destekli Tasarım)

CSV : Comma-Separated Values (Virgülle Ayrılmış Değerler)

GB : Gigabyte

GeoAI : Geospatial Artificial Intelligence (Mekânsal Zekâ)

GPU : Graphical Processing Unit (Grafik İşlemci Birimi)

HoA : Hizmet olarak Altyapı (Infrastructure as a Service)

HoF : Hizmet olarak Fonksiyon (Function as a Service)

HoP : Hizmet olarak Platform (Platform as a Service)

HoK : Hizmet olarak Konteyner (Container as a Service)

HoY : Hizmet olarak Yazılım (Software as a Service)

HTML : HyperText Markup Language (Köprülü Metin Dili)

HTTP : Hypertext Transfer Protocol (Köprülü Metin Gönderme Protokolü)

HTTPS : Hypertext Transfer Protocol Secure (Güvenli Köprülü Metin Gönderme Protokolü)

IoT : Internet of Things (Nesnelerin İnterneti)

JSON : JavaScript Object Notation (JavaScript Nesne Notasyonu)

MVT : Mapbox Vector Tiles (Mapbox Vektör Karoları)

OGC : Open Geospatial Consortium

PBF : Protocolbuffer Binary Format

REST : Representational State Transfer (Temsili Durum Transferi)

SDK : Software Development Kit (Uygulama Geliştirme Takımı)

SQL : Structured Query Language (Yapısal Sorgu Dili)

SYM : Sayısal Yükseklik Modeli

TCP : Transmission Control Protocol (Gönderim Kontrol Protokolü)

URL : Uniform Resource Locator (İnternet Kaynak Belirteci)

VPN **:** Virtual Private Network (Sanal Gizli Ağ)

YAML **:** Yet Another Markup Language (Bir Başka İmleme Dili)

WMTS **:** Web Map Tile Service (Web Harita Karo Servisi)

WMS **:** Web Map Service (Web Harita Servisi)

WSGI **:** Web Server Gateway Interface (Web Sunucu Ağ-Geçidi Arayüzü)

WPS **:** Web Processing Service (Web Analiz Servisi)

ÇİZELGE LİSTESİ

Sayfa

Çizelge 2.1 : Sunucusuz hizmet türlerinin sınıflandırılması. **11**
Çizelge 3.1 : AWS sunucusuz veri depolama hizmetleri. **26**
Çizelge 3.2 : AWS sunucusuz hesaplama hizmetleri. **27**
Çizelge 3.3 : Microsoft Azure sunucusuz veri depolama hizmetleri. **30**
Çizelge 3.4 : Microsoft Azure sunucusuz hesaplama hizmetleri. **31**
Çizelge 3.5 : Amazon DynamoDB ve Azure Cosmos DB karşılaştırması **34**
Çizelge 3.6 : AWS Lambda ve Azure Functions karşılaştırması. **35**
Çizelge 3.7 : AWS Fargate ve Azure Container Instances karşılaştırması **35**
Çizelge 4.1 : 12 Faktör yönteminin HoK ve HoF modelleri ile uygulanabilirliği. **45**
Çizelge 4.2 : Vektör karo harita servisinin 12 Faktör yöntemine göre değerlendirilmesi **57**
Çizelge 4.3 : Vektör karo harita servisinin bulut bilişim sağlayıcısı mimari değerlendirme ölçütlerine göre değerlendirilmesi **58**
Çizelge 4.4 : Raster karo harita servisi sisteminin 12 Faktör yöntemine göre değerlendirilmesi **65**
Çizelge 4.5 : Raster karo harita servisinin bulut bilişim sağlayıcısının mimari değerlendirme ölçütlerine göre değerlendirilmesi **66**
Çizelge 4.6 : Mekânsal zekâ sisteminin 12 Faktör yöntemine göre değerlendirilmesi **76**
Çizelge 4.7 : Mekânsal zekâ sisteminin bulut bilişim sağlayıcısının mimari değerlendirme ölçütlerine göre değerlendirilmesi **78**
Çizelge 4.8 : Telegram deprem bildirim sohbet botunun komutları **84**
Çizelge 4.9 : Deprem bildirim sisteminin 12 Faktör yöntemine göre değerlendirilmesi **87**
Çizelge 4.10 : Vektör karo harita servisinin bulut bilişim sağlayıcısı mimari değerlendirme ölçütlerine göre değerlendirilmesi **88**
Çizelge 4.11 : Sunucusuz API yönetimi servisinin adres tablosu ve karşılık gelen bileşenler **97**
Çizelge 4.12 : İş akışı görev tanımı doğrulama kuralları **98**
Çizelge 4.13 : İş akışı tanımı doğrulama kuralları **100**
Çizelge 4.14 : Mekânsal analiz iş akışı sisteminin 12 Faktör yöntemine göre değerlendirilmesi **109**
Çizelge 4.15 : Mekânsal zekâ sisteminin bulut bilişim sağlayıcısı mimari değerlendirme ölçütlerine göre değerlendirilmesi **111**

ŞEKİL LİSTESİ

Sayfa

Şekil 1.1 : Tez kapsamında uygulanan araştırma metodolojisi. **4**
Şekil 2.1 : Bulut bilişim hizmet türlerinin yönetimsel farklılıkları. **9**
Şekil 2.2 : Sunucusuz hizmetlerin birbirlerini olay üzerinden tetiklemeleri. **10**
Şekil 2.3 : Sunucusuz ilişkisel veri tabanı hizmetinin çalışma anında ölçeklendirilmesi **13**
Şekil 2.4 : Yapısal olmayan veri depolama hizmetinin veri saklama yapısı. **14**
Şekil 2.5 : Yapısal olmayan depolamada kova yapısı. **15**
Şekil 2.6 : Yapısal olmayan depolama hizmetindeki bir olayın hesaplama hizmetini tetiklemesi örneği. **16**
Şekil 2.7 : Yapısal olmayan depolama hizmetindeki statik bir CBS web uygulamasının yayınlanması örneği. **17**
Şekil 2.8 : Bir fonksiyona ait soğuk ve sıcak başlatma süreçleri. **19**
Şekil 2.9 : Sunucusuz hizmetlerin olay güdümlü olarak birbirlerini çalıştırmaları. **20**
Şekil 2.10 : HoF ve HoK modellerinde kullanıcı ve sağlayıcı sorumlulukları. **21**
Şekil 2.11 : Bir Docker konteynerin oluşturulma süreci. **22**
Şekil 2.12 : HoK servisi üzerinde otomatik ölçeklendirme. **23**
Şekil 3.1 : AWS, Azure, Google ve IBM bulut bilişim sağlayıcılarının 2021 son çeyreğindeki pazar payları **25**
Şekil 3.2 : Google Trends'e göre AWS, Azure, Google ve IBM bulut bilişim sağlayıcılarının sunucusuz konusunda son bir sene içindeki aranma popülerliği. **25**
Şekil 4.1 : Uygulama geliştirici, kod deposu ve uygulama çalışma ortamı ilişkisi. **39**
Şekil 4.2 : Destek servislerinin doğru şekilde kullanımı **41**
Şekil 4.3 : Konteyner üzerinde çalışan uygulamanın portlarını bağlama gösterimi **42**
Şekil 4.4 : Vektör karo harita servisi kullanım senaryosu **50**
Şekil 4.5 : Vektör karo harita servisi sistemi tasarımı ve bileşenleri **52**
Şekil 4.6 : Vektör karo harita servisinin AWS bulut bilişim sağlayıcısı üzerindeki sunucusuz hizmetlerle uygulaması **54**
Şekil 4.7 : Raster karo harita servisinin kullanım senaryosu **59**
Şekil 4.8 : Raster karo harita servisi sistemi tasarımı ve bileşenleri **61**
Şekil 4.9 : Raster karo harita servisinin Microsoft Azure bulut bilişim sağlayıcısı üzerindeki sunucusuz hizmetlerle uygulaması **62**
Şekil 4.10 : Azure Blob Storage konteynerlerinin Azure Functions konteynerinin dosya sistemine bağlanması **63**
Şekil 4.11 : Mekânsal analiz servisi sisteminin kullanım senaryosu **67**
Şekil 4.12 : Mekânsal zekâ sisteminin sunucusuz mimaride sistem tasarımı **70**
Şekil 4.13 : Mekânsal zekâ sistemi yapısal veri tabanı tabloları **72**
Şekil 4.14 : Mekânsal zekâ sisteminin Microsoft Azure bulut bilişim sağlayıcısı üzerindeki sunucusuz hizmetlerle uygulaması **73**
Şekil 4.15 : Deprem bildirim sisteminin kullanım senaryosu **79**
Şekil 4.16 : Deprem bildirim sistemi tasarımı ve bileşenleri **81**

Şekil 4.17 : Deprem bildirim sistemi tasarımının AWS bulut bilişim sağlayıcısına ait sunucusuz hizmetlerle uygulaması **83**
Şekil 4.18 : Aboneler tablosu şeması **85**
Şekil 4.19 : Amazon CloudWatch servisinden gelen deprem bildirimi gönderici bileşeni ile ilgili sorun olduğunu belirten alarm e-postası **86**
Şekil 4.20 : Mekânsal analiz iş akışı sisteminin kullanım senaryosu **90**
Şekil 4.21 : Mekânsal analiz iş akışı sisteminin tasarımı **92**
Şekil 4.22 : İş akışı yönetim servisi bileşeninin diğer bileşenlere çalışması ve OGC API Processes işlemlerinin kullanımı **94**
Şekil 4.23 : Konteynerlerin yaşam döngüsünün iş akışının çalışması ile ilişkisi **96**
Şekil 4.24 : İş akışı görev tanımı formatı **98**
Şekil 4.25 : İş akışı tanımı formatı **99**
Şekil 4.26 : İş akışı görevi konteyner imajı tanımı **102**
Şekil 4.27 : Mekânsal analiz iş akışı sisteminin AWS bulut bilişim sağlayıcısı üzerinde uygulaması **103**
Şekil 4.28 : Mekânsal taşkın analizinin iş akış şeması **105**
Şekil 4.29 : Taşkın analizinin iş akışı tanımına göre yazılımı **106**
Şekil 4.30 : Paralel ve tekrarlı iş akışı görevlerinin raster analizi için iş akışı içerisinde birlikte kullanımı **107**
Şekil 4.31 : Sayısal yükseklik modellerinde eğim ve bakı analizi iş akışının geliştirilen tanımlama modeline göre yazımı **108**

SUNUCUSUZ YAZILIM MİMARİSİYLE COĞRAFİ BİLGİ SİSTEMİ TASARIMI VE UYGULAMASI

ÖZET

Bulut bilişim teknolojileri her şeyin bir servis (Everything as a Service) olduğu bir anlayışla birlikte gelişmektedir. Her geçen gün ortaya konan yeni bir altyapı yönetim yaklaşımına da bu kapsamda isimler verilmiştir. Günümüzde bulut bilişim altyapısının yönetiminde farklılıklar sunan üç farklı servis modeli bulunmaktadır. Bu servis modelleri; Hizmet olarak Altyapı (HoA - Infrastructure as a Service), Hizmet olarak Platform (HoP - Platform as a Service), Hizmet olarak Yazılım (HoY - Software as a Service) şeklinde isimlendirilmektedir. Her bir servis modeli bir diğerine göre daha fazla bulut bilişim altyapısının yönetimini soyutlaştırarak farklılaşmaktadır. Bu üç modele ek olarak ortaya çıkan sunucusuz paradigması ise HoY ile HoP arasında konumlandırılabilir. Bu paradigma içerisinde ortaya konan sistem mimarileri de sunucusuz mimariler olarak adlandırılmaktadır.

Sunucusuz kavramı ismen yanıltıcı olabilmekte ve kullanıcıda fiziksel veya sanal bir sunucu yok algısı oluşturmaktadır. Bu algının aksine uygulamalar ve hizmetler yine sunucular üzerinde çalışmakta ancak sunucunun tüm yönetimi bulut bilişim sağlayıcısı tarafından yapılmaktadır. Sunucusuz mimariler özellikle yüksek ölçeklenebilir uygulamalarda bilişim altyapısının yönetimi giderek zorlaşması nedeniyle daha çok tercih edilmektedir.

Konteyner teknolojisi sunucusuz paradigmasının gelişmesinde büyük rol oynamıştır. Sanal makinelere göre uygulamaların başlama hızı, taşınabilirliği ve kullanılabilirliği daha iyileştirmiş olması nedeniyle konteyner kullanımı giderek yaygınlaşmıştır. Bu yeni akım farklı teknik zorlukları ve çözümlerini de beraberinde getirmiştir. Özellikle birden çok konteyner uygulamasının birlikte çalışması ve yüksek ölçeklenebilirliğin sağlanabilmesi problemi için konteyner orkestrasyon platformları çözüm olarak ortaya çıkmıştır. Bunlardan en bilinen konteyner orkestrasyon platformu Kubernetes olarak adlandırılmaktadır.

Bulut bilişim sağlayıcıları hızla konteyner orkestrasyon platformlarını kendilerine dahil ederek hizmet olarak sunmaya başlamışlardır. Bu platformların getirdiği kolaylıklar beraberinde yeni gereksinimler de doğurmuştur. Konteyner orkestrasyon yazılımlarının doğru şekilde yapılandırılması ve sağlıklı çalışabilmesi için kullanıcıları için ileri düzey bulut bilişim altyapısı bilgisine sahip olma ve eğitiminin alınmasını gerektirmektedir. Sunucusuz hizmetler bu gereklilikleri ortadan kaldırarak kullanıcıların ileri düzey altyapı bilgisi sahibi olmadan da yüksek ölçeklenebilir uygulamaları bulut bilişim platformları üzerinde çalıştırabilmelerini sağlamaktadır.

Sunucusuz mimarilerin giderek yaygınlaşması ve bulut bilişim sağlayıcıları tarafından benimsenmesi ile farklı bulut bilişim hizmet modelleri ortaya çıkmıştır. Herhangi bir yazılım sistemi mimarisinin tasarımında iki temel bileşen türü sıklıkla kullanılmaktadır. Bunlar hesaplama ve veri depolama bileşenleridir. Bu çalışmada da

sunucusuz hizmet modelleri veri depolama ve hesaplama olarak iki grupta incelenmiştir. Böylece çalışmada sunulan sistem mimarilerindeki temel gereksinimler bu gruplardan karşılanmıştır.

Çalışmada veri depolama hizmetleri de kendi içinde yapısal ve yapısal olmayan veriler için iki gruba ayrılarak incelenmiştir. İlk grupta incelenen yapısal veri depolama hizmetleri ise verilerin sorgulanma yeteneklerine göre iki alt grupta incelenmiştir.

Yapısal veri türleri; belge, sütun ailesi, anahtar-değer (key-value) ve çizge (graph) türünde sınıflandırılarak incelenmiştir. Literatürde bu veri depolama türleri NoSQL adı altında toplanmaktadır. Bunun sebebi ise veri sorgulamanın SQL dili dışında özel geliştirilmiş diller veya API'larla sunulmasıdır.

Sistem mimarilerinde yapısal veri depolama hizmetlerinin seçimine kullanım şekline bağlı olarak karar verilmelidir. Örneğin anahtar-değer veri tabanlarında anahtar alanı dışında sorgulama yapılmak istendiğinde sorgulama hızı diğer depolama türlerine göre oldukça yavaş ve verimsiz olacaktır. Çalışmada sunulan sistem tasarımlarında bu veri depolama hizmetleri kullanılarak CBS uygulamalarında nasıl katkı verebilecekleri gösterilmiştir.

Bir diğer yapısal veri depolama alt grubunda ise ilişkisel veri depolama hizmetleri bulunmaktadır. Burada incelenen veri depolama hizmetleri ise verilerin SQL diliyle sorgulanmasını sağlamaktadır. Bir diğer karakteristik özelliği ise verilerin tablolar halinde saklanması ve tablolar arasında belirli anahtar değerler üzerinden ilişki kurulabilmesidir. İlişkisel veri tabanları birçok mevcut CBS sunucusu yazılımı tarafından desteklenmektedir. Bu nedenle bu hizmet türünün varlığı mevcut CBS mimarilerinin sunucusuz mimarilere taşınmasını kolaylaştırmaktadır.

Yapısal olmayan veri depolama hizmetleri ise verilerin yapısından bağımsız olarak onları ikili (binary) objeler olarak saklanmasını sağlamaktadır. Literatürde saklanan bu verilere BLOB ismi verilmektedir. Ayrıca yapısal veri depolama hizmetlerinin klasik dosya sistemlerinden nasıl ayrıldıkları da açıklanmıştır. Depolanan objelere birer anahtar değer verilerek bu değerler üzerinden hızlı erişim sağlanmaktadır.

Yapısal olmayan veri depolama hizmetlerinin bir diğer özelliği ise yüklenen HTML sayfalarını yardımcı dosyalarıyla birlikte web sitesi olarak sunabilmesidir. Çalışmada bu özellikten faydalanılarak bir web CBS uygulamasının yapısal olmayan veri depolama hizmeti üzerinde nasıl konumlandırılabileceği de açıklanmıştır.

Çalışmada sunucusuz hesaplama hizmetleri de kendi içinde fonksiyon ve konteyner türünde iki gruba ayrılarak incelenmiştir. Çalışmada fonksiyon türündeki hizmetler Hizmet olarak Fonksiyon (HoF) ve konteyner türündeki hizmetler ise Hizmet olarak Konteyner (HoK) olarak isimlendirilmiştir. İki hizmet türü de temelde konteyner teknolojisini kullanmaktadır. Konteyner ve fonksiyon hizmet türü arasındaki fark kullanıcıların hangi seviyede geliştirme yapabilmesi üzerinden oluşmaktadır. Konteyner hizmet türündeki sunucusuz hesaplama hizmetleri kullanıcılara uygulamanın çalışacağı konteyneri özelleştirebilmesini de sağlar.

Fonksiyon türündeki sunucusuz hesaplama hizmetleri kullanıcıların yükledikleri uygulama kodunu çalıştırırlar. Platforma bağlı olarak destek verilen programlama dilleri de değişiklik göstermektedir. Uygulamalar olay güdümlü olarak çalışırlar. Olayların kaynağı ise platform üzerindeki diğer bulut bilişim servisleri olabileceği gibi internet istekleri de olabilir. Çalışmada akıllı şehir mimarilerinde sıkça kullanılan nesnelerin interneti sensörlerinin olay kaynağı olarak fonksiyonları olay güdümlü olarak nasıl çalıştırabildiği bir örnekle açıklanmıştır. Çalışmada yapısal olmayan veri

depolama hizmetlerinin sunucusuz hesaplama hizmetleri ile olan ilişkisi de incelenmiştir. Yapısal olmayan veri depolama hizmetleri üzerindeki objelere ait durum değişimleri ve yeni objelerin eklenmesi sonucu üretilen olaylar sunucusuz hesapla hizmetlerini tetikleyebilmektedir.

Konteyner türündeki sunucusuz hesaplama hizmetleri ise kullanıcıların hazırladıkları konteyner imajlarını çalıştırırlar. Özellikle bir uygulamanın çalışması için işletim sistemine kurulması gereken bağımlılıkları varsa HoF modelinde bu bağımlılıkların kurulmasına izin verilmezken HoK bu konuda çözüm sunmaktadır. Mevcut CBS sunucusu yazılımları sunucusuz mimariye taşınırken daha özgür bir çalışma ortamı sunduğu için HoK modeli tercih edilmektedir. HoK modelinin bir diğer tercih nedeni ise HoF modeline göre daha uzun sürelerde çalışmayı sağlamasıdır.

Her şeyin bir servis olduğu bulut bilişim dünyasında sunucusuz hizmetler sadece iki grup üzerinden düşünülmemelidir. Bir sistem mimarisinin ihtiyaç duyabileceği hata günlüklerinin tutulması, uygulama ayarlarının saklanması veya mesaj kuyrukları gibi hizmetlerde sunucusuz hizmetler çatısı altında sunulmaya başlanmıştır. Literatürde bu hizmetlere destek servisleri (back-end services) de denilmektedir. Çalışmada sunulan sistem mimarilerinde bu servisler de açıklanarak kullanılmıştır.

Çalışmada en çok kullanılan iki bulut bilişim sağlayıcısının sunduğu sunucusuz hizmetler veri depolama ve hesaplama türlerine göre ayrı ayrı incelenmiştir. Sunulan hizmetlerin mekânsal bilişim özellikleri de kullanım şekilleriyle birlikte verilmiştir. Her bir bulut bilişim sağlayıcısının sunucusuz hizmetlerde uyguladıkları ekonomik model de açıklanmıştır. İki bulut bilişim sağlayıcısının sunucusuz hizmetleri karşılaştırmalı olarak da incelenmiş ve farklar irdelenmiştir.

Sistem mimarilerinin değerlendirilmesi için kullanılan iki farklı değerlendirme yöntemi açıklanmıştır. Bu değerlendirme yöntemlerinin sunucusuz mimarilere uygulaması detaylı bir şekilde açıklanmıştır. Yöntemlere ait ölçüt ve prensiplerin sunucusuz mimariler için uygulanabilirlikleri değerlendirilmiştir.

Çalışmada vektör karo harita servisi, raster karo harita servisi, mekânsal zekâ, olay güdümlü deprem bildirim servisi ve mekânsal analiz iş akışı sistemlerinin sunucusuz mimaride sistem tasarımları sunulmuştur. Her bir tasarım CBS kullanım senaryoları, roller ve gereksinimler açıklanarak desteklenmiştir. Tasarımlar seçilen bir bulut bilişim sağlayıcısı üzerinde uygulanarak açıklanmıştır. Tasarım ve uygulamalar çalışmada açıklanan 12 Faktör yöntemi ve bulut bilişim sağlayıcılarının mimari değerlendirme ölçütlerine göre değerlendirilmiştir.

Mekânsal analiz iş akışı sistemi tasarımında kullanılmak üzere iş akışlarının tanımlanabilmesi için iş akışı ve iş akışı görev tanımları da geliştirilmiştir. Böylece kullanıcıların kolaylıkla sunucusuz mimaride çalışmak üzere mekânsal iş analizlerini tasarlamalarına imkân sunulmuştur.

Sunucusuz mimariler CBS kullanım senaryoları üzerinden literatürde ilk kez bu kadar kapsamlı incelenmiştir. Çalışma kapsamında incelenen sunucusuz hizmet türlerinin mekânsal özellikleri literatürde ilk kez bir arada derlenerek incelenmiştir. Sunulan mekânsal analiz iş akışı sistemi ve sistemin ortaya koyduğu iş akışı tanımlama tasarımları da literatürde özgündür.

Bu tez çalışmasının hedeflerinden biri de sunucusuz mimarilerin mekânsal bilişim sistemlerinde daha fazla kullanılmasına öncülük etmek ve mekânsal bilişim kullanıcıları içinde farkındalığının artmasını sağlamaktır. Çalışmada sunulan öncül tasarımların gelecekte yapılacak benzer çalışmalara kılavuz olması beklenmektedir.

DESIGN AND APPLICATION OF SERVERLESS ARCHITECTURES IN GEOGRAPHIC INFORMATION SYSTEM

SUMMARY

Cloud computing technologies have developed with “everything as a service” approach. New infrastructure management service models have been introduced day by day and have been named in accordance with this approach. Today, three different service models differ in the level of cloud computing infrastructure management. Current service models are Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Each model differs from the other by abstracting cloud infrastructure resources. A cloud computing system consists of three fundamental layers that require management; operating system, data and application. In the IaaS model, the user is expected to manage all these three components. Besides, in the PaaS model, data and application layers are expected to be managed by the user. Finally, in the SaaS model, all components are managed by the cloud computing provider and the user is given access to use them via a provided application interface. In addition to these three models, the serverless paradigm has emerged and is positioned between SaaS and PaaS models. System architectures that leverage this paradigm are also called serverless architectures.

The concept of serverless can be misleading for users and creates the perception that there is no physical or virtual server. On the contrary, applications and services still run on servers, but servers are abstracted from users and are fully managed by the cloud computing provider. As the management of highly scalable infrastructure is getting more difficult and complex, serverless architectures are more preferred especially in such scenarios that require high scalability.

Needless to say, container technology has played a major role in the development of the serverless paradigm. The use of containers has become increasingly popular, as it has improved start-up speed, portability and disposability of applications compared to virtual machines. Moreover, this new trend has brought different technical challenges together with solutions. In particular, container orchestration platforms have emerged as a solution to enable multiple container applications to work together for providing high scalability. The most well-known orchestration platform is named Kubernetes.

Cloud computing providers have rapidly adapted to container orchestration platforms and provide them as a service in their service portfolios. While these platforms bring convenience to container management, they also introduced new requirements in infrastructure management. Container orchestration platforms require advanced cloud computing infrastructure knowledge and training for their users so that users can properly configure them to make them work properly. Fortunately, serverless services eliminate these requirements and enable users to run highly scalable applications on cloud computing platforms without having advanced infrastructure knowledge.

With the increasing usage of serverless architectures and their adoption by cloud computing providers, different cloud computing service models have emerged in cloud computing platforms. Two essential component types are frequently used in the design of any software system architecture. These component types are called, computation

and data storage. In this study, serverless service models are also grouped and examined based on these component types. Thus, component requirements in the system architectures presented in the study were fulfilled by these groups.
In the study, data storage services are also examined by dividing them into two groups based on the stored data structure. Structural data storage services, which were examined in the first group, are also studied in two subgroups based on data querying capabilities.
It is observed that structured data storage services offer different solutions for data querying. These solutions differ according to the structure type of the data. Data structures are categorised as follows: document, column family, key-value, and graph. In the literature, these data storage types are gathered under the name of NoSQL. The reason is that the data is queried with specially developed languages or APIs other than SQL language.
The structured data type should be selected depending on data requirements. For example, the query performance is relatively slow when a field is used rather than the key field in key-value data storage services. In the system designs proposed in the study, it is shown how structured data storage services can contribute to GIS applications with these data structure types.
Relational data storage services are the second subgroup under the structured data storage services group. These data storage services provide SQL support for querying data. Another characteristic feature is that the data is stored in tables and a relationship can be established between the tables over certain key values. Relational databases are supported by many existing mature GIS server applications. Therefore, migration of existing GIS architectures relies on relational databases to serverless architectures can be considered as possible with these services.
It is observed that serverless relational database services are built on existing well-known database server applications that were not originally built for cloud infrastructures. In this study, it is explained how serverless relational database services are delivered in serverless architecture in a highly scalable manner while using existing database server applications that are not could native.
Unstructured data storage services, on the other hand, allow data to be stored as binary objects, regardless of their structure. These binary objects in the literature are called BLOB. Unstructured data storage services have a different hierarchy compared to file systems. In unstructured data storage services, objects are stored along with a key value that is very similar to the key-value database concept. Assigning a key value to the stored objects provides fast access to objects.
Another feature of unstructured data storage services is that they can serve a static website from stored HTML pages and asset files such as images and JavaScript files. In this study, it is also explained how a web GIS application can be served on the unstructured data storage service by leveraging this feature.
Serverless computational services were also examined by dividing them into two groups based on their deployment types; function and container. Serverless computational services based on function deployment are called Function as a Service (FaaS) and similarly, container deployments are called Container as a Service (CaaS). Both types of services use container technology at the foundation. The difference between container and function service type is based on what level users can deploy.
Serverless function services only allow deploying application code and abstract container management from the user. Depending on the cloud provider, the supported programming languages vary. Function applications run as event driven. The source of the events can be either other cloud computing services on the platform or web

requests. In this study, it is explained with an example that illustrates how the (Internet of Things) IoT sensors, which are frequently used in smart city systems, can be considered as event-source to trigger the functions as a part of event-driven architecture. The relationship between unstructured data storage services and serverless computing services is also examined. In addition, changes in the binary objects and the new objects on the unstructured data storage services can produce events that trigger serverless computational services.

Serverless container services, on the other hand, run custom-built container images that are developed and deployed by the user. In particular, if an application has dependencies that need to be installed on the operating system before running, they can be installed on the CaaS model deployments. In contrast, these dependencies are not allowed to install at the operating system level in FaaS model deployments. The CaaS model can be preferred when an existing GIS system is migrated to serverless architectures because it offers the freedom to customize the operating system and run heavy computational applications. It is also preferred when the application requires longer runtime that is limited in the FaaS service model.

In the cloud computing world, where everything offers as a service, serverless services should not be considered only in two groups. Services such as keeping error logs, storing application settings or message queues are also offered as part of the serverless paradigm. These services play also important roles in serverless architectures. In the literature, these supporting services are also called Back-end as a Service (BaaS) services. In this study, proposed system architectures leverage these services to demonstrate how they can be useful.

Serverless services offered by two mature and mostly used cloud computing providers are reviewed from the geospatial point of view in accordance with the serverless services categorisation presented in this study. Geospatial features offered by these providers are given together with their usage patterns. The economic model that each cloud computing provider implements in serverless services is also explained. Furthermore, the serverless services of these providers are also analysed comparatively and the differences are discussed.

Proposed system architectures are evaluated with two different evaluation methods commonly used for cloud system architectures. These evaluation methods are explained in detail with a serverless perspective. Each evaluation method defines different evaluation criteria. Thus, each criteria's applicability for serverless architectures is also discussed.

In this study, vector tile map service, raster tile map service, geospatial artificial intelligence (GeoAI), event-driven earthquake notification service and geospatial workflow systems are designed as serverless architectures and applied with a cloud provider's serverless services. Each design is supported by real-world GIS scenarios, role definitions and requirement lists. Each design is applied to a cloud provider's infrastructure by using only serverless services. At the end of each system design section, the proposed design and its application to a cloud provider are evaluated according to the evaluation methods described in the study.

As a part of the last system design, workflow and workflow task definition formats are also defined and explained with validation rules. These definitions are key elements to define the workflows to be used in the proposed geospatial workflow system. Thus, users can easily design and define geospatial analysis workflows to run on serverless architectures.

One important contribution of this study is to provide well-defined and well-evaluated serverless GIS architecture designs to solve various real-world scenarios. Serverless

architectures can reduce resource utilization and the carbon footprint of the systems. Another important contribution is to review serverless services from a geospatial point of view with extensive examples over generic scenarios. Lastly, another remarkable contribution is to present the geospatial analysis workflow system and design the workflow definition models to run complex and long-running geospatial data analyses on serverless architectures.

One of the goals of this thesis study to demonstrate the use of serverless architectures in geospatial applications and to increase serverless technologies awareness in the geospatial community. It is expected that the novel system designs presented in the study will be a reference to similar studies in the future.

1. GİRİŞ

Günümüzde gelişen teknolojiyle birlikte büyük veri (big data), nesnelerin interneti (IoT-Internet of Things), makine öğrenimi (machine learning) gibi yeni kavramlar literatürde daha çok yer almaya başlamıştır. Mekânsal bilgi teknolojileri de bu gelişmelerin içerisinde önemli bir konum kazanmıştır. Örneğin nesnelerin interneti kavramı ile iç içe geçen akıllı şehirler konseptinin bir parçası olan akıllı sensörlerin ürettiği büyük verinin analizi ve görselleştirilmesinde coğrafi bilgi sistemleri ana oyuncu olarak karşımıza çıkmaktadır (Li ve diğ, 2016). Geomatik mühendisliği ile makine öğreniminin etkileşimine örnek olarak da uzaktan algılama ve fotogrametri alanında yapılan makine öğrenimi çalışmaları verilebilir (Döş ve Uysal, 2019; Atik ve diğ, 2022). Bu çalışmalar mekânsal bilişim teknolojilerinin güncel teknolojilerle birlikte gelişerek ilerlediğini gösteren örneklerden bir kaçıdır. Bu değişim aynı zamanda üretilen ve işlenen veri miktarını da önemli ölçüde arttırmıştır. İletişim teknolojilerinin ilerlemesi ve veri kaynaklarının çoğalması üretilen veri miktarının radikal bir şekilde artışına neden olmuştur. Veri depolama maliyetlerinin azalmasıyla birlikte üretilen ham veriler zamanla silinmesine de gerek olmadan saklanabilmektedir. Bu nedenle veri sorgulama için kullanılan klasik yöntemler ile ham ve yapısal olmayan verilerin sorgulanması mümkün olmadığından "büyük veri" çatı kavramı altında yeni sorgulama ve analiz araçları geliştirilmiştir. Bu verilerin içerdiği konum bilgisinin analizinin en optimum şekilde nasıl yapılacağı da mekânsal bilişim topluluğu içinde çözülmesi gereken yeni problemler ortaya koymuştur.

Bulut bilişim teknolojileri yukarıda adı geçen teknolojilerin olgunlaşmasında ve geliştirilmesinde önemli rol oynamıştır. Verinin depolanması, işlenmesi ve sunumu konularında bulut bilişim teknolojileri ile farklı yaklaşımlar ve modeller ortaya çıkmıştır. Veri işleme ve analizi için gereken büyük işlem gücüne büyük bilişim altyapısı yatırımları yapmadan bulut bilişim ile çok kısa sürede ulaşmak mümkün hale gelmiş ve bu da bu konuda çalışan farklı disiplinlerdeki uzmanların sayısını arttırmış ve yeni akademik çalışmaların önünü açmıştır.

Günümüzde mekânsal bilişim teknolojilerinin bulut bilişim ile etkileşimi giderek artmaktadır. Mekânsal bilişim üzerine çalışan birçok akademisyen ve uzman çalışmalarını bulut bilişim altyapılarına taşıyarak bu konuda yeni algoritmalar ve uygulamalar geliştirmektedirler. Bu çalışmalar bulut bilişimin sağladığı çeşitli hizmet modelleri kullanılarak yapılmaktadır. Bulut bilişimin sağladığı hizmet modellerinden biri de sunucusuz hizmet modelidir. Bu model ile bulut bilişim kaynaklarının en verimli şekilde kullanılması mümkün olmaktadır. Bu modelle çalışan kaynaklar kullanıcıdan sunucu yönetimini tamamen soyutlarlar.

Sunucusuz hizmetlerle geliştirilen sistem ve yazılım mimarilerine sunucusuz mimariler denmektedir. Henüz coğrafi bilgi teknolojileri içerisinde sunucusuz mimariler etkin şekilde kullanılmamaktadır (Baldini ve diğ, 2017). Bu çalışmada mevcut coğrafi bilgi teknolojileri uygulamaları geliştirilen sunucusuz mimariye dayalı sistem tasarımlarıyla incelenmiştir. Tez çalışmasında sunulan sistem tasarımları ile coğrafi verilerin saklanması, sunumu ve analizi için hem bilişim altyapısının hem de insan kaynağının kullanımının verimliliğinin arttırılması hedeflenmiştir.

1.1 Tezin Amacı ve Kapsamı

Bu tezin bir amacı sunucusuz mimarileri coğrafi bilgi teknolojileri özelinde incelemek ve bu konuda yeni çözümler sunmaktır. Tez çalışmasında sunulan çözümlerin daha iyi anlaşılabilmesi için sunucusuz kavramı ve sunucusuz mimarideki hizmet modelleri sınıflandırılarak incelenmiştir. Açıklanan hizmet modelleri seçilen iki farklı bulut bilişim sağlayıcısının sunucusuz hizmet modelleri üzerinden incelenmiştir. Bulut bilişim sağlayıcılarının sundukları sunucusuz hizmetlerin mekânsal bilişim özellikleri incelenmiştir.

Tez çalışmasında vektör karo harita servisi, raster karo harita servisi, mekânsal zekâ, olay güdümlü deprem bildirim servisi ve mekânsal analiz iş akışı sistemlerinin sunucusuz mimaride sistem tasarımları sunulmuştur. Her bir tasarım CBS kullanım senaryoları, roller ve gereksinimler açıklanarak desteklenmiştir. Tasarımlar seçilen bir bulut bilişim sağlayıcısı üzerinde uygulanarak açıklanmıştır. Tasarım ve uygulamalar tezde açıklanan değerlendirme yöntemlerine göre değerlendirilmiştir.

Bu tez kapsamında aşağıdaki sorulara cevaplar aranmıştır.

- Sunucusuz kavramı nedir?

- Sunucusuz mimariye dayalı servisler nelerdir ve türleri nedir?
- Bulut bilişim sağlayıcılarının sunduğu sunucusuz hizmetler nelerdir ve hangi mekânsal bilişim özelliklerini sunmaktadırlar?
- Sunucusuz mimariye dayalı sistemler nasıl değerlendirilmelidir?
- Bir raster karo harita servisi sistemi sunucusuz mimari ile nasıl tasarlanır? Bulut bilişim sağlayıcısı üzerindeki sunucusuz servisler üzerinde nasıl uygulanır? Bu tasarım üzerinde OGC (Open Geospatial Consortium) WMTS (Web Map Tile Service) standardı nasıl uygulanır?
- Bir vektör karo harita servisi sistemi sunucusuz mimari ile nasıl tasarlanır? Bulut bilişim sağlayıcısı üzerindeki sunucusuz servisler üzerinde nasıl uygulanır? Bu tasarım üzerinde MVT (Mapbox Vector Tiles) standardı nasıl uygulanır?
- Bir mekânsal zekâ sistemi sunucusuz mimari ile nasıl tasarlanır? Bulut bilişim sağlayıcısı üzerindeki sunucusuz servisler üzerinde nasıl uygulanır? Bu tasarım üzerinde OGC WPS (Web Processing Service) standardı nasıl uygulanır?
- Bir olay güdümlü mekânsal veri işleme servisi sistemi sunucusuz mimari ile nasıl tasarlanır? Bulut bilişim sağlayıcısı üzerindeki sunucusuz servisler üzerinde nasıl uygulanır? Bu tasarım üzerinde OGC API (Application Programming Interface) – Processes (İş-süreçleri) standardı nasıl uygulanır?
- Bir mekânsal analiz iş akışı sistemi sunucusuz mimari ile nasıl tasarlanır? Bulut bilişim sağlayıcısı üzerindeki sunucusuz servisler üzerinde nasıl uygulanır?
- Bir mekânsal analiz iş akışı sisteminde iş akışları ve görev tanımları nasıl modellenmelidir?

1.2 Tez İçeriği ve Kurgu

Bu tez sunucusuz kavramının detaylı bir şekilde açıklanması ve çeşitli mekânsal bilişim servislerinin tasarımlarının sunucusuz mimarilerle tasarlanarak uygulamalarla incelenmesinden oluşmaktadır (Şekil 1.1).

Şekil 1.1 : Tez kapsamında uygulanan araştırma metodolojisi.

Birinci bölümde (Giriş) tez konusu, amacı ve içeriği hakkında ayrıntılı bilgi verilmiştir. Ayrıca literatür incelemesi yapılarak benzer çalışmalar incelenmiş ve bu çalışmalar değerlendirilmiştir.

İkinci bölümde (Sunucusuz Kavramı ve Sunucusuz Hizmet Türleri) sunucusuz kavramı açıklanmıştır. Ardından sunucusuz mimarideki hizmet türleri sınıflandırılarak incelenmiş ve açıklanmıştır. Bu bölümde bir sonraki bölümlerde kullanılan sunucusuz hizmetler hakkında kavramların anlaşılmasını kolaylaştıran ayrıntılı bilgiler verilmektedir.

Üçüncü bölümde (Bulut Bilişim Sağlayıcılarının Sunucusuz Hizmetleri) ikinci bölümde açıklanan sunucusuz hizmet türlerinin seçilen iki farklı bulut bilişim sağlayıcısı üzerindeki sunumları mekânsal bilişim özellikleriyle birlikte incelenmiştir.

Dördüncü bölümde (Sunucusuz Coğrafi Bilgi Sistemi Yazılımı Tasarımları ve Uygulamaları) sunucusuz hizmet türleri ve çeşitli Coğrafi Bilgi Sistemi standartları kullanılarak sunucusuz sistem mimarileri tasarlanarak uygulanmıştır. Sunucusuz sistem tasarımları verilmeden önce tasarımların değerlendirmesinde kullanılan yöntemler verilerek açıklanmıştır. Her tasarım bir CBS senaryosu, gereksinimler ve kullanıcı rolleri verilerek desteklenmiştir. Bölümün son kısmında verilen tasarım (Sunucusuz Mekânsal Analiz İş Akışı Sistemi Tasarımı, Uygulaması ve Değerlendirmesi) ile yeni bir iş akışı ve iş akışı görevi tanımı modeli ortaya konulmuştur. Geliştirilen bu tanımlara göre örnek mekânsal analiz iş akışları tasarlanmıştır.

Beşinci bölümde (Sonuç ve Öneriler) tezde yer alan tüm bölümler bütünsel olarak ele alınarak değerlendirilmiştir. Ayrıca sunucusuz yazılım mimarileri ile mekânsal bilişim konusunda ileride yapılabilecek çalışmalara ilişkin yol haritası da geliştirilerek sunulmuştur.

1.3 Literatür İncelemesi

Baldini ve diğ. (2017) yayınladıkları çalışmada sunucusuz kavramını tanımlamış ve bu kavrama dayalı geliştirilen yazılım mimarileri ve bulut bilişim servislerinin günümüzdeki kullanım alanlarını incelemişlerdir. Bununla birlikte bu sunucusuz yazılım ve servislerin karakteristik özelliklerini tanımlamışlardır. Çalışmada ayrıca sunucusuz mimarilerin avantajları ve dezavantajları da açıklanarak teknik zorluklar ve halen çözülmeyi bekleyen problemlerden de bahsedilmiştir.

Bebortta ve diğ. (2020) yaptıkları çalışmada sunucusuz mimarilerle büyük mekânsal verilerin analizini farklı ve iyi bilinen sunucusuz bilişim servis sağlayıcısı platformları üzerinde incelemiştir. Sunucusuz mimariler performans, güvenilirlik, ölçeklenebilirlik, güvenlik ve hız parametreleri ile ele alarak diğer mimarilerle karşılaştırmışlardır. Çalışmada sundukları çerçeve yapı ile mekânsal verinin analizinin sunucusuz hizmetlerle yaparak elde ettikleri sonuçları paylaşmışlardır. Sunucusuz

mimarilerin coğrafi bilgi teknolojilerinde daha baskın olarak kullanılacağını değerlendirmiştirler.

Jain ve diğ. (2020) çalışmalarında, AWS (Amazon Web Services) bulut bilişim sağlayıcısı üzerinde sunulan sunucusuz hesaplama hizmet modellerinden olan HoF (Hizmet olarak Fonksiyon) ve HoK (Hizmet olarak Konteyner) modelleri ile HoP (Hizmet olarak Platform) modelini karşılaştırmak için örnek bir uygulamayı üç modeldeki servisle çalıştırarak performans analizleri yapmışlardır. Sonuç olarak bulut bilişim kullanacak mühendislerin daha fazla bilgi işlem gücü ihtiyacı olduğunda ve özelleştirilmiş bir ortamda uygulama çalıştırmaları gerektiğinde HoK modelinin bu ihtiyaçları karşılayabileceği belirtilmiştir.

Anand ve diğ. (2019) yayınladıkları çalışmada geliştirdikleri gerçek zamanlı konum takip sistemini sunucusuz mimari ile nasıl uyguladıklarını açıklamışlardır. Sundukları sistem mimarisini geliştirdikleri bir takip cihazı donanımı ve yazılım ile bir bulut bilişim sağlayıcısının sunucusuz hizmetleri üzerinde uygulayarak incelemişlerdir. Uygulamada yaşadıkları teknik zorlukları paylaşarak bu zorlukların sunucusuz bulut bilişim servisleri ile çözüldüğünü belirtmişlerdir.

Mete ve Yomralıoğlu (2021a) çalışmalarında bir arazi değerlemesi uygulamasını sunucusuz mimaride farklı sunucusuz hizmet türlerini kullanarak sunmuşlardır. Bu çalışma yazarların daha önce sundukları bulut bilişim tabanlı sistem mimarisinin (Mete ve Yomralıoğlu, 2021b) sunucusuz mimaride yeniden tasarlanarak nasıl oluşturulabileceğini de göstermektedir.

Pakdil ve Çelik (2021a) yayınladıkları çalışmada sunucusuz mimarilerle mekânsal veri analizleri için iş akışları çalıştırabilecek bir sistem tasarımı sunmuşturlar. Bu tasarımda veri işleme ve web servislerin yayını için HoF ve HoK modelindeki servisler birlikte kullanılmıştır. Çalışmada OGC API Processes standardının sunucusuz mimarilerle çalışabilirliği de literatürde ilk kez incelenmiştir.

Kim ve Lin (2018) çalışmalarında AWS bulut bilişim sağlayıcısının sunduğu HoF modelindeki AWS Lambda isimli servisle geliştirdikleri Flint ismindeki sistemi sunmuşlardır. Bu çalışmada Java ve Python programlama dillerinin AWS Lambda üzerindeki soğuk başlatma sürelerinin farklılığı vurgulanarak Python dilinin daha hızlı başladığı belirtilmiştir. Sonuç kısmında büyük veri analizlerinin ortaya koydukları

Flint gibi sistemlerin tasarımlarıyla sunucusuz mimarilerde mümkün olduğu vurgulanmıştır.

Malawski ve diğ. (2017) çalışmalarında sunucusuz mimarileri bilimsel analiz süreçlerine uygulamışlardır. Açık kaynak kodlu HyperFlow yazılımını HoF modelinde sunucusuz hesaplama hizmeti olan AWS Lambda ve Google Cloud Functions servisleri üzerinde çalıştırarak sonuçları incelemişlerdir. Sonuç olarak yüksek hesaplama gücü gerektiren bilimsel hesaplamalar için HoF modelinin uygun olmadığı ancak yüksek çıktılı bilimsel hesaplamalara daha uygun olduğu belirtilmiştir. Diğer yandan bilimsel analiz uygulamalarının HoF modelinde maksimum çalışma süresi ve sunulan geçici depolama alanının limitlerinin sorun olabileceği de vurgulanmıştır.

Lee ve diğ. (2018) çalışmalarında AWS, Microsoft Azure, Google Cloud ve IBM bulut bilişim sağlayıcıları üzerinde HoF modelindeki sunucusuz hesaplama servislerini; depolama alanı performansı, işlem gücü performansı, koşut zamanlılığı (concurrency), ağ kullanımı, esneklik ve uygulama yayınlama (deployment) parametrelerine göre incelemiştir. Sonuç olarak sunucusuz HoF modelinin yüksek hesaplama ve veri okuma gücü isteyen uygulamalar için uygun olmadığı belirtilmiştir. Bununla birlikte çalışmanın yapıldığı tarihlerde denenen servislerin henüz tam olarak olgunlaşmadığı da vurgulanmıştır.

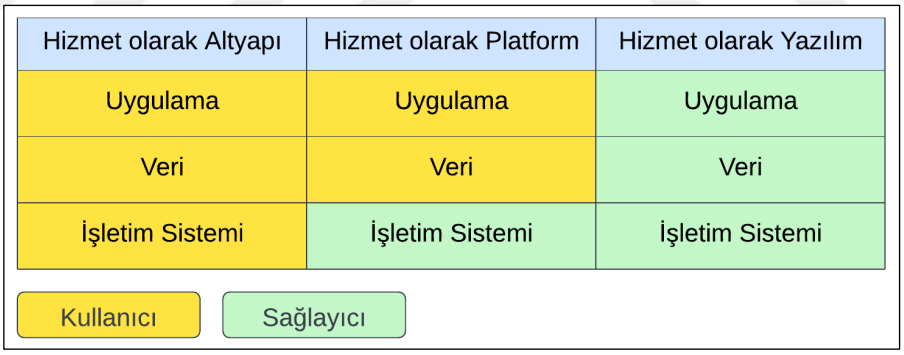
Ji ve diğ. (2012) çalışmalarında bulut bilişimi kullanarak mekânsal veri analizi iş akışının çalıştırılması için sundukları sistem mimarisinin uygulanabilirliğini değerlendirmiştirler. Çalışmanın sonuç kısmında ölçeklenebilirliğin bulut bilişim üzerinde bir avantaj olduğu ve büyük iş akışlarının çalıştırılmasında katkı sağladığı belirtilmiştir.

Krämer ve diğ. (2021) çalışmalarında mikro servis mimarisine uygun bir iş akış yönetimi sistem tasarımı geliştirmişlerdir. Bu tasarımda iş akışlarını tanımlamak için yeni bir iş akış tanımlama modeli de geliştirilmiştir. Böylece büyük veri analizi iş akışlarının kolayca tasarlanabilir ve iş akışı tasarımlarının kolayca okunabilir olması hedeflenmiştir.

Bu tez çalışmasında da sunulan mekânsal analiz iş akışı sisteminde de özgün iş akışı tanımlama modelleri benzer kaygılarla bu tez çalışmasındaki sisteme özel olarak geliştirilmiştir.

2. SUNUCUSUZ KAVRAMI VE SUNUCUSUZ HİZMET TÜRLERİ

Sunucusuz kavramı incelenmeden önce bulut bilişimdeki diğer hizmet modellerine bakmak gerekmektedir. Böylece sunucusuz kavramı diğer hizmet türleri arasından ayrıştırılarak konumlandırılabilir. Bulut bilişim teknolojileri ortaya çıktığı günden bu yana farklı hizmet modelleri ile karşımıza çıkmıştır. Tüm bu modellerin ortaya çıkmasındaki önemli motivasyon ise ölçeklenebilirliğin daha kolay bir şekilde sağlanmasıdır. Ayrıca mikro servis mimarisi gibi yeni uygulama mimarilerinin de daha yaygın kullanılması bulut bilişimdeki hizmet modellerinin farklılaşmasında yönlendirici olmuştur. Her hizmet modelinin diğerine göre farklılaşmasının en temel nedeni altyapı yönetiminin hangi derecede soyutlaştırıldığına dayandırılmaktadır (Şekil 2.1). Bu yönden incelendiğinde bugüne kadar sunulan üç hizmet modeli hakkında bilgi vermek sunucusuz hizmet modelinin anlaşılmasında katkı sağlayacaktır.



Şekil 2.1 : Bulut bilişim hizmet türlerinin yönetimsel farklılıkları.

Hizmet olarak Altyapı (HoA, Infrastructure as a Service) modelinde kullanıcı fiziksel altyapı üzerinde kurulu işletim sistemi, veri katmanı ve uygulama katmanını yönetmekle sorumludur. Sağlayıcı ise tüm bunların çalışmasını sağlayacak fiziksel sunucu bilgisayarı ve ağ donanımlarını yönetmekten sorumludur.

Hizmet olarak Platform (HoP, Platform as a Service) modelinde HoA'dan farklı olarak işletim sisteminin yönetimi de hizmet sağlayıcı tarafından yapılır. Bu nedenle kullanıcı sadece uygulama ve veri katmanlarını yönetir.

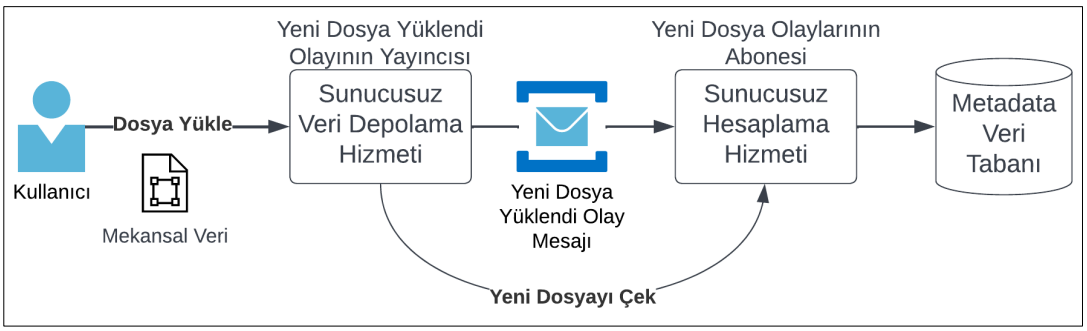
Son olarak Hizmet olarak Yazılım (HoY, Software as a Service) modelinde kullanıcı sadece hizmet sağlayıcının sunduğu yazılımdan belirlenen yetkiler çerçevesinde hizmet alır. Uygulamanın performansından, sağlığından, veri güvenliğinden ve bakımından hizmet sağlayıcı sorumludur.

Sunucusuz kavramı ise bir sunucunun yönetilme ihtiyacı olmadan sunucu uygulamalarının çalıştırılmasının sağlanması olarak açıklanabilir. Bu paradigmaya verilen isim ortamda bir sunucu olmadığını düşündürse de aslında sunucu yönetimini kullanıcıdan tamamen soyutlayarak bu operasyonun hizmet sağlayıcı tarafından sağlanmasına karşılık gelmektedir.

Sunucusuz hizmetlerin farklı türleri olsa da paylaştıkları ortak özellikler bulunmaktadır. Bu ortak özellikler şunlardır;

1. Olay güdümlü (event-driven) olmaları
2. Otomatik ölçeklenebilir (auto-scaleable) olmaları
3. Hata toleranslı (fault-tolerance) olmaları

Olay güdümlü olmaları sayesinde sunucusuz uygulamalar çalışmaya başlamak için bir olayın tetiklemesine ihtiyaç duyarlar (Lee ve diğ, 2018). Örneğin veri depolama alanına yeni kaydedilen bir dosyanın bilgisi hesaplama hizmetindeki bu olayı dinleyen uygulamayı tetikleyebilir. Bu senaryo bir mekânsal veri dosyasının (Shapefile, GeoPackage, File Geodatabase vb.) sunucusuz veri depolama alanına yüklendiğinde meta verisinin otomatik olarak okunarak veri tabanına yazılması istendiğinde uygulanabilir (Şekil 2.2). Meta veriyi okuyacak ve veri tabanına yazacak yazılım her yeni dosyada tetiklenerek çalıştırılır.



Şekil 2.2 : Sunucusuz hizmetlerin birbirlerini olay üzerinden tetiklemeleri.

Sunucusuz hizmetlerin otomatik ölçeklenebilir olmaları sayesinde kaynak tüketim optimizasyonu sağlanmaktadır. Kullanıcılar önceden bir kaynak provizyonu

yapmasına gerek kalmaz. Otomatik ölçeklendirme diğer bulut bilişim hizmet türlerinde de sunulmaktadır fakat sunucusuz hizmet türlerinde otomatik ölçeklendirme ile sıfır sunucu sayısından ihtiyaca göre artabilirken diğer hizmet türlerinde en az bir çalışan sunucu her zaman bulunmaktadır (Castro ve diğ, 2019).

Sunucusuz hizmetlerde çalışma anında herhangi bir hata olması durumunda süreç yeniden belirli sayıda başarılı olana kadar tekrar ettirilir (Jangda ve diğ, 2019). Bu nedenle sunucusuz mimari için geliştirilen uygulamaların bu duruma uygun olarak geliştirilmesi beklenmektedir. Örneğin daha önceki çalışmanın ortasında hata veren işlemin o ana kadar veri tabanına yazdığı veriyi bir sonraki tekrar çalışmada tekrar yazmaması için uygulama geliştirilirken bu senaryo dikkate alınmalıdır.

Sunucusuz hizmetler iki farklı türde sınıflandırılabilir. Bunlar veri depolama ve hesaplama hizmetleri olarak sınıflandırılmaktadır.

Çizelge 2.1 : Sunucusuz hizmet türlerinin sınıflandırılması.

Veri Depolama	Yapısal (structured) ve ilişkisel (relational) veri depolama
	Yapısal olmayan (unstructured) veri depolama
Hesaplama	Fonksiyon (Hizmet olarak Fonksiyon)
	Konteyner (Hizmet olarak Konteyner)

Veri depolama türündeki hizmetler yapısal ve yapısal olmayan veri türlerine göre iki alt grupta incelenmektedir (Ruparathna, 2019). Hesaplama türündeki hizmetler de çalışma ortamlarına göre fonksiyon ve konteyner olarak iki alt grupta incelenmektedir (Schachar, 2019; Chowhan, 2018).

2.1 Veri Depolama Hizmetleri

Bulut bilişimde HoP modelindeki veri depolama sistemleri en az bir tane aktif veri depolama sunucusunu devamlı olarak çalıştıracak şekilde tasarlanmışlardır. Ayrıca bu servislerin sağlıklı çalışabilmesi için doğru şekilde yapılandırılmaları da gerekmektedir. Aksi halde yoğun istek altında istenen performansı veremeyebilir veya tamamen durabilirler. Ek olarak bakım ve yedekleme takibinin de kullanıcı tarafından yapılması gerekmektedir. Sunucusuz bir sistem tasarımı eğer HoP modelindeki veri depolama hizmeti bileşenini kullanırsa bu sistemin zayıf noktası olarak karşımıza

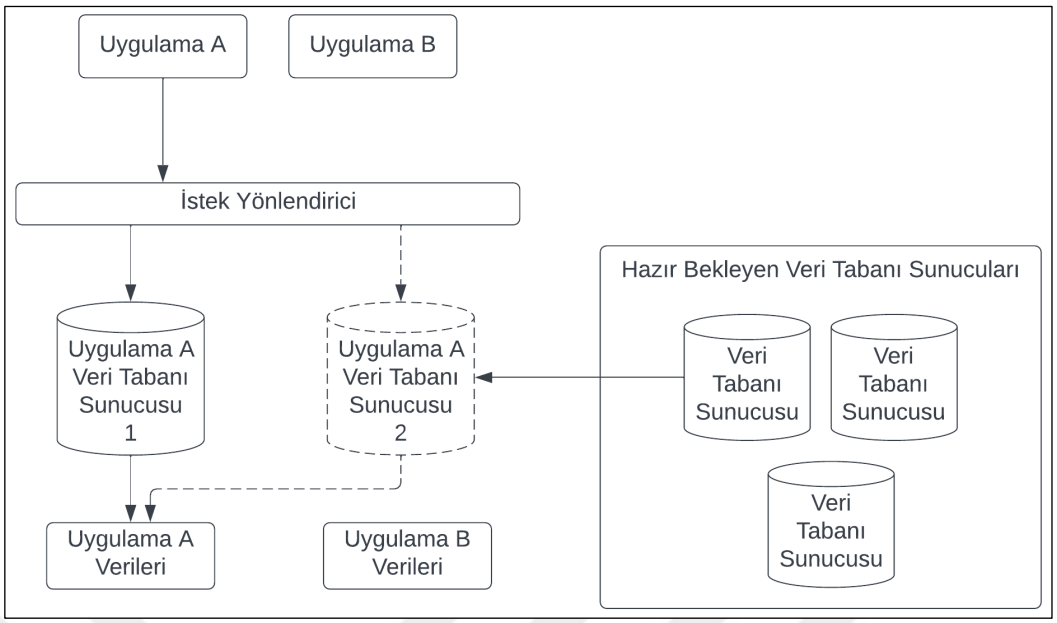
çıkacaktır. Bu nedenle sunucusuz sistemlerde veri depolama bileşeninin de sunucusuz hizmet modelinde olması tutarlılık sağlayacaktır.

Sunucusuz veri depolama hizmetleri yapısal ve yapısal olmayan verilere göre iki ana grupta incelenmektedir.

2.1.1 Yapısal ve ilişkisel veri depolama hizmetleri

Yapısal ve ilişkisel veri depolama hizmetleri genel olarak belirli bir şemada ve kuralla verileri saklayan ve SQL (Structured Query Language) veya diğer özel sorgulama dilleri ile sorgulama desteği veren hizmetlerden oluşmaktadır. Yapısal ve ilişkisel veri depolama servisleri halihazırdaki veri tabanı yazılımlarının bulut bilişim sağlayıcısı tarafından özelleştirilmesi ya da bulut bilişim sağlayıcısının geliştirdiği özel veri tabanı yazılımları üzerinden sunulmaktadır. Yaygın olarak kullanılan bir veri tabanı sunucusu yazılımının sunucusuz hizmet modeli üzerinden sunulması kullanıcılar için sunucusuz modele geçişte kolaylık sağlamaktadır. Örneğin geleneksel sistemlerde çalışan bir CBS yazılımının kullandığı açık kaynak kodlu PostgreSQL veri tabanı sunucusu yazılımını değiştirmeden sunucusuz hizmet modelinde kullanmaya devam edebilmesi sunucusuz modele göçü kolaylaştıracaktır.

İlişkisel veri tabanı sunucularının sunucusuz hizmet modeli ile sunumu için bir istek yönlendirici yazılımın veri tabanı istemcisi ile veri tabanı sunucusu arasındaki iletişimi yönetmesi gerekir. Böylece gelen isteklerin çokluğuna göre yeni veri tabanı sunucularını çalıştırabilir veya tam tersi durumda çalışan veri tabanı sunucusunu yeniden kullanılmak üzere beklemeye alabilir (Solanki, 2021). Böyle bir durumda bu hizmeti kullanan uygulamalar arasında veri tabanı sunucusu havuzunda hazırda kullanılmayı bekleyen veri tabanı sunucuları paylaştırılır. Bu da veri tabanı sunucusu kaynaklarının verimli şekilde kullanılması ile kaynak optimizasyonunu sağlar. Şekil 2.3'te gösterildiği gibi Uygulama A'dan gelen istekler arttığı için "İstek Yönlendirici" yazılım hazırda bekleyen bir veri tabanı sunucusunu Uygulama A'nın verilerine bağlayarak yükü iki sunucuya paylaştırarak otomatik ölçeklendirmeyi sağlamış olur. Bununla birlikte Uygulama B ise veri tabanını kullanmadığı için "İstek Yönlendirici" tarafından Uygulama B için bir veri tabanı sunucusu çalıştırılmaz. Ancak yine de Uygulama B'nin verileri her an tekrar kullanılmak üzere saklanır.



Şekil 2.3 : Sunucusuz ilişkisel veri tabanı hizmetinin çalışma anında ölçeklendirilmesi (Solanki, 2021).

Sunucusuz yapısal ve ilişkisel veri tabanlarının bir diğer özelliği ise bulut bilişim sağlayıcısının hizmet verdiği birden fazla coğrafi bölgeye kopyalanıp çoğaltılarak tek bir veri tabanı sunucusu gibi çalışabilmeleridir. Böylece birçok ülke veya kıtada kullanılması beklenen bir CBS uygulamasına veri tabanı kaynaklı gecikme problemini önleyeceğinden erişim daha hızlı olacaktır.

Sunucusuz yapısal veri tabanlarının verileri sorgulanırken kullanılan sorgulama dili veri tabanı teknolojisine göre değişmektedir. Yapısal veri tabanına sorgulama yapılırken HTTP (Hypertext Transfer Protocol) protokolü üzerinden sunulan web servisleri kullanılmaktadır. Sorgulama yapılırken genel olarak SQL dilinin kullanılmadığı bu türdeki veri tabanlarına literatürde "NoSQL" veri tabanı da denilmektedir. Uygulama geliştiriciler yapısal veri tabanı ile olan sorgulama ve yönetimsel işlemler için bulut bilişim sağlayıcısının veya veri tabanı yazılımının kendisinin sunduğu özel yazılım kütüphanelerini kullanmaktadırlar.

Sunucusuz yapısal veri tabanları sakladıkları veri türlerine göre günümüzde dört ana grupta incelenmektedir (Hecht ve Jablonski, 2011).

- Belge veri tabanları
- Sütun ailesi veri tabanları
- Anahtar-değer (key-value) veri tabanları

- Çizge (graph) veri tabanları

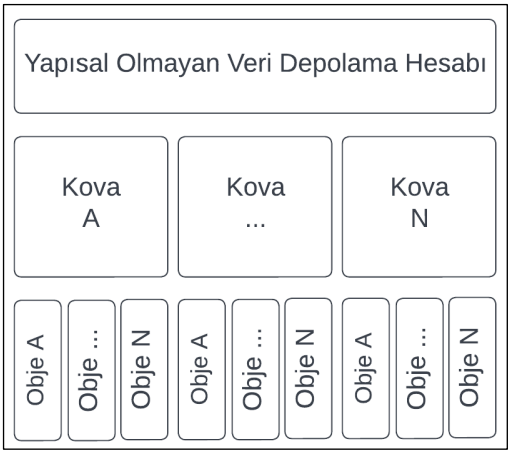
Her bir veri türü kullanım senaryosuna bağlı olarak bir diğerine göre avantajlar veya dezavantajlar içermektedir. Eğer uygulama için doğru veri türü seçimi yapılmazsa hem tüketilen veri tabanı kaynakları artacaktır hem de uygulama performansı düşük olacaktır (Kleppmann, 2017).

Yapısal veri tabanı sunucusu yazılımı mimarileri bulut bilişim teknolojileri üzerinde çalışmak üzere veya dağıtık sistemlerle çalışmak üzere tasarlandığından sunucusuz bir hizmet olarak sunumu daha geleneksel yaklaşımla tasarlanan ilişkisel veri tabanı sunucusu yazılımlarına göre daha kolay olmaktadır (Grolinger ve diğ, 2013). Baralis ve diğ. (2017) yaptıkları çalışmada ilişkisel ve yapısal iki farklı veri tabanı hizmetinin bir bulut bilişim sağlayıcısı üzerinde performanslarını karşılaştırmıştırlar. Elde ettikleri sonuca göre coğrafi verilerin sorgulanmasında yapısal veri tabanı hizmetinin bulut bilişim altyapısı üzerinde daha iyi sonuç vermekte olduğu belirtilmiştir.

Yapısal ve ilişkisel veri depolama hizmetlerinin birçoğu eklentiler veya dahili mekânsal veri işlemcileriyle mekânsal sorgulara ve veri türlerine destek vermektedirler.

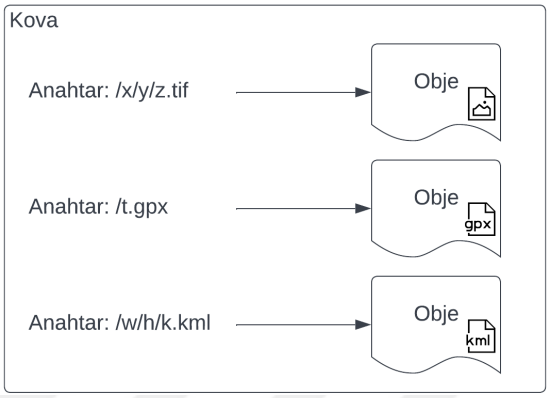
2.1.2 Yapısal olmayan veri depolama hizmetleri

Yapısal olmayan veri depolama hizmetleri verileri obje olarak saklamaktadır. Bu objelere BLOB (Binary Long Objects) da denmektedir. İkili (binary) biçemde (formatta) olan her şey obje şeklinde saklanabilir (Amirian ve diğ, 2014). Bir yapısal olmayana veri depolama hizmeti hesabına ait mantıksal organizasyon Şekil 2.4'te gösterilmektedir.



Şekil 2.4 : Yapısal olmayan veri depolama hizmetinin veri saklama yapısı.

Objelerin bir arada gruplanarak depolandığı varlık kümeleri kova (bucket) veya konteyner (container) terimi ile ifade edilmektedir (Şekil 2.5). Bu çalışmada varlık kümeleri için ‘kova’ teriminin kullanımı, hesaplama teknolojileri için kullanılan ‘konteyner’ terimi ile kavramsal olarak karıştırılmaması için tercih edilmiştir. Bir kova içerisindeki objelere erişim bir anahtar değer üzerinden olmaktadır. Bir hesap içindeki kova isimleri ve bir kova içerisindeki anahtar değerler benzersiz olmaktadır.



Şekil 2.5 : Yapısal olmayan depolamada kova yapısı.

Yapısal olmayan verileri ortak özelliklerine göre bölümlendirerek çok sayıda obje saklayan kova içerisindeki erişim kolaylaştırılabilir. Anahtar değerinde “/” sembolü kullanılarak bölümlendirme yapılmaktadır (Kapadia ve diğ, 2015). Örneğin aynı gün ve aya ait veriler saklanırken anahtar değerin başına “/05/21/” eklenirse Mayıs ayının 21. gününe ait veriler gruplandırılabilir. Bu sayede 21 Mayıs verilerine tüm verilerin kontrol edilmesine gerek kalmadan hızlıca erişilmiş olur. Her ne kadar bir kovanın alabileceği en fazla veri miktarı bulut sağlayıcının kurallarına göre değişebilir olsa da veriye erişim hızı kovanın büyüklüğüne göre değişmemektedir

Kovalar için bulut sağlayıcıları tarafından koyulan limitler bulunabilmektedir. Örneğin bir bulut bilişim kullanıcısı hesabında en fazla yüz adet kova oluşturabilmesine izin verilebilir.

Saklanan objelere erişim yapısal olmayan veri depolama hizmeti üzerinden sunulan REST (Representational State Transfer) API’lar aracılığı ile olmaktadır. Bununla birlikte uygulama geliştiricilerin yazılım geliştirmesini kolaylaştırmak için farklı programlama dilleri için SDK’ler (Software Development Kit) de sunulmaktadır.

Saklanan objelerin güvenliği ise dahili izin kontrol yapılarıyla sağlanmaktadır. Objeler herkese erişime açık olarak saklanabilirken sadece bir kullanıcı, kullanıcı rolü veya kullanıcı grubu için de erişim izni verilerek saklanabilmektedir. Güvenli erişim için

modern web teknolojilerinde yaygın olarak kullanılan jeton (token) tabanlı yetkilendirme teknolojileri kullanılmaktadır. Bununla birlikte istenen geçici adresler de üretilerek sistemin dışındaki istemcilere veya kullanıcılara belirli süreler için erişim verilebilir.

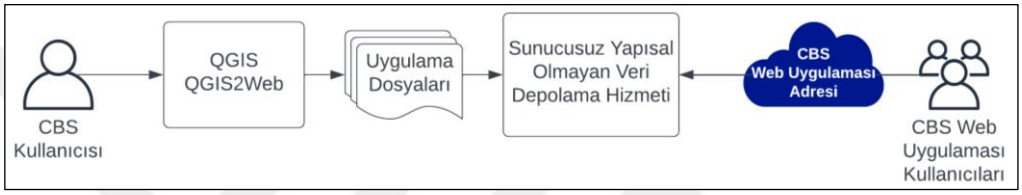
Yapısal olmayan veri depolama hizmetlerinde saklanan veriler eğer kendi içinde tutarlı bir şekilde belli bir şemaya sahipse sunucusuz veri analiz servisleri sorgulama ve kümeleme (aggregation) işlemlerini bu veriler üzerinde yapabilmektedir. Bu servisler ise Hizmet olarak Sorgu (HoS – Query as a Service) olarak adlandırılmaktadır (Marroquín ve diğ, 2018). Öncelikle verilerin şeması kullanılan sunucusuz veri analiz hizmetine tanıtılması gerekmektedir. Bu amaçla CSV (Comma-Separated Values), JSON (JavaScript Object Notation) ve Apache Parquet gibi şema tutan biçemler objeler saklanırken yaygın olarak tercih edilmektedir.

Mekânsal indeksleme (spatial indexing) yapısal olmayan verilerin uygulama geliştirme ara yüzleri üzerinden hızlıca sorgulanmasını mümkün kılmaktadır (Moten, 2019). İndekslemenin tutulması ve sorgulanması için yapısal sunucusuz veri depolama hizmetleri kullanılabilmektedir. Örneğin bu sayede büyük bir coğrafi alana ait raster karo verileri içinden bir coğrafi alana düşen raster karolar kolayca bulunabilmekte ve erişilebilmektedir.

Yapısal olmayan veriler üzerindeki veya depolama alanındaki değişiklikler birer olay (event) üretmektedir. Bu üretilen olayları dinlemek ve buna göre işlem yapmak üzere aynı bulut bilişim sağlayıcısı üzerinde başka diğer sunucusuz hizmetler kullanılabilir. Böylece bir yapısal olmayan veri depolama hizmeti üzerinde bir obje değişikliği veya yeni bir objenin eklenmesi halinde hangi işlemlerin yapılacağı ve hangi hizmetlerin bu duruma karşı aksiyon alacağı bir iş akışı olarak tasarlanabilmektedir (Sampé ve diğ, 2017). Örneğin yeni yüklenen bir CAD (Computer-aided Design) dosyasının oluşturacağı olay bir sunucusuz hesaplama hizmeti üstündeki uygulamayı tetikleyebilir. Bu uygulama da yüklenen dosyayı CBS veri tabanına uygun bir şekilde dönüştürerek kaydedebilir (Şekil 2.6).

Şekil 2.6 : Yapısal olmayan depolama hizmetindeki bir olayın hesaplama hizmetini tetiklemesi örneği.

Son olarak sunucusuz yapısal olmayan veri depolama hizmeti eğer destekliyorsa statik internet sayfaları için bir web sunucusu olarak kullanılabilir. Bunun için yüklenen dosyaların "herkese erişime açık" olarak yüklenmesi ve hizmet üzerindeki ilgili ayarların yapılması gerekmektedir (Nadon, 2017; Soueidi, 2015). Örneğin QGIS isimli masaüstü CBS yazılımı ve QGIS2Web eklentisi kullanılarak oluşturulacak bir statik web CBS uygulaması yapısal olmayan veri depolama hizmetleri üzerine yüklenerek doğrudan yayınlanabilir. Böylece Şekil 2.7'de gösterildiği gibi bir CBS kullanıcısı herhangi bir sunucu kurulumu yapmadan ve ileri bulut bilişim sistem yönetimi konularını hâkim olmadan da yüksek ölçeklenebilir bir CBS web uygulamasını kolaylıkla yayına alabilmektedir (Gandhi, 2021).



Şekil 2.7 : Yapısal olmayan depolama hizmetindeki statik bir CBS web uygulamasının yayınlanması örneği.

2.2 Hesaplama Hizmetleri

Bulut bilişimin beraberinde getirdiği en büyük avantaj kullanıcılardan fiziksel altyapı yönetimini soyutlamak olsa da yine de platform üzerindeki sanal kaynakların yönetimini kullanıcılardan istemektedir (Jonas ve diğ, 2019). Bulut bilişim üzerinde HoA (Hizmet olarak Altyapı) ve HoP (Hizmet olarak Platform) türündeki hizmet modelleri sanal makinelerin, sanal ağ adaptörlerinin (virtual network adapter), sanal sabit sürücülerin (virtual disk) veya sunucu yazılımlarının ayarlarını kullanıcılardan yapmalarını isterler. Bununla birlikte bu sanal kaynakların yedeklenmesi ve bakımının da kullanıcı tarafından takip edilmesi beklenmektedir. Bir uygulamanın bulut bilişim altyapısı üzerinde sağlıklı ve ölçeklenebilir yayın yapması için gerekenler şöyle sıralanabilir (Jonas ve diğ, 2019);

1. Fazladan hazır yedek sunucu bulundurmak. Bir sunucu çalışmadığında yedek sunucu üzerinden uygulama hizmet vermeye devam edebilir.
2. Fazladan birkaç farklı coğrafi bölgede sunucu bulundurmak. Bir bölgedeki tüm veri merkezleri hizmet dışı kaldığında uygulama başka bir coğrafi bölgeden veri merkezinden hizmet vermeye devam edebilir.

3. Trafik yükü dağıtım sisteminin kurulması. Gelen istekler en az yük altındaki sunucuya iletilir.
4. Otomatik ölçeklendirmenin kurulması. Uygulamaya gelen isteklerin çokluğu veya azlığına göre gerektiğinde uygulama ek sunucular üzerinde birlikte çalışarak performans kaybı olmadan çalışmaya devam edebilir. Tam tersi durumda da sunucu eksiltilebilir.
5. İzleme (monitoring) sisteminin kurulması. Uygulama veya sistemin sağlığı takip edilebilir.
6. Günlük\Kayıt (logging) sisteminin kurulması. Uygulama içerisindeki hatalar ve performans sorunları takip edilebilir.
7. İşletim sistemlerinin ve destek yazılımlarının güncel tutulması. Böylece güvenlik tehditlerine karşı uygulama ve sistem korunmuş olur.
8. Yeni sunuculara uygulamanın otomatik yüklenebilmesi: Otomatik ölçeklendirme sırasında uygulama elle müdahale gerekmeden kendiliğinden eklenen yeni sunucu üzerinde çalışmaya başlayabilir.

Tüm bu ayarlamalar ve kurulumların yapılması ve bakım prosedürlerinin belirlenmesinden sonra bulut bilişim kullanıcısı geliştirdiği uygulamayı platforma yükleyerek sağlıklı ve performanslı bir şekilde çalışmasını sağlayabilmektedir. Bu sorumlulukların kullanıcı üzerine yüklenmesi nedeniyle bulut bilişim kullanıcılarının çalışılan platform ve kullanılan teknolojiler hakkında ileri düzey bilgi sahibi olması gerekmektedir.

Sunucusuz hesaplama hizmetleri fonksiyon ve konteyner tabanlı altyapı türüne göre iki modelle “Hizmet olarak Fonksiyon” ve “Hizmet olarak Konteyner” olarak incelenmektedir (Schachar, 2019; Chowhan, 2018). Her iki grubun da ortak özelliğin yukarıda verilen sekiz maddenin yapılandırılmasını kullanıcıdan soyutlamasıdır. Böylece kullanıcı sadece çalıştırmak istediği uygulama veya konteynere odaklanabilmektedir. Geri kalan altyapı yönetiminin sorumluluğu bulut bilişim sağlayıcı üzerinde olmaktadır.

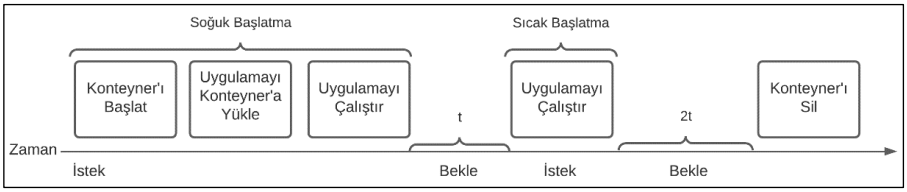
HoK modeli HoF modeline göre uygulama geliştiriciler için daha fazla özgür bir ortam sağlandığından HoF üzerindeki kısıtlamalarından etkilenen uygulamalar için HoK modeli alternatif olmaktadır (Kiener ve diğ, 2021).

2.2.1 Hizmet olarak fonksiyon

Sunucusuz hesaplama modellerinden biri olan HoF modeli belli bir amaç için geliştirilmiş uygulamayı çalıştırmak için kullanılmaktadırlar. Bazı durumlarda eğer bulut bilişim sağlayıcı da izin veriyorsa daha karmaşık birden fazla amaç için geliştirilmiş uygulamaları da -örneğin web API'ları gibi- çalıştırabilmektedirler.

Bulut bilişim sağlayıcısı HoF servisi üzerinde çalışabilen programlama dillerini ve o dillerin hangi sürümlerinin desteklendiğini yayınladıkları dokümantasyonla açıklamaktadır. Uygulama bu dillerden birini kullanarak geliştirilebilir. HoF servisinin kullanıcısı uygulamasını bu kıstaslara göre hazırlaması gerekmektedir. Bununla birlikte uygulama geliştiricinin uygulamada kullanılması gereken bir çatkı (framework) kütüphanesi veya diğer kütüphaneler olabilir. Uygulama geliştiricinin bu kütüphanelerin HoF servisleri ile uyumluğuna dikkat etmesi gerekmektedir ve gerekiyorsa ek geliştirmelerle bunları uyumlu hale getirmelidir.

Temelde birer konteyner yönetimi olarak çalışan HoF mimarileri, özel olarak geliştirilen konteynerlerin sisteme yüklenen kod parçalarını istek alındığı anda olabilecek en hızlı şekilde çalıştırılması üzerine tasarlanmışlardır. İlk istek geldikten sonra uygulamanın geliştirildiği programlama diline göre uygun konteyner seçilir. Bu konteyner içine uygulama yüklenerek "soğuk başlatma" denen süreç başlatılmaktadır. Süreç sonunda uygulama çalışmaya başlar ve gelen istek uygulamaya işlenmek üzere iletilir. Uygulama gelen isteği işledikten sonra ürettiği sonucu geri döndürerek çalışma süreci tamamlanır. Eğer bir sonraki istek platform tarafından belirlenen süre içerisinde gelirse daha önceki istek için çalıştırılmış ve bekleyen konteyner yeniden kullanılarak daha hızlı şekilde cevap üretilebilmektedir. Bu şekilde başlayan sürece de "sıcak başlatma" adı verilmektedir. Eğer uzun süre istek gelmezse bekleyen konteyner kapatılır. Bu nedenle uzun süre çalışmamış veya ilk kez çalışacak olan bir fonksiyonun ilk cevap dönme süresi ardı sıra gelen isteklere cevap dönme sürelerine göre uzun olmaktadır. Chowhan (2018) çalışmasından yararlanılarak fonksiyonların çalışma döngüsünü açıklamak için Şekil 2.8 oluşturulmuştur.

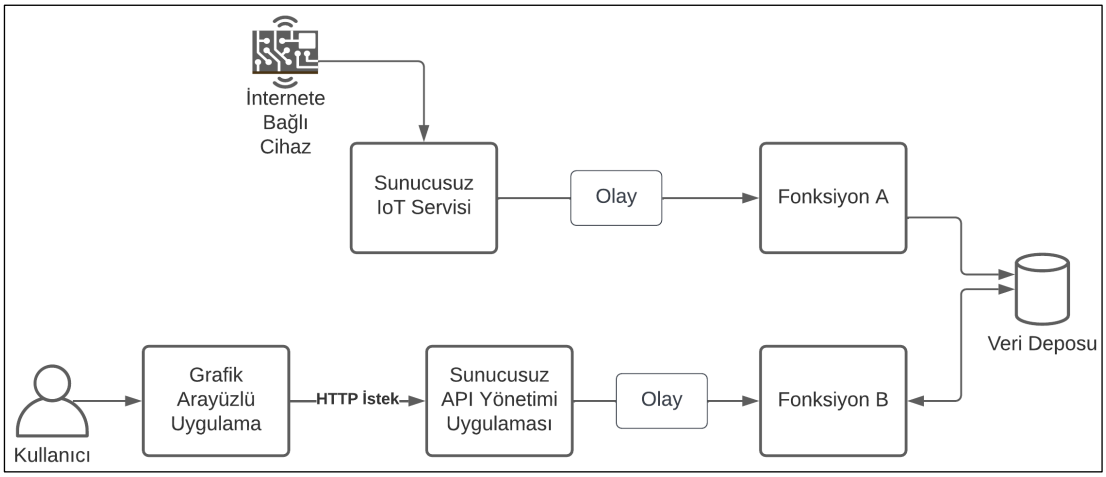


Şekil 2.8 : Bir fonksiyona ait soğuk ve sıcak başlatma süreçleri.

Bir fonksiyon platformun belirlediği ölçütlerde ölçeklenebilir. Bir yük dağıtıcısı bileşen gelen veya bekleyen isteklere göre fonksiyonun anlık olarak çalıştığı konteyner sayısına karar verir. Bir fonksiyon çalıştığı konteynerlerin hesaplama kapasitesinin idare edebileceğinden fazla istek almaya başlarsa yük dağıtıcısı bileşeni otomatik olarak yeni bir konteyner daha çalıştırarak gelen istekleri dengeli olarak dağıtır ve işlem yükünü paylaştırmış olur. Bunun sağlanabilmesi için fonksiyonlar için geliştirilen kodların ek olarak elle bir müdahale gerekmeden çalışmaya başlaması beklenmektedir. Aksi halde otomatik olarak başlatma ve buna dayalı olan ölçeklenebilirlik sağlıklı olarak uygulanamayacaktır.

Fonksiyonların uzun sürelerle çalışma beklenmez. Bir olay kaynaklı olarak çalışmaya başlarlar ve programlanan işlemleri başarılı veya başarısız olarak tamamlandıktan sonra çalışmalarını sonlandırmaları beklenir. Bununla bağlantılı olarak fonksiyonların çalıştıkları platform üzerindeki kaynak tüketimlerinin optimizasyonu sağlanır ve bu sayede donanım kaynakları devamlı olarak tek bir fonksiyon için çalışmaz. Çalışması sonlanan fonksiyondan boşalan donanım kaynağı bir başka çalışacak fonksiyon için ayırılmış olur. (Adzic ve Chatley, 2017).

Fonksiyonlar olay güdümlü olarak çalışabildikleri gibi kendileri de çalışma sorununda yeni olaylar üreterek başka bir sunucusuz hizmetin çalışmasına neden olabilirler. Bu sayede fonksiyonlar ve diğer sunucusuz hizmetler sıralı veya eş zamanlı olarak çalışarak bir akış halinde çalışabilirler. Barbieri ve Bonanni (2019) çalışmasından yararlanılarak fonksiyonun diğer servislerle olan ilişkisini açıklamak üzere Şekil 2.9 oluşturulmuştur.

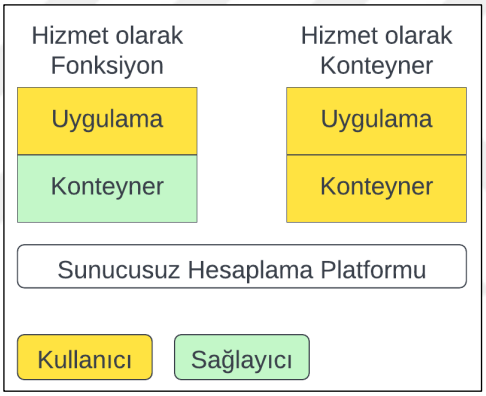


Şekil 2.9 : Sunucusuz hizmetlerin olay güdümlü olarak birbirlerini çalıştırmaları.

Fonksiyonların çalıştığı konteynerler üzerinde tutulan veriler çalışma süresi sonunda silinirler, bu nedenle fonksiyonlar durum saklamazlar (Jonas ve diğ, 2019). Çalışma zamanlarında kalıcı veri saklamak üzere bir sunucusuz veri depolama çözümü kullanılabilir. Bu sayede ardışık olarak çalışan fonksiyonlar birbirlerine veri aktarabilirler. Ayrıca bu saklanan veriler başka sistemler tarafından da kullanılmak üzere de tutulabilir.

Çalışma zamanında fonksiyonların ürettikleri günlük kayıtları ve performans metrikleri bir izleme aracı tarafından toplanarak saklanır (Lynn ve diğ, 2017). Böylece platform kullanıcısının fonksiyonun ürettiği hata kayıtlarına, kaynak tüketim ve yönetim metriklerine ulaşması ve izlemesi sağlanmış olmaktadır.

2.2.2 Hizmet olarak konteyner



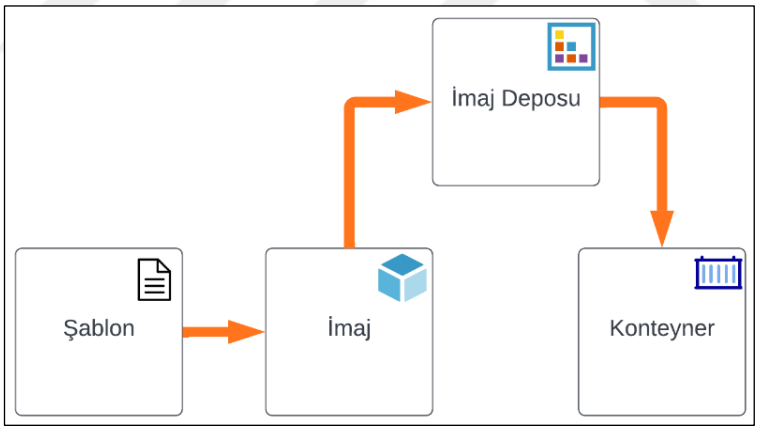
Şekil 2.10 : HoF ve HoK modellerinde kullanıcı ve sağlayıcı sorumlulukları.

Sunucusuz hesaplama modellerinden bir diğeri olan HoK servisleri fonksiyon servislerine göre daha özgür bir çalışma ortamı sunar. Bunun nedeni ise Şekil 2.10'da gösterildiği gibi platformun kullanıcıya fonksiyonların aksine sadece uygulama üzerinde değil uygulamanın çalıştığı konteynerin üzerinde de yetki alanı sunmasıdır (Léger ve Broshar, 2021). Böylece bir uygulama geliştirici uygulamanın çalışacağı konteynerde uygulamanın özel bağımlılıklarını da yükleyerek uygulamayı çalıştırabilir.

Bu bağlamda, fonksiyon ortamlarında istenen bağımlılıkların veya ayarların eksik olması nedeniyle çalışmayan mevcut CBS uygulamalarını çalıştırmak için de konteyner teknolojisi kullanılabilir (Zaragozí ve diğ, 2020). Örneğin MapServer isimli açık kaynak kodlu CBS sunucu uygulaması HoF servisleri üzerinde çalışamazken

uygulama geliştiriciler tarafından sunulan konteyner imajı sayesinde HoK servisleri üzerinde çalışabilmektedir (MapServer, 2022).

Literatür incelemesi yapıldığında birçok farklı konteyner mimarisi olduğu görülmektedir (Siddiqui ve diğ, 2019). Bu mimariler içerisinde Docker isimli konteyner mimarisi birçok farklı bulut bilişim sağlayıcısı tarafından desteklendiği için öne çıkmaktadır. Bu mimari incelendiğinde bir konteynerin hangi işletim sistemiyle, hangi bağımlılıklarla çalışacağı ve hangi uygulamayı çalıştıracağı bir şablon olarak hazırlanır. Hazırlanan şablon derlenerek imaja çevrilir. Hazırlanan şablon istenirse başka bir kullanıcı ile paylaşılabilir ve bu kullanıcı da bu şablonu derlediğinde yine aynı imaj üretilir. Bu nedenle Docker teknolojisi taşınabilirlik yönünden de avantajlıdır. Kullanıcı isterse ürettiği imajı bir ortak imaj deposuna yükleyebilir (Nickoloff ve Kuenzli, 2019; Poccia, 2020). İmaj konteyner olarak çalışacağı zaman yeniden derlenmez ve doğrudan çalışmaya başlar. Eğer konteynerin çalışacağı sunucu üzerinde yoksa imaj ortak depodan imajı çekilerek çalıştırılır (Şekil 2.11). Bu çalışma prensipleri Docker teknolojisi kullanan bulut bilişim platformları üzerinde de bu şekilde uygulanmaktadır.

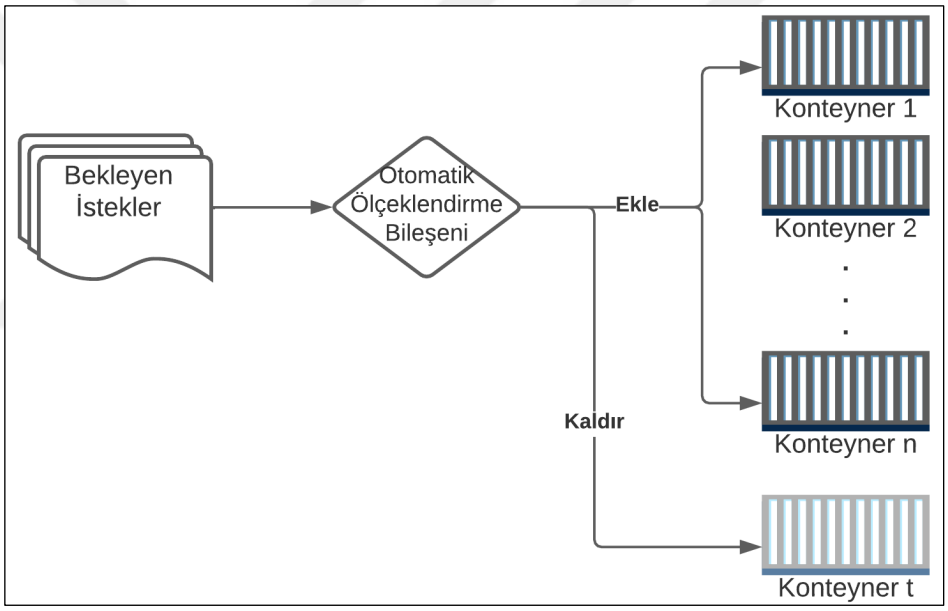


Şekil 2.11 : Bir Docker konteynerin oluşturulma süreci.

Docker teknolojisini kullanan HoK servisleri ilgili konteynere ait imajı imaj deposundan çekerek uygulamayı çalıştırırlar. HoF servisinden farklı olarak konteynerin çalışması içindeki uygulamanın işini bitirmesi ile bitmeyebilir. Bu durumda konteyner süresiz bir şekilde dışarıdan durdurma talebi gelene kadar çalışmaya devam eder. Bu nedenle HoK servisleri konteynerlerin yaşam döngülerinin kontrolü için araçlar sunarlar. Bunun yanında konteyner imajı içerisindeki uygulamanın çalışmasını bitirdiğinde kendiliğinden kapanacak şekilde de tasarlanabilir. Devamlı çalışan konteyner uygulamalarına örnek olarak CBS sunucusu

uygulamaları gösterilebilir. Bu uygulamalar bir internet iletişim protokolü üzerinden gelecek olan isteği devamlı olarak beklerler. İstek geldiği anda ise işleyip cevap dönerek bir sonraki istek için beklemeye devam ederler.

HoK servisi eğer aynı anda gelen istek veya bekleyen işlem sayısı artarsa gelen talebi daha hızlı karşılamak üzere çalışan konteyner sayısını arttırabilir (Şekil 2.12). Bu nedenle HoK servisi de HoF servisine benzer şekilde otomatik olarak ölçeklenebilir. Otomatik ölçeklendirmenin doğru çalışabilmesi için konteyner imajları otomatik olarak başlatılıp elle müdahale gerekmeden istek almaya hazır olacak şekilde üretilmelidirler. Eğer sistem üzerindeki çalışma yükü azalırsa HoK servisi ihtiyaç duymadığı konteynerleri otomatik olarak kapatır. Otomatik ölçeklendirmenin nasıl ve hangi kurallarla çalışacağı HoK servisinde kullanıcı tarafından belirlenmektedir.



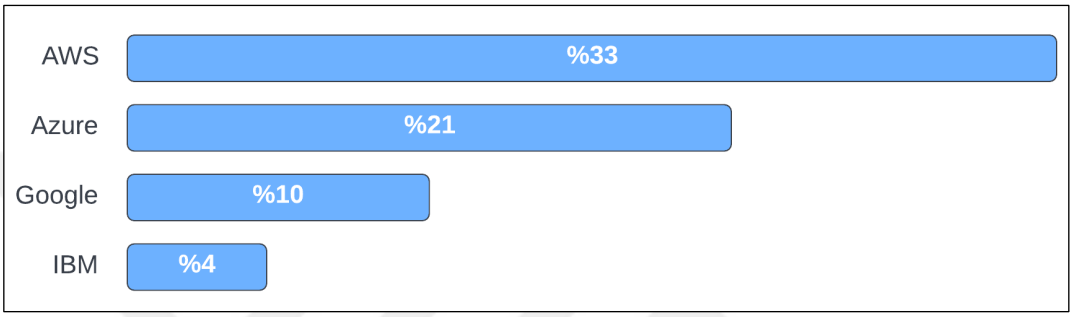
Şekil 2.12 : HoK servisi üzerinde otomatik ölçeklendirme.

Konteynerler üzerinde çalışan uygulamaların kalıcı veri depolama ihtiyaçları platform üzerindeki bir veri depolama biriminin konteynere dosya sistemi üzerinden bağlanması (mount) ile karşılanabilmektedir. Konteyner çalışmasını tamamladığında veri depolama birimi serbest bırakılarak yeniden başka bir konteynere bağlanmak üzere bekleyebilir. Kullanıcıdan veri depolama alanını seçerken ve kullanırken bazı konulara dikkat etmesi beklenir. Örneğin birden fazla konteyner aynı veri üzerine aynı anda yazmak isterse veri bir konteyner için ulaşılamaz olabilir ve yazılmak istenen veri de kaybedilebilir. Bu nedenle veri depolama çözümü dikkatli seçilmelidir. Örneğin HoK üzerinde çalışan CBS sunucusunda kalıcı veri depolamada raster veriler için

dosya sistemi çözümü kullanılabilir. Buna karşın kalıcı depolama olarak veri tabanı sunucusunda ise eşzamanlı okuma ve yazma yeteneği ile vektör veriler saklanabilir.

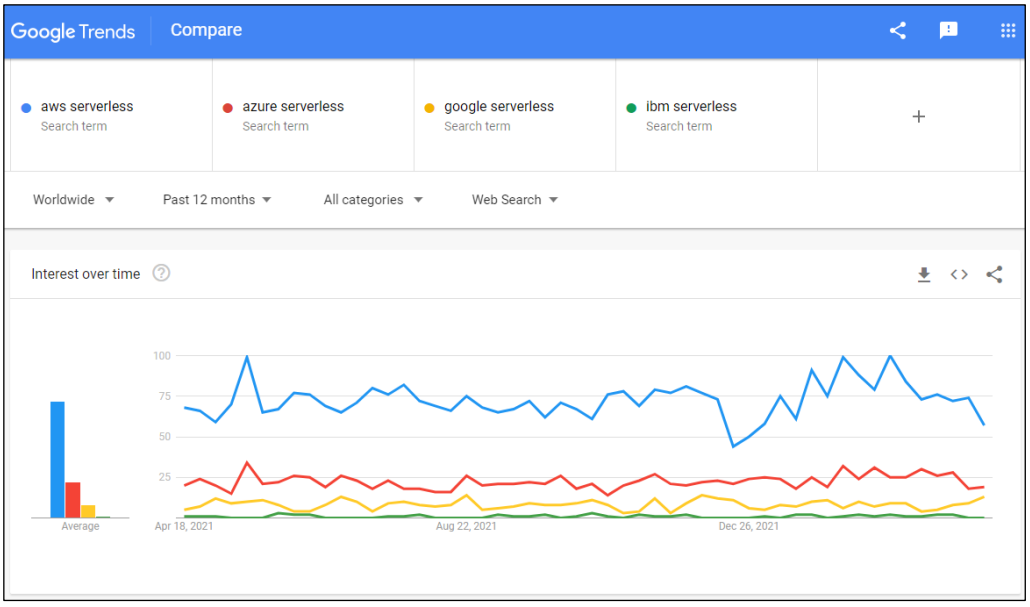
3. BULUT BİLİŞİM SAĞLAYICILARININ SUNUCUSUZ HİZMETLERİ

Baldini ve diğ. (2017) yaptıkları çalışmada AWS, Azure, Google ve IBM bulut bilişim sağlayıcısı firmanın sunucusuz hizmetler verdiğinden bahsetmiştirler. Şekil 3.1'de bu dört bulut bilişim firmasının pazar payları verilmiştir.



Şekil 3.1 : AWS, Azure, Google ve IBM bulut bilişim sağlayıcılarının 2021 son çeyreğindeki pazar payları (Richter, 2022).

Şekil 3.2'de ise yine aynı dört bulut bilişim firmasının adı ile "serverless" kelimesinin son bir yıl içinde ne kadar çok arandığı analizi Google Trends aracı yapılarak gösterilmiştir (Google Trends, 2022).



Şekil 3.2 : Google Trends'e göre AWS, Azure, Google ve IBM bulut bilişim sağlayıcılarının sunucusuz konusunda son bir sene içindeki aranma popülerliği.

Bu iki grafiğe göre AWS ve Azure bulut bilişim sağlayıcılarının diğerlerine göre daha fazla tercih edildiği ve aynı şekilde sunucusuz mimariler için de daha fazla arandığı görülmektedir.

Bu bölümde AWS ve Microsoft Azure bulut bilişim sağlayıcılarının sunduğu sunucusuz hizmetler 2. Bölüm'de verilen sınıflandırmayı takip ederek depolama ve hesaplama türünde ele alınmıştır. Bu hizmetlerin ayrıca sunduğu mekânsal bilişim özellikleri de incelenmiştir.

3.1 Amazon Web Services (AWS) Tarafından Sunulan Sunucusuz Hizmetler ve Mekânsal Bilişim Özellikleri

3.1.1 Sunucusuz veri depolama hizmetleri

AWS bulut bilişim sağlayıcısının yapısal ve ilişkisel veri depolama hizmetleriyle birlikte yapısal olmayan veri depolama hizmetlerini de sunmaktadır (Çizelge 3.1).

Çizelge 3.1 : AWS sunucusuz veri depolama hizmetleri.

Yapısal ve ilişkisel veri depolama	Amazon DynamoDB
	Amazon Aurora Serverless
Yapısal olmayan veri depolama	Amazon S3

AWS bulut bilişim sağlayıcısının yapısal olmayan veri depolama için sunduğu sunucusuz hizmeti Amazon S3 olarak isimlendirilmektedir. Amazon S3 hizmeti verileri BLOB türünde saklamaktadır (Gulabani, 2015). Amazon S3 üzerinde saklanan objelerin içerdiği coğrafi verilerin sorgulanması için platform üzerindeki bir başka sunucusuz sorgulama servisi Amazon Athena kullanılabilmektedir. Bununla birlikte bu servis üzerinde açık olarak sunulan raster ve vektör veriler platform üzerinde geliştirilecek sunucusuz uygulamalar için hızlı ulaşılabilir birer coğrafi veri deposu olarak da kullanılabilir (URL-1).

Yapısal ve ilişkisel verilerin saklanması için Amazon Aurora Serverless ve Amazon DynamoDB servisleri sunulmaktadır. Amazon Aurora Serverless servisi açık kaynak kodlu PostgreSQL ve MySQL veri tabanı sunucusu yazılımlarının AWS tarafından özelleştirilmiş sürümleri üzerinden sunulmaktadır (Zois, 2021). PostgreSQL eklenti ekosisteminde mekânsal verinin saklanması ve analizi için geliştirilmiş olan PostGIS eklentisi Amazon Aurora Serverless ile çalışabilmektedir. PostGIS eklentisi ile birçok

farklı türden mekânsal veri saklanabilmektedir ve SQL sorguları ile gelişmiş mekânsal analizler yapılabilmektedir. Bu sayede mekânsal verilerin saklanması ve ileri düzeyde sorgulanması mümkün olmaktadır (Mete ve Yomralıoğlu, 2021a).

Amazon DynamoDB anahtar-değer belge türü verilerin saklanmasını sağlayan servistir (Kalid ve diğ, 2017). Amazon DynamoDB veri sorgulama işlemlerini SQL dili desteğini kısıtlı bir şekilde vererek ve sağladığı API üzerinden sunmaktadır. Bu hizmet coğrafi verilerin saklanmasına ve sorgulanmasına kısıtlı olarak destek vermektedir. Sadece nokta türündeki coğrafi verilerin GeoHash kodlaması ile saklanarak sorgulanması mümkün olmaktadır. Bu amaçla geliştirilen Amazon DynamoDB GeoHash kütüphaneleri ile uygulama geliştirici geliştirdiği uygulama üzerinden kolayca mekânsal analizler yapabilmektedir (Beswick, 2020).

3.1.2 Sunucusuz hesaplama hizmetleri

AWS bulut bilişim sağlayıcısının sunduğu sunucusuz hesaplama hizmetleri Çizelge 3.2'de de gösterildiği gibi fonksiyon ve konteyner türünde ikiye ayrılmaktadır.

Çizelge 3.2 : AWS sunucusuz hesaplama hizmetleri.

Fonksiyon – HoF	AWS Lambda
Konteyner – HoK	AWS Fargate

HoF modelindeki servisin adı AWS Lambda olarak adlandırılmaktadır (Chapin ve Roberts, 2020). AWS Lambda fonksiyonlarının kullanabilecekleri maksimum bellek kapasitesi kullanıcı tarafından belirlenir. Bu nedenle uygulamanın ihtiyacı olan bellek miktarının önceden doğru tespitini yaparak yapılandırmak önem arz etmektedir. Verilebilecek maksimum bellek miktarı ise konteyner başına 10GB'dır.

AWS Lambda fonksiyonları olay güdümlü olarak çalışmaktadırlar. Olayların kaynağı diğer AWS servislerindeki durum ve veri değişimleri ile HTTP üzerinden gelen istekler olabilir. Fonksiyonlar olay güdümlü olmaları sayesinde diğer AWS servisleriyle birlikte çalışabilirler. Örneğin, Amazon DynamoDB'de meydana gelen veri değişimleri AWS Lambda üzerindeki uygulamayı tetikleyerek çalıştırabilir.

AWS Lambda servisinin çalışma süresi 15 dakika ile limitli olduğundan, çalışması uzun süren mekânsal büyük veri analizi veya mekânsal zekâ uygulamaları için uygun olmayabilir. Bununla birlikte bir fonksiyonun aynı anda en fazla bin isteğe cevap

verebilme limiti bulunmaktadır. Özellikle herkese açık ve çok sayıda kullanıcıya hizmet vermesi beklenen internet servisi mimarileri için bu limit bir engel olabilir. İstenirse AWS destek hattından bu limitin arttırılması talep edilebilir. AWS Lambda uygulamasının çalıştığı geçici her bir konteyner başına verilen depolama alanı ise maksimum 10GB'dır.

AWS Lambda servisi başlıca Java Script, Python, Ruby, Java, Go ve C# programlama dillerini desteklemektedir, ancak, servis ayrıca konteyner imajı çalıştırma desteği de sunduğu için uygulamada konteyner üzerinde çalışabilen her dille çalışma imkânı da sunmaktadır. Her ne kadar bu yaklaşım HoK modeline benzetilse de AWS Lambda üzerinde her konteyner imajına izin verilmez. İzin verilen konteyner imajlarının AWS tarafından hazırlanmış baz (base) imajlardan veya Lambda Runtime API'nın kurulu olduğu özel imajlardan üretilmesi gerekmektedir (Poccia, 2020). HoF modeli ile çalışamaya uygun olmayan bir CBS sunucusu yazılımı konteyner olarak hazırlanarak çalıştırılabilir. Bu imkan ile ayrıca halihazırdaki bir uygulamanın da sunucusuz mimari servislerinden faydalanması sağlanmış olmaktadır.

HoK modelindeki servis AWS Fargate olarak adlandırılmaktadır (Vohra, 2018). AWS Fargate, Docker teknolojisi ile konteyner yönetimini sunucusuz olarak sağlamaktadır. Bu servis üzerinde çalıştırılmak istenen konteyner imajları bir konteyner imaj deposunda çekilir. Bu imaj deposuna AWS Fargate erişim için yetkilendirilir.

AWS Fargate üzerindeki her bir Windows işletim sistemi tabanlı konteyner 20GB, Linux işletim sistemi tabanlı konteyner ise 200 GB (gigabyte) geçici depolama alanı üzerinde çalışmaktadır. Eğer kalıcı bir depolama ihtiyacı varsa bunun için platform üzerinde sunulan sunucusuz veri depolama hizmetleri kullanılabileceği gibi sanal disk servisinden de faydalanılabilir. Servis aynı anda bin adet konteyner çalıştırabilir. İstenirse AWS destek hattından bu limitin arttırılması talep edilebilir. Bir diğer limit ise AWS Fargate servisinin grafik işlem birimi (GPU – Graphical Processing Unit) desteği sunmamasıdır. Bu nedenle grafik işlem birimine ihtiyaç duyan mekânsal zekâ uygulamaları AWS Fargate üzerinde çalışamamaktadır (URL-2).

Platform üzerinde servisler arasındaki etkileşimin güvenliği için AWS üzerine kimlik ve yetki erişim sistemi (IAM-Identity Access Management) bulunmaktadır. Bu servis üzerinde AWS Lambda veya AWS Fargate uygulamasına rol ataması yapılarak

platform üzerindeki diğer kaynaklara hangi yetkilerle erişebileceği belirlenir ve denetlenir (Pothecary, 2021).

3.1.3 Ekonomik model

Amazon S3 hizmeti saklanan verinin ve depolama alanına yapılan veri trafiğinin büyüklüğüne bağlı olarak ücretlendirme politikası izlemektedir (Hashimoto, 2015). Amazon Aurora Serverless servisinin ücretlendirilmesi saatlik aktif olan veri tabanı sunucusunun sayısı, saklanan veri miktarı ve yapılan sorgulama sayısı üzerinden yapılmaktadır (Weaver, 2021). Örneğin, saklanan veriler hiçbir zaman sorgulanmazsa sadece depolama ücreti ödenmektedir. Bir diğer örnek ise; veri tabanında verilerin ayda bir kez bir saat boyunca sorgulanması durumunda, yapılan sorgunun kullandığı sunucuların bir saatlik kullanım ücreti ve saklanan veri miktarı ve süresi üzerinden ücretlendirilir. Amazon DynamoDB'nin ücretlendirilmesi ise veri okuma ve yazma üzerinden kullanım miktarına göre yapılmaktadır (Astrova ve diğ, 2017).

AWS Lambda fonksiyonları çalışma süresi boyunca atanan bellek miktarına ve toplam çalışma süresine göre ücretlendirilirler (Sbarski, 2022). AWS Fargate servisinin ücretlendirilmesi de benzer şekilde atanan işlemci sayısı ve bellek miktarının kullanım sürelerine göre hesaplanmaktadır (Vohra, 2018). Konteyner için sunulan geçici veri depolama alanından daha büyük bir depolama alanına ihtiyaç duyulduğunda Amazon Elastic File System (Amazon EFS) adlı sunucusuz disk hizmeti kullanılabilir. Bu hizmet ise kullanılacak diskin kapasitesi ve yapılan yazma ve okuma sayısına göre ücretlendirilmektedir (Wittig ve diğ, 2018).

3.2 Microsoft Azure Tarafından Sunulan Sunucusuz Hizmetler ve Mekânsal Bilişim Özellikleri

3.2.1 Sunucusuz veri depolama hizmetleri

Microsoft Azure üzerinde sunucusuz veri depolama hizmetleri yapısal ve ilişkisel veri depolama ile yapısal olmayan veri depolama şeklinde iki grupta incelenmiştir. (Çizelge 3.3).

Çizelge 3.3 : Microsoft Azure sunucusuz veri depolama hizmetleri.

Yapısal ve ilişkisel veri depolama	Azure Cosmos DB
	Azure SQL Database Serverless
Yapısal olmayan veri depolama	Azure Blob Storage

Microsoft Azure bulut bilişim sağlayıcısının yapısal olmayan veri depolama için sunduğu sunucusuz hizmeti Azure Blob Storage olarak isimlendirilmektedir. Bu servis üzerinde istenen herhangi bir dosya ikili biçemde obje olarak saklanabilmektedir (Ahmed ve diğ, 2018). Saklanan veriler üzerinde doğrudan mekânsal analizler yapmak için bir hazır çözüm bulunmamaktadır. Tomey (2017) yaptığı çalışmada sunucusuz Azure Data Lake Analytics hizmetinin sorgulama yeteneklerini genişleterek Azure Blob Storage üzerinde bulunan yapısal olmayan verileri sorgulamıştır.

Yapısal ve ilişkisel veri depolama hizmeti olan Azure Cosmos DB üzerinde yapısal anahtar-değer, belge, sütun ailesi ve çizge veri türlerine destek verilmektedir (Paz, 2018). Microsoft Cosmos DB üzerinde saklanabilen her veri tipine göre bir API ismi verilmiştir (Vemula, 2019).

Anahtar-değer türündeki verileri tutmak için Table API servisi kullanılmaktadır. Bu servis mekânsal verilerin sorgulanmasına destek vermemektedir.

Belge türündeki verilerin saklanması için SQL API hizmeti kullanılmaktadır. Bu hizmet mekânsal sorguları desteklemektedir. Bununla birlikte mekânsal verileri adresleyerek (geospatial indexing) daha hızlı sorgulanması da desteklenmektedir (Microsoft, 2022a). Her ne kadar NoSQL tabanlı bir veri tabanı hizmeti olsa da SQL API adından da anlaşılacağı gibi SQL dilindeki sorgulara da destek vermektedir. OGC Simple Features standartlarına uygun şekilde bazı mekânsal sorgu metotlarını da desteklemektedir (Microsoft, 2022b).

Sütun ailesi tipindeki verilerin saklanması ve sorgulanması için Azure Cosmos DB Cassandra API sunulmaktadır (Paz, 2018). Bu isim arka planda kullandığı açık kaynak kodlu veri tabanı yazılımı olan Apache Cassandra'dan gelmektedir. Bu servis mekânsal verilerin sorgulanması için bir destek sağlamamaktadır.

Son olarak çizge verileri saklamak ve sorgulamak için sunulan servisin adı Gremlin API olarak geçmektedir. Bu servis de Cassandra API gibi kullandığı açık kaynak kodlu Apache TinkerPop yazılımının Microsoft Azure tarafından özelleştirilmiş bir sürümünün sunumudur. Yapılan sorgulama diline Gremlin dendiği için adını buradan

almıştır. Bu servis üzerinde doğrudan mekânsal sorgulama yapılamamaktadır. Ancak mekânsal bilişimde en çok kullanılan rota analizleri için tercih edilmektedir. Ferreira (2014) yaptığı çalışmada çizge veri tabanı üzerinde mekânsal bilişim uygulaması geliştirmiştir. Bu çalışmada ayrıca bir ilişkisel veri tabanı ve çizge veri tabanının en kısa yol analizi karşılaştırması da yapılmıştır. Bu karşılaştırmaya göre çizge veri tabanı en kısa yol analizi için ilişkisel veri tabanına göre daha hızlı sonuç vermektedir.

Microsoft Azure üzerinde ilişkisel sunucusuz veri tabanı hizmeti için Azure SQL Database Serverless isimli servisi sunmaktadır. Bu servis Microsoft tarafından geliştirilen SQL Server veri tabanı sunucusu teknolojisinin bulut bilişim için özelleştirilmiş bir sürümüne dayanmaktadır. Bu servis üzerinde OGC'nin Simple Feature Access standardına göre mekânsal veri sorgulama ve saklamaya destek verilmektedir (Microsoft, 2022c).

3.2.2 Sunucusuz hesaplama hizmetleri

Microsoft Azure firmasının sunduğu sunucusuz hesaplama hizmetleri Çizelge 3.4'te de gösterildiği gibi fonksiyon ve konteyner türünde ikiye ayrılmaktadır (Yusuf, 2021).

Çizelge 3.4 : Microsoft Azure sunucusuz hesaplama hizmetleri.

Fonksiyon – HoF	Azure Functions
Konteyner – HoK	Azure Container Instances

Platformda HoF modelinde sunulan servisin adı Azure Functions olarak geçmektedir (Satapathi ve Mishra, 2021). Bu servis C#, JavaScript, F#, Java, Python gibi programlama dillerinde yazılmış uygulamaları çalıştırabilmektedir. Azure Functions sunucusuz çalışma planı üzerinden konteyner çalıştırma desteği sunmamaktadır.

Azure Functions birden fazla çalışma planı sunmaktadır. Bu çalışma planlarına göre ücretlendirme ve sunulan özellikler değişmektedir. Bu tez kapsamında bu çalışmalar planlarından sunucusuz çalışma planı (Consumption Plan) üzerinden Azure Functions incelenmiştir.

Bu servise ait fonksiyonlar platformdaki diğer kaynaklardaki durum değişiklikleri ve HTTP istekleri kaynaklı olaylarla olay güdümlü olarak çalışabilmektedirler. Örneğin aynı platform üzerinde bulanan Azure Maps isimli servis ve IoT çözümleri kullanılarak takip edilen bir cihazın belirlenmiş coğrafi alanlara (geofence) girmesinin

analizini yaparak bunların kayıtlarını tutan bir uygulama geliştirilebilir (Microsoft, 2022d).

Azure Functions çalışan fonksiyonlara 5 TB geçici veri depolama alanı sunmaktadır (Maslov ve Petrashenko, 2021). Her çalışma sonunda konteynerler birlikte bu geçici depolama alanı da yok edildiğinden kalıcı veriler için platformdaki sunucusuz depolama alanı hizmetleri kullanılabilir. Azure Blob Storage üzerindeki bir saklama alanı fonksiyonun konteynerindeki dosya sistemi üzerine bağlanabilmektedir. Bu sayede bulut tabanlı veri depolama hizmetlerini desteklemeyen bir uygulamayı dosya sistemi kullanarak yine sunucusuz mimari ile kullanmak mümkün olmaktadır. Örneğin bir raster karo harita servisi uygulaması hiyerarşik bir şekilde üretilmiş ve Azure Blob Storage üzerinde saklanmış karoları dosya sisteminden okuyarak servis edebilir.

Azure Functions üzerindeki fonksiyonlar Azure Application Insight servisi ile ortak çalışarak fonksiyonlarda üretilen hata günlükleri ve performans metriklerini tutarak kullanıcının analiz edebilmesi sağlanır. Bununla birlikte kullanıcı tanımlayabileceği alarmlar ile koşulların sağlanması durumunda e-posta ve kısa mesaj gibi yollarla uyarı alabilir.

Sunucusuz çalışma planındaki fonksiyonlar maksimum 10 dakika çalışma süresine sahiptirler. Ayrıca sunucusuz plan üzerindeki bir fonksiyon aynı anda Windows işletim sistemi tabanlı ise 200 konteynere kadar veya Linux işletim sistemi tabanlı ise 100 konteynere kadar ölçeklenebilir (Microsoft, 2022e). Her bir konteyner için maksimum ayrılan bellek miktarı ise 1.5 GB'dır (Maslov ve Petrashenko, 2021).

Microsoft Azure, HoK modelindeki konteyner çalıştırma servisini Azure Container Instances (ACI) adıyla sunmaktadır (Satapathi ve Mishra, 2021). Bu servis üzerinde kullanıcı hazırladığı Windows veya Linux işletim sistemi tabanlı konteyner imajını sunucusuz mimariyle çalıştırabilir.

ACI üzerinde çalışan konteynerlerin kalıcı depolama alanı olarak sunucusuz dosya saklama hizmeti Azure Files kullanılabilir. Konteynerdeki bir dosya sistemi yolu bu hizmete bağlanarak (mount) uygulamanın bu alanda saklanan dosyalara erişimi sağlanır.

Bir ACI hesabında biri ana olmak üzere maksimum 60 tane farklı Linux işletim sistemi tabanlı konteyner aynı anda birlikte çalışabilmektedir. Ana konteyner dışındakiler yardımcı olmak amacıyla çalıştırılır. Windows işletim sistemi tabanlı ACI hesabı ise

maksimum bir tane konteyner çalıştırabilmektedir. Ayrıca maksimum bir hesaptan beş tane port haberleşmeye açılabilir.

ACI servisi kullanılan grafik işlemcisi desteği de verebilmektedir. Makine öğrenimi, yapay zekâ, mekânsal zekâ (GeoAI) gibi uygulamalar bu özellikten faydalanarak daha hızlı sonuç üretebilirler. Bir ACI hesabı maksimum 16 GB bellek ve 4 çekirdekli işlemciye sahip olabilir (Ifrah, 2020). Aynı hesap içinde çalışan konteynerler bu kaynakları paylaşımlı olarak kullanırlar. Kaynak paylaşımı servis tarafından otomatik olarak yapılmaktadır.

ACI konteynerleri istenirse Azure Functions ile entegre edilerek konteyneri başlatma, durdurma ve kapatma gibi yaşam döngüleri kontrol edilebilir. Böylece dolaylı yoldan konteynerler olay güdümlü olarak çalıştırılabilirler. Bu senaryo Azure Functions hizmetinin çalışma sürelerinin veya işlem gücünün uygulama için yeterli olmadığı durumda da tercih edilebilir. Bu entegrasyonun bir diğer faydası da ölçeklenebilirliğin bu yolla sağlanmasıdır. ACI hizmeti otomatik ölçeklendirme sistemi sunmamaktadır ancak Azure Functions kullanılarak bu sağlanabilmektedir (Kerkhove, 2021).

3.2.3 Ekonomik model

Azure Blob Storage hizmeti saklanan verinin ve depolama alanıyla yapılan veri trafiğinin büyüklüğüne bağlı olarak ücretlendirme politikası izlemektedir (Daher ve Hajjdiab, 2018). Azure Cosmos DB servisinin ücretlendirilmesi servise yapılan her bir milyon sorgu isteği başına ve saklanan veri miktarına göre ücretlendirilmektedir (Piancazzo, 2022). Azure SQL Database Serverless ise seçilen veri tabanı işlem gücü kaynağına ve kaynakların saniye zaman birimindeki kullanımına göre ücretlendirilmektedir. Ayrıca Azure SQL Database Serverless için kullanılan veri miktarı da aylık olarak veri miktarı başına göre ücretlendirilmektedir.

Azure Functions fonksiyonları çalışma süresi boyunca atanan bellek miktarına ve toplam çalışma süresine göre ücretlendirilirler. ACI servisinin ücretlendirilmesi de benzer şekilde atanan işlemci sayısı ve bellek miktarının kullanım sürelerine göre hesaplanmaktadır Her iki hizmet için de kullanılacak kalıcı depolama alanlarının ücretlendirilmesi ayrıca yapılmaktadır. ACI hizmetinde grafik işlemcisi kullanıldığında ayrıca onun da kullanım süresi ve işlemci türüne göre ücretlendirme yapılmaktadır.

3.3 AWS ve Microsoft Azure Sunucusuz Hizmetlerinin Karşılaştırılması

AWS ve Microsoft Azure bulut bilişim sağlayıcılarının bir önceki bölümde sunucusuz hizmetleri incelenmiştir. Bu bölümde ise bu iki sağlayıcı tarafından sunulan sunucusuz veri depolama ve hesaplama servisleri karşılaştırılacaktır.

Sunucusuz yapısal olmayan veri depolama hizmetleri Amazon S3 ve Azure Blob Storage için karşılaştırıldığında gözle görülür bir fark görülmemektedir. İki servis arasında terminoloji farkı bulunmaktadır. Amazon S3'te veri objelerinin varlık kümesine kova (bucket) adı verilirken Azure Blob Storage hizmetinde konteyner (container) adı verilmektedir. Amazon S3 üzerinde hali hazırda bulunan açık mekânsal veriler ise bu verilerin kullanılacağı bir uygulama senaryosu için avantajlı olmaktadır.

Sunucusuz yapısal ve ilişkisel veri tabanı hizmetleri NoSQL çözümleri üzerinden karşılaştırıldığında Azure Cosmos DB'nin hem destek verdiği veri tipleri hem de mekânsal yetenekleri açısından Amazon DynamoDB'den avantajlı olduğu görülmektedir (Çizelge 3.5).

Çizelge 3.5 : Amazon DynamoDB ve Azure Cosmos DB karşılaştırması.

Özellik Adı	Amazon DynamoDB	Azure Cosmos DB
Veri Tipleri	Anahtar-Değer	Anahtar-Değer, Belge, Çizge, Sütun Ailesi
Mekânsal Veri Saklama	Var (Sadece Nokta)	Var
Mekânsal Sorgulama	Var	Var

Sunucusuz yapısal ve ilişkisel veri tabanı hizmetleri ilişkisel ve SQL dilini destekleyen çözümler üzerinden karşılaştırıldığında ise Azure SQL Database Serverless servisi ile Amazon Aurora Serverless servisinin farklı iki veri tabanı sunucusu teknolojisi kullandığı görülmektedir. Amazon Aurora Serverless çözümü PostgreSQL ve PostGIS desteği ile Azure SQL Database Serverless çözümüne göre daha fazla mekânsal sorgulama yetenekleri sunduğu görülmektedir.

Çizelge 3.6'te sunucusuz HoF modeli ile hesaplama hizmetlerinin karşılaştırılması yapılmıştır. AWS Lambda üzerinde Windows işletim sistemi tabanlı konteynerler çalıştırılamamaktadır. Bu nedenle karşılaştırma tablosu her iki servisinde Linux işletim sistemi tabanlı konteyner çalıştıracağı senaryosu üzerinden yapılmıştır.

Çizelge 3.6 : AWS Lambda ve Azure Functions karşılaştırması.

Özellik Adı	AWS Lambda	Azure Functions
Programlama Dilleri	C#, JavaScript, Java, Python, PowerShell Core, Go, Ruby	C#, JavaScript, Java, Python, PowerShell Core, TypeScript
Konteyner Çalıştırma Desteği	Var	Yok
Maksimum Çalışma Süresi	15 dakika	10 dakika
Ölçeklenebilirlik	1000	100
Maksimum Bellek Miktarı	10 GB	1.5 GB
Geçici Veri Depolama Alanı	10 GB	5 TB

AWS Lambda ve Azure Functions arasında geçici veri depolama alanı, maksimum çalışma süresi, maksimum bellek miktarı ve geçici depolama alanında belirgin farklılıklar olduğu görülmektedir. Desteklenen dillere bakıldığında AWS Lambda'nın farklı olarak Go ve Ruby dillerine destek verdiği görülürken Azure Functions'ın da farklı olarak TypeScript diline destek verdiği görülmektedir.

HoK modelindeki servislerin AWS ve Microsoft Azure bulut sağlayıcıları üzerindeki karşılaştırılmasında AWS Fargate ve Azure Container Instances servisleri kullanılmıştır (Çizelge 3.7).

Çizelge 3.7 : AWS Fargate ve Azure Container Instances karşılaştırması.

Özellik Adı	AWS Fargate	Azure Container Instances
İşletim Sistemi Desteği	Windows ve Linux	Windows ve Linux
Grafik İşlemcisi Desteği - GPU	Yok	Var
Otomatik Ölçeklendirme	Var	Yok
Aynı Anda Çalışan Konteyner	1000	60
Maksimum Bellek Miktarı	30 GB	16 GB
Maksimum İşlemci Çekirdeği	4	4
Geçici Veri Depolama Alanı	Windows: 100 GB Linux: 200 GB	50 GB

Azure Container Instances servisinin GPU desteği sunması AWS Fargate servisine göre grafik işlemciyi kullanan yapay zekâ ve makine öğrenimi uygulamalarında avantaj sağlamaktadır. AWS Fargate ile gelen dahili otomatik ölçeklendirme hizmeti ve bin konteynere kadar ölçeklenebilme Azure Container Instances karşısında avantaj olarak görülmektedir.

4. SUNUCUSUZ COĞRAFİ BİLGİ SİSTEMİ YAZILIMI TASARIMLARI VE UYGULAMALARI

Günümüzde birçok web CBS uygulaması raster karo veya vektör karo harita servislerini altlık olarak kullanmaktadırlar (Netek ve diğ, 2020). Bu servisler üretilmiş statik karo verilerinin belirli ayrıntı seviyelerine göre bir şema doğrultusunda sunulmasını sağlamaktadır. Bu bölümde bu iki harita servisi türü sunucusuz mimari ile tasarlanarak her biri seçilen bir bulut bilişim sağlayıcısı üzerinde uygulanmıştır.

Akıllı şehir, nesnelerin interneti, büyük veri gibi uygulamalar ve paradigmalar üretilen ve saklanan veri miktarını büyük ölçüde arttırmıştır. Bu verilerin içerdiği mekânsal bilgi ile geliştirilen mekânsal zekâ uygulamalarının da günümüzde giderek önemi artmaktadır (Kamel Boulos ve diğ, 2019). Bu bölümde bir mekânsal zekâ sistemin sunucusuz mimariye uygun olarak tasarımı, bir bulut sağlayıcı üzerindeki uygulaması ve değerlendirmesi de paylaşılmıştır.

CBS uygulamalarının verilerinin bulut bilişim üzerine taşınması ile mekânsal veri işleme ve analizi iş akışları da bulut bilişim sistemleri üzerinde yapılabilmektedir. Bu bölümde sunucusuz bir akış bazlı mekânsal veri işleme sistemi tasarımı geliştirilmiştir. CBS kullanıcılarının bulut bilişim hakkında ileri düzey teknik bilgi gerektirmeden bu sistemi kullanabilmeleri için de iş akışı tanımlama modelleri de geliştirilerek sunulmuştur. Geliştirilen iş akışı tanımlarının daha iyi anlaşılabilmesi için de örnek mekânsal iş analizi senaryoları verilerek tasarım desteklenmiştir.

Sunulan her sunucusuz sistem mimarisinin tasarım ve uygulaması 12 Faktör yöntemine ve bulut bilişim sağlayıcılarının geliştirdiği bulut bilişim mimarisi değerlendirme ölçütlerine göre değerlendirilmiştir.

4.1 Mimari Tasarımların Değerlendirmesinde Kullanılan Yöntemler

Sunucusuz bulut bilişim üzerindeki sistem mimarilerinin değerlendirilmesinde iki farklı yöntem kullanılmıştır. Bu yöntemler sadece sunucusuz mimariler için değil aynı zamanda diğer tüm bulut bilişim altyapısı kullanan mimariler için

uygulanabilmektedir. Bu tez kapsamında bu yöntemler sunucusuz mimariler özelinde incelenmiş ve uygulanabilirlikleri değerlendirilmiştir.

Kullanılan yöntemlerdeki prensipler haricinde tez kapsamında sunulan sunucusuz mimari tasarımlar aşağıdaki hususlar da dikkate alınarak geliştirilmiştir.

- Sistem tasarımlarındaki tüm uygulamalar HoK ve HoF modelleri üzerinde çalışacak şekilde tasarlanmalı ve geliştirilmelidir.
- Uygulamaların ihtiyacı olan tüm destek servisleri yine sunucusuz hizmetlerle karşılanmalıdır.
- Sistem tasarımı ve uygulamalar otomatik ölçeklendirmeye uygun olmalıdır.
- Uygulamalar asenkron çalışmalıdır. Böylece sistem kaynakları tek bir işlem için bekletilmeden diğer işlemler için de kullanılabilir kılınmalıdır.
- Uygulamalar hatalara dayanıklı olmalıdır. Herhangi bir hata oluşması durumunda yarım kalan süreç yeniden denenmelidir. Belli bir sayıdan sonra süreç yeniden deneyerek devam edemiyorsa süreç ile ilgili durum bilgileri hata oluşturan kök neden çözüldükten sonra yeniden işleme devam edebilmek için kaydedilmelidir.
- Veri depolama kaynakları hariç hiçbir uygulama veya veri tabanı sunucusu için donanım kaynağı boşta bekletilmemelidir. HoK ve HoF modelindeki uygulamalar görevlerini tamamladıktan sonra kapanarak çalışma süreleri sonunda kullanılan kaynakları tekrar serbest bırakabilmelidir.

4.1.1 12 Faktör yöntemi

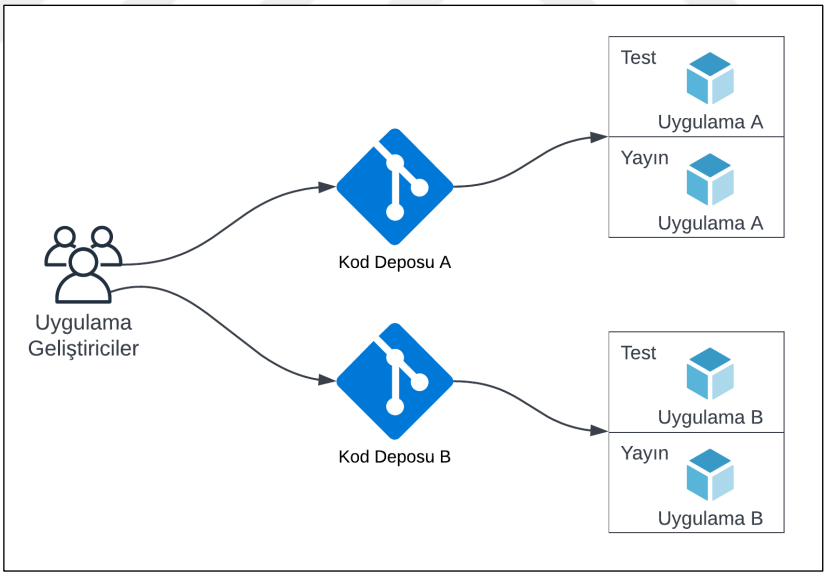
Günümüzde uygulama geliştiriciler uygulamalarını bulut bilişim altyapılarına taşımak için çalışmaktadır. Bu gelişim içerisinde yeni yaklaşımlar ve prensipler de beraberinde geliştirilmektedir. 2012 yılında Heroku bulut bilişim sağlayıcısında çalışan bir grup mühendis 12 Faktör isimli bir manifesto yayınlamıştırlar (Wiggins, 2012). Bu manifestoda bir bulut bilişim uygulamasının en doğru şekilde yapılandırılması için kılavuz olacak on iki temel madde geliştirmiştirler. Bu maddelere dayanarak geliştirilecek uygulamaların klasik yerinde (on-premise) bilişim altyapılarında çalışan uygulamalara göre daha taşınabilir ve hatalara dayanıklı olması hedeflenmiştir.

Bu tez çalışmasında 12 Faktör yöntemindeki on iki maddenin sunucusuz bulut bilişim uygulamaları üzerinde uygulanabilirliği incelenmiştir. Daha sonra sunucusuz bulut bilişim için uygun bulunan maddeler ile tezde sunulan sistem tasarımları değerlendirilmiştir.

Kod deposu (Codebase)

Uygulama geliştirme süreçlerinde üretilen kodlar bir revizyon sistemi üzerinde tutularak arşivlenmektedir. Bu arşivlerin her birine kod deposu denir. Kod depoları sayesinde birden fazla uygulama geliştirici aynı uygulama üzerinde ortak çalışarak çalışmalarını birleştirebilirler. Bu yöntem her bir kod deposu bir uygulamanın kaynak kodunu tutacağını belirtmektedir. Bu kod deposu uygulamanın test ve yayın ortamlarına yapılan güncellemeler için tek kaynaktır (Şekil 4.1).

Sunucusuz mimariyle uygulama geliştirilirken de bu yöntem kullanılabilir. HoF ve HoK hizmet modellerine göre geliştirilecek uygulamaların kaynak kodları her bir konteyner veya fonksiyon uygulaması için ayrı bir kod deposunda olacak şekilde tutulmalıdır.



Şekil 4.1 : Uygulama geliştirici, kod deposu ve uygulama çalışma ortamı ilişkisi.

Bağımlılıklar (Dependencies)

Bağımlılıklar prensibine göre bulut bilişim uygulaması hiçbir zaman bir bağımlılığın ortam tarafından sunulacağına güvenmez ve kendisi bağımlılıklarını çalışmadan önce yüklenmesini sağlar.

Bu prensip HoK modeli için tamamen uygulanabilir olsa da HoF modeli için her platform üzerinde uygulanamayabilir. Örneğin raster verilerin işlemesi için gereken bir uygulama kütüphanesinin kurulabilmesi için işletim sisteminde kurulması gereken başka bir yardımcı uygulama HoF modelinde konteyner platform tarafından yönetildiği için kurulamamaktadır. Bu nedenle bu prensip HoF modeline dayalı sunucusuz mimariler için uygulanabilir olmamaktadır.

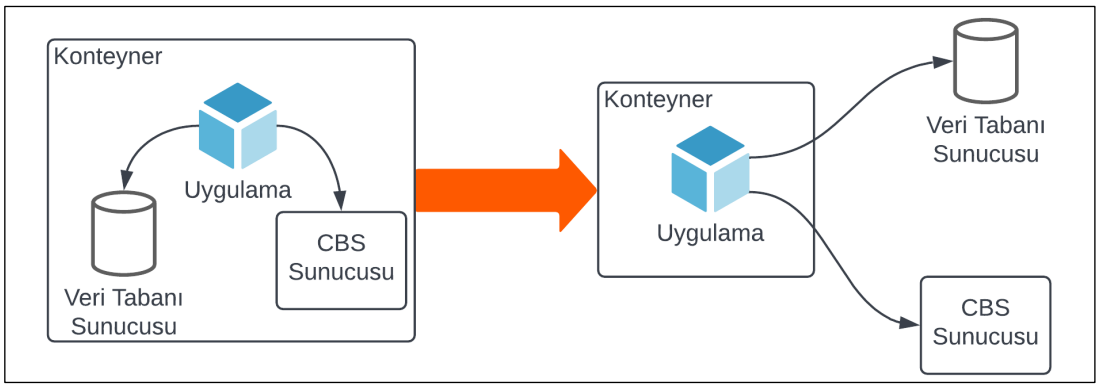
Ayarlar (Config)

Bu prensip uygulamaya ait yapılandırma ayarlarının nasıl saklanması gerektiğini belirtmektedir. Bir bulut bilişim uygulaması ihtiyacı olan yapılandırma parametrelerini uygulamanın kodları içerisinde tutmamalıdır. Bu parametrelerin değerleri uygulama yeniden yayınlanmadan değiştirilebilmelidir.

Sunucusuz hesaplama servislerinde uygulamanın ihtiyaç duyabileceği değişkenlerinin tutulması için çözümler bulunmaktadır. Bir çözüm olarak çalışan konteynerin işletim sisteminin ortam değişkenleri düzenlenerek bu parametreler sağlanmaktadır. Bununla birlikte bulut bilişim sağlayıcıları veri tabanı şifresi gibi gizli bilgilerin şifrelenerek daha güvenli bir şekilde saklanabileceği ve uygulamaların kullanabileceği hizmetler sunmaktadırlar. AWS bulut bilişim sağlayıcısı gizli uygulama yapılandırma parametreleri için Key Management Service (KMS) sunucusuz hizmetini sunmaktadır. Benzer şekilde Microsoft Azure ise Key Vault sunucusuz hizmeti ile bu özelliği sunmaktadır.

Destek servisleri (Backing services)

Destek servisleri uygulamanın ihtiyaç duyabileceği veri tabanı, mesaj kuyruğu veya önbellek yazılımları gibi ağ üzerinden ulaşılan servisleri kapsamaktadır. Bu prensibe göre bu servisler uygulama ile aynı ortamda çalışmamalıdır. Tüm destek servislerine ağ üzerinden erişilmeli ve destek servisi başka bir sunucu üzerinde olmalıdırlar. Örneğin bir CBS sunucusu uygulaması ile CBS veri tabanı sunucusu aynı konteyner üzerinde çalışmamalıdır (Şekil 4.2).



Şekil 4.2 : Destek servislerinin doğru şekilde kullanımı.

HoF modeline dayalı sunucusuz hesaplama hizmetlerinde konteyner bulut bilişim sağlayıcısı tarafından yönetildiği için çalışacak uygulama dışında başka bir uygulamanın birlikte çalıştırılmasına izin verilmez bu nedenle HoF modeli bu prensibi doğal olarak karşılamaktadır. HoK modelinde ise konteyneri kullanıcı kontrol edebileceğinden konteynerde birden fazla uygulama çalıştırılabilir. Bu nedenle bu prensip HoK modelinde ihlal edilebileceğinden dikkat edilmelidir.

Derle, yayınla ve çalıştır (Build, release, run)

Bu prensibe göre derleme, yayınlama ve çalıştırma süreçleri birbirlerinden ayrı fazlarda olmalıdır. Her bir yayın bir sürüm numarası almalıdır ve istenirse bir hata durumunda bir önceki sürüme dönülebilmelidir. Bunun için üretilen önceki sürümler yeniden yayınlanabilecek şekilde saklanmalıdır. Derleme fazı uygulama geliştirici tarafından istendiği zaman çalıştırabilmelidir.

Sunucusuz hizmetler bu prensibe uygun olarak sürekli entegrasyon (continuous integration) ve sürekli dağıtım (continuous deployment) araçlarıyla birlikte çalışarak uygulamaları yayına alabilirler. Bu araçlar kod deposundan uygulamanın son halini çekerek derler, testlerini çalıştırır ve testlerde hata yoksa yeni sürümü yayınlarlar. Ayrıca eski sürümleri de olası bir durumda geri dönülmek üzere arşivlemektedirler.

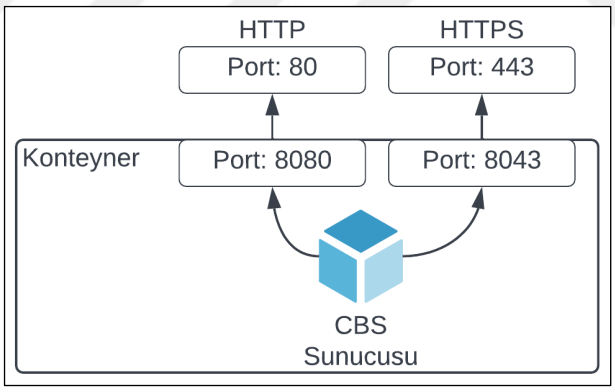
Süreçler (Processes)

Süreçler prensibine göre bir uygulama durumsuz (stateless) olmalıdır ve böylece çalışan birden çok süreç birbirleriyle durumlarını paylaşmadan çalışabilmelidir. Örneğin bir uygulama ürettiği verileri dosya sistemine veya hafızaya kaydederek bir sonraki çalışacak süreç tarafından kullanılmasını beklememelidir. Böyle bir ihtiyaç için veri tabanı sunucusu, dağıtık önbellek (distributed cache) sunucusu gibi destek servisleri kullanılarak veriler saklanmalı ve süreçler arasında paylaşılmalıdır.

Sunucusuz HoF modelindeki uygulamalar olay güdümlü olarak durumsuz çalışırlar ve kalıcı depolama ihtiyaçları için destek servislerine ihtiyaç duyarlar. Bu nedenle uygulamanın durumsuz olması uygulama geliştiricinin kontrolünde değildir. Sunucusuz HoK modelindeki uygulamalar ise uygulama geliştirici tarafından istenirse durum tutabilirler ve bu prensibe aykırı davranabilirler. Bu neden bu prensibin HoK modelindeki uygulamalar tarafından uygulanması uygulama geliştiricinin kararındadır.

Port bağlama (Port binding)

Bu prensibe göre konteyner içerisinde çalışan uygulamaların ihtiyaç duyduğu iletişim portlarının bağlanması (binding) gerekir. Böylece bir uygulama konteyner içindeki iken çalıştığı port üzerinden iletişim kurabilir ve sağlıklı olarak çalışabilir. Örneğin bir CBS sunucusu uygulaması TCP (Transmission Control Protocol) 8080 portu üzerinden HTTP ve TCP 8043 üzerinden HTTPS (Hypertext Transfer Protocol Secure) isteklerine cevap veriyorsa bu uygulama HTTP'nin rezerve edilen TCP 80 portu ile HTTPS'in rezerve edilen TCP 443 portlarına bağlanmalıdır (Şekil 4.3). Bu sayede uygulamanın istemcilerle iletişimi doğru şekilde kurulabilmektedir.



Şekil 4.3 : Konteyner üzerinde çalışan uygulamanın portlarını bağlama gösterimi.

Bu prensip HoF modelinde çalışan sunucusuz uygulamalar için uygulanamamaktadır çünkü HoF uygulamaları olay güdümlü olarak çalışmakta ve platform tarafından belirlenen portlar üzerinden bağlanarak iletişim kurmaktadırlar. HoK modelinde ise port bağlama ayarları kullanıcı sorumluluğunda olduğu için bu prensip uygulanabilmektedir.

Eş zamanlılık (Concurrency)

Uygulamalar gerektiğinde birden fazla süreçte çalışarak aynı anda birçok bekleyen işlemi tamamlayabilmelidirler. Örneğin bir uygulama raster verisi üzerinde filtre

uygulamak üzere geliştirilmişse, gerektiğinde platformun izin verdiği ölçüde eş zamanlı olarak çalışarak bekleyen on farklı raster verisini on ayrı süreçle işleyerek filtreyi uygulamalıdır. Süreçler prensibinde de belirtildiği gibi eş zamanlı çalışan süreçler birbirleriyle veri paylaşımını doğrudan değil destek servisleri üzerinden yapmalıdırlar.

Sunucusuz mimariye dayalı tüm uygulamalar bu prensibe uymaktadırlar. Her uygulama ayrı bir süreç olarak sürece ait konteyner içinde çalışarak yaşam döngüsünü tamamlar. Birden fazla süreç gerektiğinde birden fazla konteyner ile uygulama platformun izin verdiği limitler içerisinde ölçeklenerek eş zamanlı olarak çalışabilir.

Kullanılabilirlik (Disposability)

Uygulamalar çok kısa sürede çalışmaya başlamalı ve kapanabilmelidirler. Eğer işletim sisteminin süreç yöneticisi tarafından uygulamaya sonlandırma sinyali gönderilirse uygulama en zararsız şekilde ve kısa sürede kapanmalıdır. Bununla birlikte uygulamalar ani kapanmalara hazır olmalıdır. Herhangi bir donanım veya konteyner sistemi hatası durumunda uygulamalar ani şekilde kapanabilirler. Bu nedenle uygulama geliştirici sürecin tekrar kaldığı yerden devam edebilmesini sağlayacak geliştirmeleri de yapmalıdır.

Sunucusuz mimariler kaynakların ideal kullanılmasını hedeflediği için uygulamaların hızlı başlaması önemlidir. Ayrıca bulut bilişim sağlayıcılarının sunucusuz hizmetleri çalışma süresine bağlı ekonomik model geliştirmiştirler. Uygulamaların çalışmalarını kısa sürede tamamlaması sunucusuz mimariler için ekonomik olarak da verimlilik sağlar.

Geliştirme, test ve yayın ortamlarının benzerliği (Dev/prod parity)

Uygulamaların geliştirilme, test ve yayın ortamlarının birbirlerine olan benzerliğinin en yüksek seviyede olması önemlidir. Bu sayede ortam kaynaklı bir sorun yayın aşamasına gelmeden önce görülerek henüz test yayınındayken çözülebilir. Ayrıca bu benzerlik sayesinde yeni eklenen bir özellik daha hızlı ve güvenilir bir şekilde yayına gidebilir.

Sunucusuz hesaplama sistemleri konteyner tabanlı oldukları için bu teknolojinin getirdiği en büyük avantaj; konteyner içerisindeki çalışma ortamının üzerinde çalıştığı donanımdan bağımsız her ortamda aynı çalışabilme yeteneğidir. Bu nedenle bu prensip sunucusuz mimariler için kolayca karşılanabilmektedir.

Günlük tutma (Logs)

Uygulamalar günlük kayıtlarını dosya sistemi üzerinde tutmayarak bir günlük toplama sisteminde biriktirmelidirler. Merkezi bir yerde toplanan günlük kayıtları zamana ve türüne göre gruplandırılarak analiz edilebilir. Uygulamalar günlüklerin nasıl ve nerede tutulacağı ile ilgilenmezler ve günlük kayıtlarını standart yollarla oluşturmaya devam ederler. Uygulamanın çalışacağı platform bu günlük kayıtlarını toplayarak biriktirmekle sorumludur.

Sunucusuz mimarilerde günlük kayıtlarının konteynerlerden toplanarak biriktirilmesi platform araçlarına bağlıdır. Günlük kayıtlarının tutulacağı, takip ve analiz edileceği servisin sunucusuz mimariyle uyumlu olması gerekmektedir.

Yönetim süreçleri (Admin processes)

Yönetimsel tek seferlik görevlerin çalışmasına destek verilmelidir. Örneğin uygulamanın ihtiyaç duyacağı kütüphanelerin yüklenmesi için ilgili bir komutun çalıştırılması gerekebilir. Bu durumla uygulama geliştiricinin sitem üzerinde tek seferlik bu görevi çalıştırabilmesi gerekmektedir.

Uygulama geliştirici yapılması gereken yönetimsel görevleri platformun desteklediği dil ve formatta deklare ederek yayın sırasında çalıştırılmasını sağlayabilir. Bunun için sürekli dağıtım sistemi kullanılabilir.

Yönetimsel sorumluluklar sunucusuz mimariyle çalışan sistemlerde soyutlanarak platform tarafından yönetilmektedir. Bu nedenle bu prensip sunucusuz mimarilerde uygulanabilir değildir.

12 Faktör yönteminin sunucusuz sistem mimarilerine uygulanabilirliğinin değerlendirilmesi

Çizelge 4.1'de 12 Faktör'ün sunucusuz hesaplama hizmetlerine uygulanabilirlik matrisi verilmiştir. Bu matrise göre HoK ve HoF modelindeki sunucusuz hizmetlerde ortak olarak "Yönetim Süreçleri" prensibinin uygulama geliştirici tarafından doğrudan uygulanamadığı görülmektedir. Sunucusuz sistemlerde yönetimsel süreçler platformun tarafından kontrol edilmektedir. Bu prensip dışındaki diğer prensiplerin HoK modeli ile uygulanabilir olduğu görülmektedir. HoF modelinde ise "Bağımlılıklar", "Süreçler", "Port Bağlama" prensipleri kullanıcı kontrolünde olmadığı için uygulanabilir değildirler.

Uygulanabilir olmayan prensipler aslında ihlal edilmemektedir. Bu prensipler kullanıcının sorumluluk alanında olmadığı ve platform tarafından doğrudan uygulandığı için sunucusuz mimarilerde doğrudan uygulanabilir olmayan olarak değerlendirilmiştir. Bu tez çalışmasında verilen sistem tasarımları kullanılan sunucusuz hesaplama hizmet modeline göre uygulanabilir prensipler üzerinden değerlendirilecektir.

Çizelge 4.1 : 12 Faktör yönteminin HoK ve HoF modelleri ile uygulanabilirliği.

Prensip Adı	HoK	HoF
Kod Deposu	✓	✓
Bağımlılıklar	✓	×
Ayarlar	✓	✓
Destek Servisleri	✓	✓
Derle, Yayınla ve Çalıştır	✓	✓
Süreçler	✓	×
Port Bağlama	✓	×
Eş Zamanlılık	✓	✓
Kullanılabilirlik	✓	✓
Geliştirme, Test ve Yayın Ortamlarının Benzerliği	✓	✓
Günlük Tutma	✓	✓
Yönetim Süreçleri	×	×

4.1.2 Bulut bilişim sağlayıcılarının mimari değerlendirme ölçütleri

Bulut bilişim sağlayıcıları kullanıcıların oluşturdukları sistem mimarilerini belli kıstaslar üzerinden değerlendirmek ve onlara tavsiyeler sunmak amacıyla kılavuz uygulamalar geliştirmiştirler. 3. Bölüm'de incelenen bulut bilişim sağlayıcıları ayrıca tezde sunulan sistem tasarımlarının uygulanmasında da kullanılan platformlar oldukları için sundukları mimari değerlendirme ölçütleri bu bölümde açıklanmıştır. Açıklanan bu ölçütler her bir sunulan sistem tasarımının ve uygulamasının değerlendirilmesinde kullanılacaktır.

AWS bulut bilişim sağlayıcısının sunduğu mimari değerlendirme kılavuzunun adı AWS Well-Architected olarak geçmektedir. Bu kılavuzda sunulan değerlendirme ölçütleri altı ana başlık altında toplanmaktadır;

1. Maliyet Optimizasyonu
2. Operasyonel Mükemmellik
3. Performans Verimliliği

4. Güvenilirlik
5. Güvenlik
6. Sürdürülebilirlik

Microsoft Azure bulut bilişim sağlayıcısının sunduğu değerlendirme kılavuzunun da adı çok benzer şekilde Azure Well-Architected olarak geçmektedir. Bu aracın sunduğu değerlendirme ölçütleri beş ana başlık altında toplanmaktadır (Sahay, 2020);

1. Maliyet Optimizasyonu
2. Operasyonel Mükemmellik
3. Performans Verimliliği
4. Güvenilirlik
5. Güvenlik

İki aracın da başlıkları karşılaştırıldığında AWS mimari değerlendirme aracının ek olarak sürdürülebilirlik başlığını da değerlendirmeye aldığı görülmektedir.

Maliyet optimizasyonu

Maliyet optimizasyonu sunucusuz mimarilerde en önemli konu başlığıdır. Kaynaklar sadece olay güdümlü olarak çalıştırıldıkları için çalışma süresi boyunca harcanan kaynak üzerinden ücretlendirilmektedirler. Maliyet optimizasyonunu arttırmak için servis için atanan bellek ve işlemci miktarının doğru seçilmesi önemlidir. Ayrıca eğer izin veriliyorsa maksimum eş zamanlı çalışabilecek konteyner veya fonksiyon sayısının da sınırlandırılması otomatik ölçeklendirme ile oluşabilecek beklenmedik maliyetlerin önlenmesini sağlamış olmaktadır.

Sistem tasarımlarında kullanılabilecek gereksiz bileşenler de maliyetleri arttıracaktır. Bu nedenle sunucusuz mimarilerde bileşenlerin birbirleriyle doğrudan iletişim kurma yetenekleri değerlendirilmelidir. Eğer iki bileşen birbirleri ile doğrudan iletişim kurabiliyorsa araya fazladan bileşen sokulmamalıdır.

Operasyonel mükemmellik

Operasyonel mükemmelliğin sağlanabilmesi için sunucusuz bir mimari içerisindeki tüm bileşenlerin takip edilmesi ve günlük kayıtlarının tek bir yerden izlenmesi önem taşımaktadır (Obey, 2022). Uygulamaların yeni sürümlerinin yayınlanmasının otomasyonunun sağlanmalıdır. Bir diğer önemli nokta ise sistem üzerindeki

kullanıcıların rolleri doğru şekilde tanımlanmalı ve sorumlulukları açıkça belirtilmelidir.

Performans verimliliği

Sunucusuz mimarilerde performansa etki eden en önemli faktör uygulamanın çalışmaya başlaması için geçen süredir. Eğer uygulama geç tepki veriyorsa HoF ve HoK modellerine göre çalışan altyapılarda konteyner hazır durumda dahi olsa uygulamanın çalışmaya başlaması beklenecektir.

Eğer uygulama aynı istekleri çok sık alıyor ve bu isteklere aynı cevapları dönüyorsa önbellek servisleri kullanılarak cevap verme süresi düşürülerek çalışma performansı arttırılabilmektedir.

Güvenilirlik

Sunucusuz mimariye dayalı bir sistem eğer bulut bilişim sağlayıcısı üzerinde konuşlandırılacaksa güvenirliğin arttırılması için tek bir bölgedeki veri merkezi üzerinde değil birçok bölgeye yayılarak konuşlandırılabilir. Bu sayede bir bölgede meydana gelebilecek olası kesintilerde etkilenilmemiş olunur. Sunucusuz mimariye uygun geliştirilen uygulamalar oluşan hatalara karşı süreci yeniden tekrar edebilir olarak tasarlanmalıdırlar. Sunucusuz mimariye uygun olarak tasarlanan sistemin doğru şekilde yapılandırılarak örneğin eğer gerekiyorsa eş zamanlı çalışma limitleri de arttırılmalıdır.

Güvenlik

Sunucusuz mimarilerde kullanılan her bir sistem bileşeni bir rol ve buna bağlı olarak yetkiler içerisinde görevini yerine getirmektedir. Bu nedenle sistem bileşenlerine verilecek yetkiler ihtiyacından fazlası olmamalıdır. Eğer sistem bileşenlerine geniş yetkiler verilirse herhangi bir sistem bileşeninin güvenlik zafiyeti tüm sistemi tehlikeye atabilir. Bir diğer güvenlik zafiyeti ise uygulamaların çalışırken diğer destek servislerine bağlantıda ihtiyaç duyacağı kimlik bilgileri gibi bilgilerin güvenli bir şekilde saklanması gerekmektedir.

Sürdürülebilirlik

Sunucusuz mimariye dayalı sistemlerdeki bileşenlerin kullandıkları donanım kaynakları sadece çalışma süresinde enerji tüketirler ve daha sonra bu kaynağı serbest bırakarak diğer sistemlerdeki bileşenler için kullanılabilir hale getirirler. Bu yaklaşım

fiziksel donanım kaynaklarından elde edilen verimliliği en iyi hale getirirken bir diğer yandan da oluşan karbon ayak izini azaltarak sürdürülebilir bir altyapının oluşturulmasını sağlar. Bulut bilişim sağlayıcıları veri merkezlerinin enerji ihtiyaçları için doğa dostu teknolojiler kullanabilmektedirler. Örneğin AWS bulut bilişim sağlayıcısı bu konuda yaptığı yatırımları ve güncel durumunu devamlı olarak paylaşmaktadır (Amazon, 2022a).

Sürüm yayınlama otomasyonu ile sistemlerdeki bileşenlerin nasıl yayınlanacağı tanımlanır. Ayrıca kodlar kod deposunda arşivlenmektedir. Bu iki husus uygulama geliştiricinin değişmesi durumunda yeni gelen uygulama geliştiricin sistemi daha kolay öğrenmesini sağlayarak sistemin sürdürülebilirliğine katkı sağlar (Sanchez ve diğ, 2020).

Bulut bilişim sağlayıcılarının mimari değerlendirme ölçütlerinin sunucusuz sistem mimarilerine uygulanabilirliğinin değerlendirilmesi

Bu tez çalışmasında sürdürülebilirlik prensibi sunulan tüm sunucusuz sistem mimarilerinde kaynak optimizasyonu, sürüm otomasyonu ve kod deposu ile aynı şekilde uygulanmıştır ve ayrıca sistem mimarisi özelinde değerlendirilmemiştir.

Tez çalışmasında maliyet optimizasyonu prensibi de sunulan tüm sistem mimarilerinde aynı şekilde uygulanmış ve ayrıca sistem mimarisi özelinde değerlendirilmemiştir. Sunulan sistemler sadece sunucusuz bileşenlerle tasarlanmış ve uygulanmıştır. Sistem tasarımlarında gereğinden fazla bileşen kullanılmamıştır. Ayrıca her sistem tasarımında günlük tutma servisi ile hataların takibi yapılarak kaynakların hata nedeniyle boşuna çalışmasının önüne geçilmiştir.

Tezde sunulan tüm sistemler için operasyonel mükemmellik prensibi aynı şekilde uygulanmış ve ayrıca sistem mimarisi özelinde değerlendirilmemiştir. Bu prensip kapsamında tüm sistemlerde sürüm otomasyonu kullanılmıştır. Ayrıca her sistemdeki kullanıcıların rolleri de detaylı şekilde açıklanmıştır.

Sonuç olarak “Performans Verimliliği”, “Güvenilirlik”, ve “Güvenlik” prensipleri üzerinden tüm sunucusuz sistem mimarisi tasarımları ve uygulamaları özel olarak değerlendirilmiştir.

4.2 Sunucusuz Vektör Karo Harita Servisi Tasarımı, Uygulaması ve Değerlendirmesi

Coğrafi verilerin vektör formatında web CBS uygulamaları için web servisleri üzerinden sunulması raster verilerin sunumu kadar kolay olmamaktadır (Antoniou ve diğ, 2009). Bu nedenle raster veriler harita web servisleri için daha çok tercih edilmektedir ancak raster verilerin de vektör verilere göre dezavantajları vardır (Bertolotto ve Egenhofer, 2001). Örneğin raster veri üzerinde web CBS uygulaması ile kullanıcı tarafından değişiklik yapılmak istendiğinde mümkün olmamaktadır. Gelişen teknoloji ile vektör verilerin web haritaları üzerinde görselleştirilmesi için çeşitli yöntemler geliştirilmiştir (De Beukelaar, 2018). Vektör karo, harita üzerindeki bir küçük bölgenin verisinin vektör olarak paketlenmesi ile oluşturulmaktadır. Oluşturulan karolar bir arada görselleştirildiğinde bir bütün haritayı oluştururlar (Lopez ve diğ, 2017).

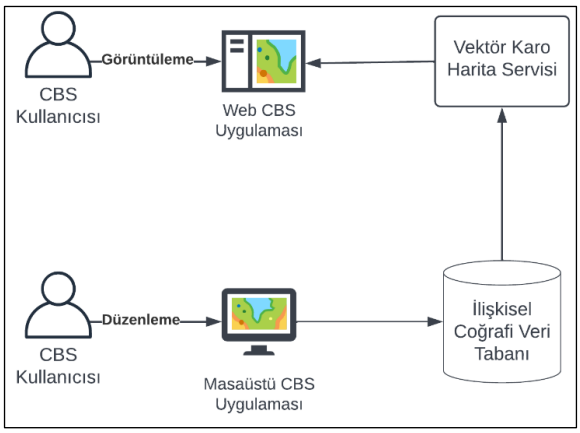
OGC'nin "Vector Tiles Engineering Report" isimli raporunda mevcut vektör karo formatları incelenmiştir (OGC, 2018). Bu rapora göre vektör karolar üç farklı formatta sunulabilmektedir. Bunlardan MapBox Vector Tile (MVT) formatı ikili (binary) formatta saklanmakta ve geniş uygulama desteği sunmaktadır. MVT formatı Google'un geliştirdiği Protocol Buffer Format (PBF) kodlamasını kullanmaktadır. Bu kodlama ile verilerin boyutu küçültülerek daha kolay taşınabilmektedir (Li ve diğ, 2017).

Bu tez çalışmasında vektör karo harita servisinin sunucusuz hesaplama ve veri depolama hizmetleri kullanılarak oluşturulan sistem tasarımı sunulmuştur. Bu tasarım da vektör karo formatı olarak MVT kullanılmıştır.

4.2.1 Tasarım

Senaryo

Sistem mimarisi tasarımında kullanılan senaryo bir CBS web uygulaması kullanıcılarının bir veri tabanındaki coğrafi vektör verilere ve özniteliklerine vektör karo harita servisi ile güvenli bir şekilde ulaşabilmeleri üzerinde kurgulanmıştır (Şekil 4.4).



Şekil 4.4 : Vektör karo harita servisi kullanım senaryosu.

Veri tabanındaki coğrafi veriler masaüstü CBS kullanıcıları tarafında da sürekli güncellenmektedir. Vektör karo servisi her zaman veri tabanındaki güncellenen yeni verileri servis etmektedir.

Gereksinimler

Senaryoda verilen kurgunun sunucusuz bir sistem mimarisi tarafından gerçekleştirilebilmesi için gereksinimler belirlenmiştir. Vektör karo harita servisi tasarımı aşağıdaki gereksinimlere göre geliştirilmiştir.

1. Vektör karo formatı MVT olmalı ve PBF ile kodlanmalıdır.
2. Vektör karo sunucusu:
    a. Vektör karo sunucusu uygulaması sunucusuz hesaplama hizmetleri üzerinde çalışabilmelidir.
    b. Vektör karo sunucusu MVT formatında harita yayını yapabilmelidir.
    c. Üretilen vektör karolar ön belleğe alınarak bir sonraki istekte daha hızlı sunulabilmelidir.
3. Veri depolama:
    a. Vektör karolara kaynak olan coğrafi veriler sunucusuz veri depolama hizmetleri üzerinde saklanabilmeli, sorgulanabilmeli ve düzenlenebilmelidir.
    b. Coğrafi verileri depolayan sunucusuz veri depolama hizmeti ilişkisel veri tabanı olmalıdır.
    c. Coğrafi verileri depolayan sunucusuz veri depolama hizmeti yaygın olarak kullanılan ArcGIS Pro veya QGIS masaüstü uygulamaları tarafından veri düzenleme ve sorgulama için doğrudan bağlantıyı desteklemelidir.

4. Güvenlik:
    a. Masaüstü CBS uygulaması sunucusuz ilişkisel veri tabanı hizmetine güvenli ağ üzerinden erişmelidir.
    b. Vektör karo sunucusuna HTTPS protokolü üzerinden yetkili kullanıcılar erişebilmelidir.
5. Sistemdeki uygulamaların sürümleri önce test sonra yayın ortamına alınmalıdır. Test ortamında hata görülen yeni sürüm yayınlanmamalıdır.
6. Geliştirilen kodlar kod deposunda saklanmalı ve sürüm yayınlama otomasyonu kurulmalıdır.

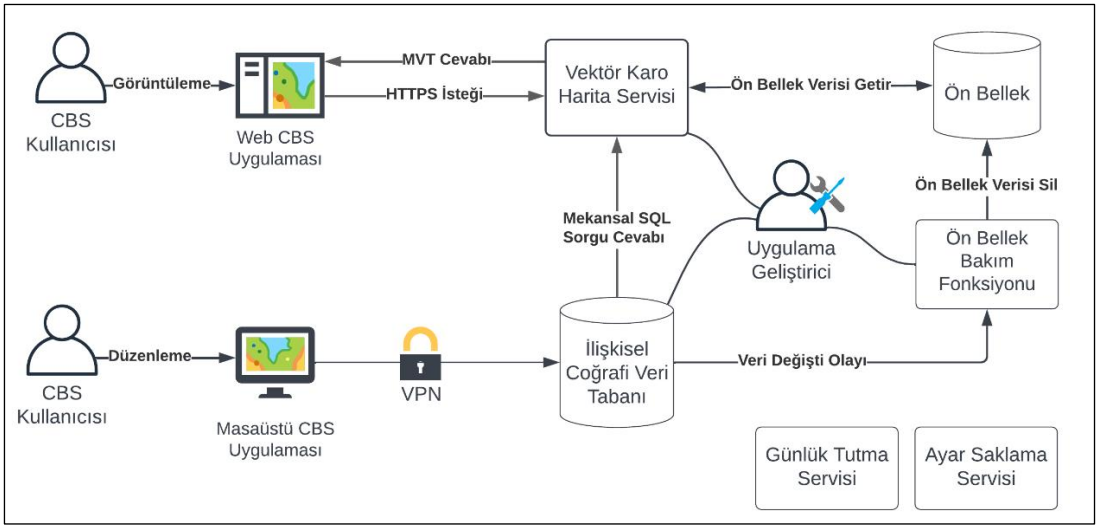
Roller

Geliştirilen sistem mimarisinde iki kullanıcı rolü bulunmaktadır. CBS kullanıcısı rolü sistem üzerinde CBS web ve masaüstü uygulamalarının kullanıcısıdır. Bu roldeki kullanıcı web veya masaüstü uygulamalarından sadece birinin de kullanıcısı olabilir. Masaüstü CBS kullanıcısı ilişkisel veri tabanına güvenli şekilde bağlanmak için VPN (Virtual Private Network) aracının da kullanıcısıdır. Aynı şekilde masaüstü CBS kullanıcısı ilişkisel coğrafi veri tabanı üzerinde doğrudan sorgulama, ekleme ve güncelleme işlemlerini yapabilmek için gerekli yetkilere sahiptir.

Uygulama geliştiricisi rolü ise vektör karo servisi tasarımındaki uygulama bileşenlerini geliştirir. Sistem sunucusuz hizmetlerle çalışacağından bu roldeki kullanıcının ileri düzey bulut bilişim altyapı yönetimi bilmesi gerekmemektedir. Uygulamalar seçilen sunucusuz hesaplama hizmetinin destek verdiği tüm dillerde geliştirilebilir. Uygulama geliştirici kod geliştirmelerini kod deposu üzerinde arşivler. Uygulama geliştiricinin kod depolarına erişim yetkisi bulunmaktadır. Kod deposundan yayın platformuna kadar olan süreci kurgulayarak derleme, test ve yayın otomasyonlarını sağlamalıdır. Uygulama geliştirici sistemi test ve yayın ortamı olmak üzere iki farklı ortamda yayınlar. Uygulama geliştirici sistemdeki performans metrikleri ve günlük kayıtlarının tutulduğu bileşenin de kullanıcısıdır. Bu bileşen üzerinden sistemin performansını ve sağlığını takip eder.

Sistem tasarımı ve bileşenleri

Sistem tasarımı verilen senaryo, kullanıcı rolleri ve gereksinimlere göre Şekil 4.5’te gösterilen şekilde tasarlanmıştır.



Şekil 4.5 : Vektör karo harita servisi sistemi tasarımı ve bileşenleri.

Vektör karo harita servisi bileşeni ilişkisel coğrafi veri tabanı ve ön bellek bileşenleri ile çalışarak harita yayını yapmaktadır. Web CBS uygulamasından HTTPS protokolü üzerinden gelen vektör karo isteklerini ilişkisel coğrafi veri tabanı sunucunun anlayacağı mekânsal SQL sorgularına dönüştürerek istenen bölgeye ait coğrafi verileri sorgulamaktadır. Sorgulama sonucu tabanı sunucusu cevabını MVT formatına dönüştürür ve PBF olarak kodlayarak bir vektör karoyu oluşturur.

Oluşturulan vektör karo verisi ön bellek depolama alanında saklanır. Bir sonraki istek yine aynı bölgeye aitse bu sefer veri tabanı sunucu sorgulanmadan veri önbellek üzerinden getirilir. Böylece daha hızlı bir şekilde cevap üretilmiş olur.

Ön bellek bakım fonksiyonu veri tabanı üzerindeki değişikliklere bağlı şekilde olay güdümlü olarak çalışmaktadır. Veri tabanında yapılan değişikliklerin ön bellekteki hangi vektör karolara karşılık geldiğini hesaplayarak o vektör karo verilerini siler. Böylece vektör karo harita servisi sunucusu bir sonraki aynı bölgeye ait vektör karo isteğinde ön bellekte bulamadığı vektör karoyu yeniden güncel veriyle üretir. Ön bellek temizleme sürecinde meydana gelebilecek bir hata durumunda veri tabanı sunucusunda ek bir tabloda hatalı işlemler kaydedilerek saklanmaktadır. Uygulama geliştirici hatanın düzeltilmesinden sonra bu tablodaki işlemleri ön bellek temizleyici fonksiyona tekrarlattırarak veri kaybı yaşanmamasını sağlamaktadır.

Web CBS uygulaması mimari tasarımda vektör karo harita servisi istemcisi olarak konumlandırılmıştır. Bu uygulama MVT formatını ve vektör karo servislerini desteklemektedir. Böylece CBS kullanıcısı coğrafi veriye vektör karo servisi üzerinden ulaşarak uygulama tarafından ekrana çizilen haritayı kullanabilmektedir.

Web CBS uygulaması vektör karo harita servisinden gelen coğrafi objelerin kartografik görselleştirilmesini de yapmaktadır. Bunun için kartografik görselleştirme kuralları önceden Web CBS uygulamasına kullanılan istemci teknolojisine uygun olarak kodlanmalıdır.

Masaüstü CBS uygulaması ilişkisel coğrafi veri tabanına bağlanabilmektedir. Bu uygulama ile CBS kullanıcısı coğrafi verileri görüntüleyebilir, sorgulayabilir, yeni coğrafi objeler ekleyebilir ve mevcut coğrafi objelerin geometri ile özniteliklerini güncelleyebilir.

CBS istemci uygulamalarından gelen isteklere CBS kullanıcısının yetkilerine göre erişim izini verilir. Kullanıcının yetkisi olmadığı istekler reddedilir. Bu amaçla hangi vektör karo harita servisine hangi kullanıcı veya kullanıcı grupları için erişim yetkisi verileceği bilgisi başka bir servis veya sunucusuz yapısal veri tabanı üzerinde saklanabilir. Bu tasarımda yetkilendirme sisteminin detayları tasarımı yalın ve anlaşılır tutmak için verilmemiştir. Literatürdeki herhangi bir yetkilendirme sistemi bu mimari üzerinde uygulanabilir.

Uygulama geliştiricinin vektör karo harita servisi ve ön bellek bakım fonksiyonunu geliştirmesi ve yayınlaması beklenmektedir. Bununla birlikte ilişkisel veri tabanı üzerindeki değişikliklerin ön bellek bakım fonksiyonunu tetiklemesi için ilgili geliştirmeleri de veri tabanı sunucusu üzerinde yapmalıdır.

Günlük tutma servisi sistem tasarımındaki bileşenlerde meydana gelen hata ve diğer günlük kayıtlarını tutar. Uygulama geliştirici hata kayıtlarını okuyarak uygulamalardaki hataların kök nedenlerini bularak çözebilmektedir. Bu servis ayrıca sistem bileşenlerine ait metrikleri de tutarak performans analizlerinin yapılmasını sağlamaktadır. Uygulama geliştirici bu bilgileri değerlendirerek uygulamaların iyileştirilmesi için kararlar almaktadır.

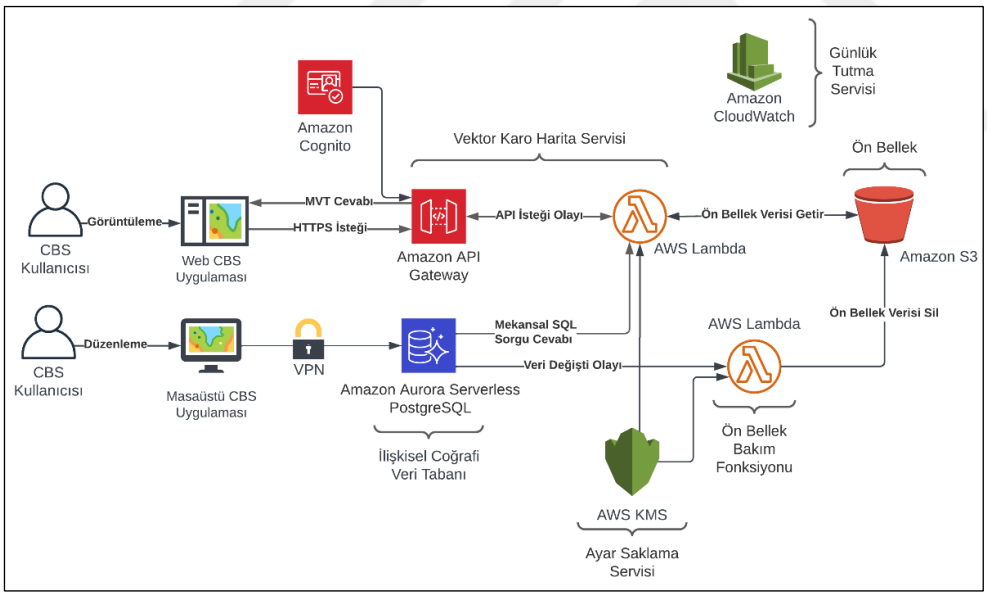
Ayar saklama servisi uygulamaların veri tabanı ve ön bellek destek servislerine bağlanama bilmesi için bağlantı adreslerini ve kimlik bilgilerini güvenli bir şekilde saklamaktadır. Ayar saklama servisi uygulama geliştiricinin sürüm yayın otomasyonu üzerinden gerektiğinde yeni bağlantı parametreleri ile güncellenebilir.

4.2.2 Uygulama

Sunulan sistem tasarımı Şekil 4.6'te gösterildiği şekilde AWS bulut bilişim sağlayıcısına ait sunucusuz hizmetlerle uygulanmıştır.

Bu uygulamada kullanılan bulut sağlayıcısı platformunun seçiminde karar verici olarak AWS ve Microsoft Azure tarafından sunulan sunucusuz ilişkisel veri tabanı teknolojileri rol oynamıştır.

Mimari tasarımın gereksinimlerine uygun olarak seçilen Tegola isimli vektör karo harita servisi uygulaması sadece ilişkisel veri tabanı olarak sadece PostgreSQL yazılımını desteklemektedir (URL-3). Bu veri tabanı teknolojisi AWS üzerinde Amazon Aurora Serverless isimli servis ile sunucusuz olarak sunulmaktadır. Bu servisin platform seçiminde karar vermede rol oynayan bir diğer özelliği de veri değişikliklerinde ürettiği olaylarla AWS Lambda sunucusuz hesaplama servisini çalıştırmak üzere tetikleyebilmesidir (Amazon, 2022b). Bu özellikler Microsoft Azure'da bulunmamaktadır.



Şekil 4.6 : Vektör karo harita servisinin AWS bulut bilişim sağlayıcısı üzerindeki sunucusuz hizmetlerle uygulaması.

Açık kaynak kodlu Tegola vektör karo sunucusu yazılımı tezde vektör karo sunucusu olarak kullanılmak üzere seçilmiştir. Tegola, Go programlama dilinde geliştirilmiştir. Vektör karolarını MVT tanımlamasına göre PBF ile kodlayarak oluşturmaktadır. Tegola yazılımı vektör karolar için veri kaynağı olarak PostgreSQL ilişkisel veri tabanına ve GeoPackage mekânsal dosya tabanlı veri tabanı formatına destek

vermektedir. Yazılımın ayrıca ön bellekleme yeteneği de bulunmaktadır. Ön belleğe alınan verilerin saklanması için Amazon S3 sunucusuz veri depolama hizmetini ve Redis ön bellek sunucusu yazılımını kullanabilmektedir. Bu yazılım HoF modelinde çalışan AWS Lambda üzerinde çalışabilmektedir (Fitzsimmons, 2017). İstemcilerden gelen HTTPS isteklerini alarak AWS Lambda servisine ileten Amazon API Gateway isimli internet trafiği kontrol ve yönlendiricisi olarak görev yapan sunucusuz destek servisine de ihtiyaç duyulmuştur. Bu servis gelen HTTPS isteklerinin birer tetikleyici olaya dönüştürülerek AWS Lambda fonksiyonuna iletilmesinde ve fonksiyon ile istemciler arasındaki trafiğin sağlıklı bir şekilde yönetilmesinde kullanılmaktadır (Patterson, 2019). CBS kullanıcılarının vektör karo harita servisine yetkilendirilmiş bir şekilde ulaşabilmesi için Amazon Cognito sunucusuz kimlik yönetim sistemi kullanılmıştır.

Ön bellek verilerini saklamak için AWS üzerinde sunulan sunucusuz veri depolama hizmeti Amazon S3 kullanılmıştır. Ön bellek üzerindeki güncelliğini yitiren vektör karo verilerinin tespiti ve silinmesi için bir başka AWS Lambda üzerinde çalışan uygulama konumlandırılmıştır. Bu uygulama veri tabanından gelen veri değişikliklerini dinleyerek Amazon S3'ün sunduğu API üzerinden eski vektör karo verilerini silmektedir. Böylece kaynak veride yapılan değişiklikler gerçek zamana yakın bir şekilde anında ön bellekten silinmektedir.

Sistemdeki bileşenler arasında iletişim için yetkilendirmeler AWS Identity and Access Management (IAM) hizmeti üzerinden bileşenlere rol ataması yapılarak sağlanmıştır. Örneğin ilişkisel coğrafi veri tabanı bileşeni ön bellek temizleyicisi fonksiyonuna erişerek onu tetikleyebilmektedir ancak ön bellek temizleyicisi fonksiyon ilişkisel coğrafi veri tabanı bileşenine erişememektedir. AWS KMS hizmeti uygulamaların veri tabanına erişim için kullandıkları kimlik bilgilerini güvenli bir şekilde saklamak için ayar saklama servisi olarak kullanılmıştır. Günlük tutma servisi olarak ise Amazon CloudWatch hizmeti ile tüm bileşenler takip edilerek ve hata günlükleri tutularak izlenmektedir.

4.2.3 Değerlendirme

Uygulanan sistem tasarımı ve uygulaması Çizelge 4.2'de gösterildiği gibi 12 Faktör yöntemine göre değerlendirilmiştir. Sunulan sistemde sadece HoF modelindeki sunucusuz hesaplama servisleri kullanılmıştır. Bu nedenle 12 Faktör yöntemindeki

“Bağımlılıklar”, “Süreçler”, “Port Bağlama” ve “Yönetim Süreçleri” prensipleri doğrudan uygulanamadığı için değerlendirilmemiştir.

Çizelge 4.2 : Vektör karo harita servisinin 12 Faktör yöntemine göre değerlendirilmesi.

Prensip Adı	Değerlendirme
Kod Deposu	Uygulama geliştirici geliştirdiği uygulamaları kod deposu üzerinde arşivlemektedir.
Ayarlar	Sistem tasarımındaki uygulamaların gizli ayarları ayar saklama servisi üzerinde saklanmıştır. Gizli olmayan ayarlar ise uygulamaların çalıştığı konteynerlerin ortam değişkenleri üzerinde tutulmuştur. Sistem tasarımının uygulamasında ise AWS KMS servisi gizli ayarların güvenli bir şekilde saklanması için kullanılmıştır.
Destek Servisleri	Uygulamalar veri tabanı ve yapısal olmayan veri deposu hizmetlerine ağ üzerinden bağlanarak ulaşmıştırlar.
Derle, Yayınla ve Çalıştır	Uygulama geliştirici sürüm yayınlama süreçlerinin otomasyonunu kurgulamaktan sorumludur. Sistem tasarımındaki uygulamaların bu otomasyonla yeni sürümleri yayınlanmaktadır.
Eş Zamanlılık	Uygulamalar HoF modelinde çalıştırılmak üzere geliştirildiği için eş zamanlı çalışabilirler. Eş zamanlı çalışmanın yönetimi bulut bilişim sağlayıcısı tarafından yapılmaktadır.
Kullanılabilirlik	Sistem herhangi bir süreçte hata olması durumunda yeniden süreci tekrar edecek şekilde tasarlanmıştır. Ayrıca uygulamalar hızlı başlayacak ve kapanacak şekilde tasarlanmıştır. Uygulamada vektör karo servisi için seçilen Tegola yazılımı Go programlama dili ile geliştirilmiştir ve bu dil AWS Lambda üzerinde en hızlı çalışan dillerden biridir (Jackson ve Clynch, 2018).
Geliştirme, Test ve Yayın Ortamlarının Benzerliği	Uygulama geliştirici sistem tasarımını test ve yayın ortamlarında birbirlerine benzeyecek şekilde yayınlar. Geliştirmeler uygulama geliştiricinin bilgisayarında yapılmaktadır.
Günlük Tutma	Sistem bileşenleri tarafından üretilen günlük kayıtları için günlük tutma servisi bileşeni kullanılmıştır. Uygulamada ise Amazon CloudWatch sunucusuz günlük tutma servisi kullanılmıştır.

Sistem tasarımı ayrıca mimari değerlendirme ölçütlerine göre Çizelge 4.3'de değerlendirilmiştir.

Çizelge 4.3 : Vektör karo harita servisinin bulut bilişim sağlayıcısı mimari değerlendirme ölçütlerine göre değerlendirilmesi.

Ölçüt Adı	Değerlendirme
Performans Verimliliği	12 Faktör yönteminin "Kullanılabilirlik" prensibine uygun olarak uygulamalar en kısa sürede işlemleri tamamlamak üzere tasarlanmıştır.
Güvenilirlik	Sistem bileşenleri günlük tutma servisi ile izlenerek hatalar takip edilmektedir. Bununla birlikte hatalara karşı süreçlerin kaybolmaması için ek önemler tasarlanmıştır.
Güvenlik	Sistemde kullanıcı ve bileşenlerin yetkileri detaylı bir şekilde tanımlanmıştır. Veri tabanı sunucusunun masaüstü CBS uygulaması ile iletişimi güvenli ağ üzerinden yapılmaktadır. Web CBS uygulaması vektör karo servisi ile yetkilendirilmiş HTTPS istekleriyle iletişim kurmaktadır.

4.3 Sunucusuz Raster Karo Harita Servisi Tasarımı, Uygulaması ve Değerlendirmesi

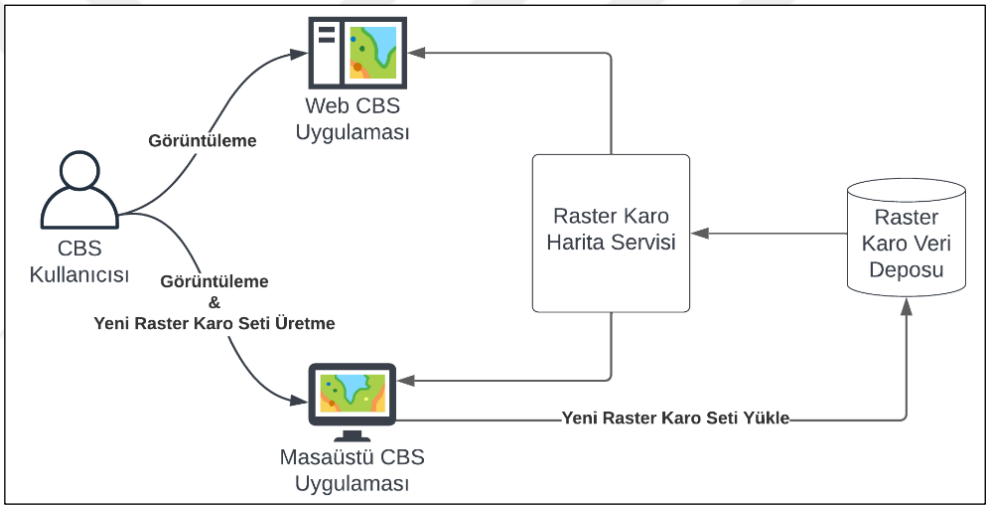
Her ne kadar günümüz web CBS uygulamaları ve web haritaları vektör harita servislerini daha çok tercih etseler de raster veri formatı CBS uygulamalarında halen en çok kullanılan veri formatlarından biridir (Netek ve diğ, 2020). CBS sunucusu yazılımlar raster karo harita servisi sunabilirler. OGC raster harita servisleri için WMTS ve WMS (Web Map Service) standartlarını tanımlamıştır (OGC, 2010; OGC, 2006). WMS standardına dayanan harita servisleri güncel coğrafi veriyi kullanarak raster haritayı dinamik olarak üretip sunabilmektedirler (Souissi ve Mainguenaud, 2014). Kaynak veri vektör ise CBS sunucusu tarafından kartografik görselleştirme kuralları ile harita üretilerek raster formatta servis edilmektedir. WMTS ile çalışan raster harita servisleri ise raster karolar halinde önceden üretilmiş haritaları servis etmek için kullanılmaktadırlar. Bu nedenle WMS raster harita servisleri güncel verilerin gösterimine ihtiyaç olan senaryolarda tercih edilmektedir. WMS raster harita servislerinin dezavantajı ise CBS sunucusu üstünde kartografik üretimi her istekte yaparak sunucuya ek iş yükü oluşturmasıdır (Blower, 2010). Bu nedenle çok sık değişmeyen coğrafi verilerin raster olarak servis edilmesinde WMTS raster karo harita servisleri tercih edilmektedir.

Bu tez çalışmasında sunucusuz hesaplama ve veri depolama hizmetleri kullanılarak raster karo harita servisinin sistem tasarımı geliştirilmiş ve sunulmuştur. Bu tasarım da raster karo harita servisi standardı olarak WMTS kullanılmıştır.

4.3.1 Tasarım

Senaryo

Sistem mimarisi tasarımında kullanılan senaryo CBS web ve masaüstü uygulaması kullanıcılarının harita altlığı olarak bir raster karo harita servisini güvenli bir şekilde kullanabilmeleri üzerinde kurgulanmıştır (Şekil 4.7). Aynı zamanda masaüstü CBS kullanıcıları yeni raster karo setlerini üretip raster veri deposuna yükleyerek yeni harita servisleri oluşturabilirler.



Şekil 4.7 : Raster karo harita servisinin kullanım senaryosu.

Gereksinimler

Senaryoda verilen kurgunun sunucusuz bir sistem mimarisi tarafından gerçekleştirilebilmesi için gereksinimler belirlenmiştir. Raster karo harita servisi tasarımı aşağıdaki gereksinimlere göre geliştirilmiştir.

1. Raster karo harita servisi sunucusu yazılımı sunucusuz hesaplama hizmetleri üzerinde çalışmalıdır.
2. Raster karo harita servisi sunucusu yazılımı OGC WMTS standartlarında harita yayını yapabilmelidir.
3. Raster karo dosyalar sunucusuz depolama alanında depolanabilmelidir.
4. Raster karo harita servisine HTTPS protokolü üzerinden yetkili kullanıcılar erişebilmelidir.

5. CBS kullanıcıları ürettikleri raster karo veri setlerini depolama alanına yükleyebilmelidir.
6. CBS kullanıcıları yüklenen yeni raster karo veri seti ile yeni WMTS harita servisi yayını yapabilmelidirler.
7. Sistemdeki uygulamaların sürümleri önce test sonra yayın ortamına alınmalıdır. Test ortamında hata görülen yeni bir sürüm yayınlanmamalıdır.
8. Geliştirilen kodlar kod deposunda saklanmalı ve sürüm yayınlama otomasyonu kurulmalıdır.

Roller

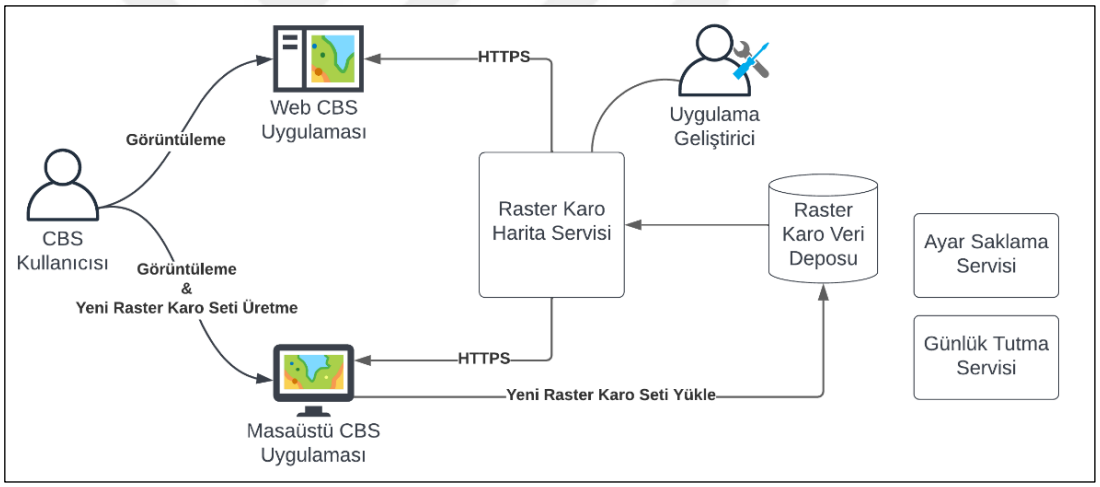
Geliştirilen sistem mimarisinde iki kullanıcı rolü bulunmaktadır. CBS kullanıcısı rolü sistem üzerinde CBS web ve masaüstü uygulamalarının kullanıcısıdır. Bu roldeki kullanıcı web veya masaüstü uygulamalarından sadece birinin de kullanıcısı olabilir. Masaüstü CBS kullanıcısın ayrıca raster veri deposuna da erişimi bulunmaktadır. Masaüstü CBS kullanıcısı kullandığı CBS araçları ile ürettiği raster karo veri setini raster karo veri deposuna yükleyebilmelidir.

Uygulama geliştiricisi rolü ise raster karo servisi tasarımındaki uygulama bileşenlerini geliştirir. Sistem sunucusuz hizmetlerle çalışacağından bu roldeki kullanıcının ileri düzey bulut bilişim altyapı yönetimi bilmesi gerekmemektedir. Uygulamalar, seçilen sunucusuz hesaplama hizmetinin destek verdiği tüm dillerde geliştirilebilir. Uygulama geliştirici yaptığı geliştirme çalışmalarını kod deposu üzerinde arşivlemelidir ve bu kod deposuna erişebilmelidir. Kod deposundan yayına kadar olan sürüm yayınlama süreçlerini kurgulayarak otomasyonu uygulama geliştirici sağlar. Uygulama geliştirici sistemi test ve yayın ortamı olmak üzere iki farklı şekilde yayınlar. Uygulama geliştirici ayrıca CBS kullanıcısının yeni bir WMTS servisini nasıl yayına alacağını adım adım açıklayan bir dokümantasyonunu da oluşturmalı ve paylaşmalıdır. Uygulama geliştirici isterse WMTS servisinin kolayca yayınlanabilmesi için yardımcı bir araç da CBS kullanıcıları için geliştirebilir. Böylece CBS kullanıcısının teknik detaylara hâkim olması gerekmeden kolayca yeni harita servisi yayınlayabilir. Uygulama geliştirici sistemdeki performans metrikleri ve günlük kayıtlarının tutulduğu bileşenin de kullanıcısıdır. Bu bileşen üzerinden sistemin performansını ve sağlığını takip eder.

Sistem tasarımı ve bileşenleri

Sistem tasarımı verilen senaryo, kullanıcı rolleri ve gereksinimlere göre Şekil 4.8'te gösterilen şekilde tasarlanmıştır.

Raster karo harita servisi bileşeni raster veri deposu bileşeninde bulunan verileri kullanarak harita yayını yapmaktadır. Raster karo harita servisi uygulaması Web CBS uygulamasından HTTPS protokolü üzerinden gelen WMTS standardındaki raster karo istekleri işleyerek raster karo veri deposundan istenen coğrafi alana düşen verileri çekerek cevap oluşturur. Eğer gelen isteği yapan CBS kullanıcısının talep edilen harita servisine yetkisi yoksa istek reddedilir. Bu amaçla hangi harita servisinin hangi kullanıcı veya kullanıcı grupları için erişim yetkisi verileceği bilgisi başka bir servis veya sunucusuz yapısal veri tabanı üzerinde saklanabilir. Bu tasarımda yetkilendirme sisteminin detayları tasarımı yalın ve anlaşılır tutmak için verilmemiştir. Literatürdeki herhangi bir yetkilendirme sistemi bu mimari üzerinde uygulanabilir.



Şekil 4.8 : Raster karo harita servisi sistemi tasarımı ve bileşenleri.

Web CBS ve masaüstü CBS uygulamaları mimari tasarımda raster karo harita servisi istemcisi olarak konumlandırılmıştır. Bu uygulamalar WMTS standardındaki harita servislerini desteklemektedir. Böylece CBS kullanıcısı raster karo servisi üzerinden harita verilerine ulaşarak servis edilen haritayı kullanabilmektedir.

Masaüstü CBS uygulaması kullandığı CBS araçları ile ürettiği raster karo veri setini raster karo veri deposuna doğrudan veya yardımcı araçlarla yükleyebilmektedir. Yeni raster veri seti yüklendikten sonra CBS kullanıcısı ilgili ayarları yaparak bu verileri yeni bir WMTS servisi olarak yayınlamaktadır.

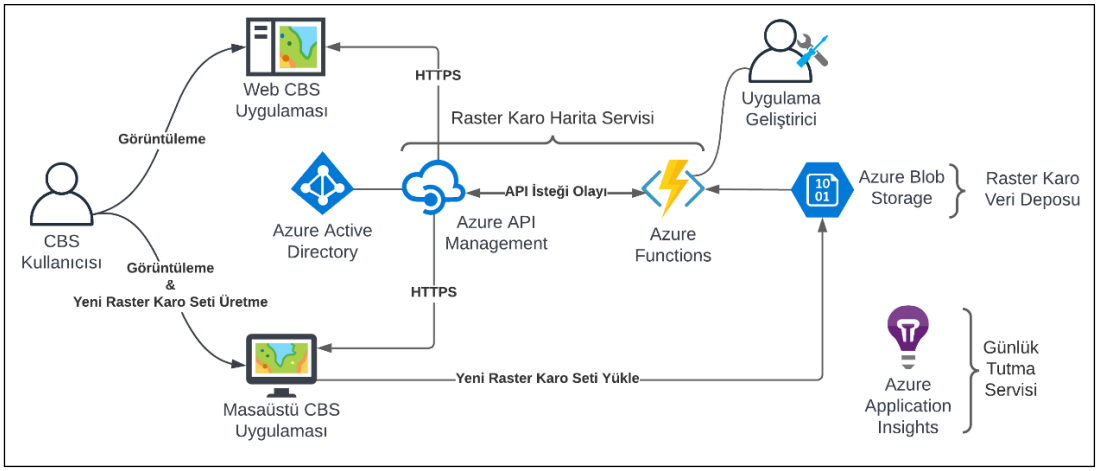
Uygulama geliştiriciden raster karo harita servisi fonksiyonunu geliştirmesi ve yeni sürümlerini gerektiğinde yayınlaması beklenmektedir. Bununla birlikte uygulama geliştirici diğer sistem bileşenlerinin yapılandırmalarının da doğru şekilde yapıldığından emin olmak için çeşitli testler yürütebilir. Bu testlerin sonuçlarını değerlendirmede günlük tutma servisinden faydalanabilir.

Günlük tutma servisi sistem tasarımındaki bileşenlerde meydana gelen hata ve diğer günlük kayıtlarını tutar. Uygulama geliştirici bu hata günlük kayıtlarını okuyarak uygulamalardaki hataların kök nedenlerini bularak çözebilmektedir. Bu servis ayrıca sistem bileşenlerine ait metrikleri de tutarak performans analizlerinin yapılmasını sağlamaktadır. Uygulama geliştirici bu bilgileri değerlendirerek uygulamaların iyileştirilmesi için kararlar almaktadır. Uygulama geliştirici ayrıca ayar saklama servisinde saklanan ayarların gizliliğinden ve güvenliğinden de sorumludur.

Ayar saklama servisi raster karo harita servisinin, raster karo veri deposu ve diğer destek servislerine bağlanabilmesi için bağlantı adreslerini ve kimlik bilgilerini güvenli bir şekilde saklamaktadır. Ayar saklama servisindeki bilgiler sürüm yayın otomasyonu üzerinden gerektiğinde yeni bağlantı parametreleri ile güncellenebilir.

4.3.2 Uygulama

Raster karo harita servisi sistem tasarımı Şekil 4.9'da gösterildiği şekilde Microsoft Azure bulut bilişim sağlayıcısına ait sunucusuz hizmetlerle uygulanmıştır.

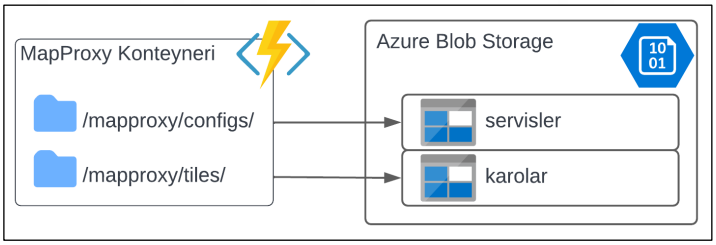


Şekil 4.9 : Raster karo harita servisinin Microsoft Azure bulut bilişim sağlayıcısı üzerindeki sunucusuz hizmetlerle uygulaması.

Raster karo harita servisi sunucusu yazılımı için açık kaynak kodlu MapProxy yazılımı kullanılmıştır (URL-4). Bu yazılım Python programlama dilinde geliştirilmiştir ve bu

dil Azure Functions tarafından desteklenmektedir. MapProxy yazılımının sunucusuz mimaride çalıştırılması için Azure Functions sunucusuz hesaplama hizmeti kullanılmıştır. Bu hizmet için Linux işletim sistemi seçilmiştir. Python web sunucusu uygulamaları için geliştirilen WSGI (Web Server Gateway Interface) standardı MapProxy tarafından da kullanılmaktadır. Azure Functions'a gelen HTTPS isteklerinin WSGI standardına dönüştürülebilmesi için yardımcı kütüphaneler kullanılarak MapProxy HoF modeli üzerinde çalıştırılabilmektedir (Bitner, 2019).

Raster karo veri deposu olarak Azure Blob Storage yapısal olmayan sunucusuz veri depolama hizmeti kullanılmıştır. Raster karo harita servislerinin MapProxy'nin tanımladığı formatta saklanabilmesi için Azure Blob Storage üzerinde "servisler" isimli konteyner açılmıştır. Üretilen raster karo setlerinin saklanabilmesi için de "karolar" isimli konteyner açılmıştır. Bu iki Azure Blob Storage konteyneri Azure Functions üzerinde çalışan Linux işletim sistemine dosya sistemindeki iki klasörle ayrı ayrı bağlanmıştır (Şekil 4.10). Bu sayede MapProxy yazılımı Azure Blob Storage desteği olmadan da sunucusuz veri depolama hizmetini dosya sistemi üzerinden kullanabilmektedir. Microsoft Azure bulut bilişim sağlayıcısının bu sistem tasarımı için seçilmesinde bu özellik karar verici olmuştur.



Şekil 4.10 : Azure Blob Storage konteynerlerinin Azure Functions konteynerinin dosya sistemine bağlanması.

Web CBS ve masaüstü CBS uygulamaları raster karo harita servisine güvenli ve yetkilendirilmiş bir şekilde ulaşabilmektedir. Bunun sağlanabilmesi için raster karo harita servisi sunucusu ile istemci uygulama arasında sunucusuz API yönetimi servisi kullanılmıştır. Microsoft Azure bulut bilişim sağlayıcısında sunucusuz API yönetim servisinin adı Azure API Management olarak geçmektedir. Bu servis gelen HTTPS isteklerini Azure Functions olaylarına dönüştürerek fonksiyonunun HTTP olaylarına güdümlü olarak çalışmasını sağlar.

Microsoft Azure bulut bilişim sağlayıcısındaki sunucusuz kullanıcı ve rol yönetimi servisi Azure Active Directory olarak adlandırılmaktadır. Kullanıcı oturum açma,

yetkilendirme ve rol yönetimi işlemleri Azure Active Directory üzerinden yönetilebilmektedir. Azure API Management ile Azure Active Directory hizmeti birlikte kullanılmıştır. İstemci uygulamalardan gelen HTTPS istekleri Azure Active Directory yardımı ile kontrol edilerek yetkilendirilmektedir.

Sistemdeki bileşenler arasında iletişim için yetkilendirmeler Azure Managed Identities adlı kimlik sistemi ile sağlanmıştır. Raster karo harita servisinin üzerinde çalıştığı Azure Functions'a atanan kimlik bilgisi raster karo veri deposu bileşenine erişim için yetkilendirilmiştir. Bu özellik sayesinde sistem tasarımında sunulan ayar servisi bileşeni için ek bir servisin kullanılmasına gerek kalmamıştır.

Günlük tutma servisi olarak Azure Application Insight servisi kullanılmıştır. Bu servis merkezi bir şekilde günlük kayıtlarını ve performans metriklerini tüm sistem bileşenlerinden toplayabilmektedir. Sistem bileşenlerinin ve uygulamaların günlük ve metrik toplama ayarları uygula geliştirici tarafından yapılarak günlük tutma servisinin çalışması sağlanmaktadır. Uygulama geliştirici bu servis üzerinden sistemi takip ederek hata düzeltme ve iyileştirmeler için gerekli aksiyonları alabilmektedir.

4.3.3 Değerlendirme

Sunulan sistem tasarımı ve uygulaması Çizelge 4.4'te gösterildiği gibi 12 Faktör yöntemine göre değerlendirilmiştir. Geliştirilen sistemde sadece HoF modelindeki sunucusuz hesaplama servisleri kullanılmıştır. Bu nedenle 12 Faktör yöntemindeki "Bağımlılıklar", "Süreçler", "Port Bağlama" ve "Yönetim Süreçleri" prensipleri doğrudan uygulanamadığı için değerlendirilmemiştir.

Çizelge 4.4 : Raster karo harita servisi sisteminin 12 Faktör yöntemine göre değerlendirilmesi.

Prensip Adı	Değerlendirme
Kod Deposu	Uygulama geliştirici geliştirdiği uygulamaları kod deposu üzerinde arşivlemektedir.
Ayarlar	Sistem tasarımında ayarların saklanması için ayar saklama servisi bileşeni kullanılmıştır. Uygulamada ise ayarların saklanması için ek bir servis kullanımına ihtiyaç olmamıştır. Microsoft Azure bulut bilişim sağlayıcısının sunduğu Managed Identities ile destek hizmetlerine güvenli bağlantı ek parametrelerin bir servis üzerinde saklanmasına ihtiyaç olmadan sağlanmıştır.
Destek Servisleri	Sistem tasarımında raster karo servisi yapısal olmayan veri deposuna ağ üzerinden ulaşmaktadır. Uygulamada ise yapısal olmayan veri deposu hizmetine Azure Functions tarafından sunulan dosya sistemi bağlama üzerinden ulaşılmıştır. Böylece bu bağlantının da platform sağlayıcı tarafından yönetilmesi sağlanarak prensip ihlal edilmemiştir.
Derle, Yayınla ve Çalıştır	Uygulama geliştirici sürüm yayınlama süreçlerinin otomasyonunu sistem bileşenlerinin yeni sürümlerini veya yeni yapılandırma ayarlarını yüklemek için kullanır.
Eş Zamanlılık	Uygulama için HoF modelindeki sunucusuz hesaplama hizmeti seçilmiştir. HoF modelinde çalışan uygulamalar eş zamanlı olacak şekilde platform tarafından çalıştırılmaktadır.
Kullanılabilirlik	Raster karo servisi senkron istek-cevap (request-reply) modeline göre HTTPS protokolü üzerinden çalışmaktadır. İstemciden gelen bir isteğe zaman aşımına uğramadan hızlı şekilde cevap verebilmesi için hızlı başlamalı ve kapanmalıdır. Seçilen MapProxy uygulaması Python ile geliştirilmiştir. Python, Linux tabanlı Azure Functions üzerinde en hızlı tepki veren dillerden biridir (Maissen ve diğ, 2020).
Geliştirme, Test ve Yayın Ortamlarının Benzerliği	Uygulama geliştirici sistem tasarımını test ve yayın olacak şekilde iki farklı ortamda ve bu ortamlar birbirlerine benzeyecek şekilde yayınlar. Geliştirmeler uygulama geliştiricinin bilgisayarında yapılmaktadır.
Günlük Tutma	Sistem bileşenleri tarafından üretilen günlük kayıtları için günlük tutma servisi bileşeni eklenmiştir. Uygulamada ise Azure Application Insight günlük tutma servisi kullanılmıştır.

Sistem tasarımı ayrıca mimari değerlendirme ölçütlerine göre Çizelge 4.5'de değerlendirilmiştir.

Çizelge 4.5 : Raster karo harita servisinin bulut bilişim sağlayıcısının mimari değerlendirme ölçütlerine göre değerlendirilmesi.

Ölçüt Adı	Değerlendirme
Performans Verimliliği	12 Faktör yönteminin "Kullanılabilirlik" prensibine uygun olarak raster karo servisi için seçilen açık kaynak uygulama sistem mimarisi üzerinde en kısa sürede ve hızlı çalışacak şekilde seçilmiştir.
Güvenilirlik	Sistem bileşenleri günlük tutma servisi ile izlenerek hatalar takip edilmektedir.
Güvenlik	Sistemde kullanıcı ve bileşenlerin yetkileri detaylı bir şekilde tanımlanmıştır. CBS kullanıcısının raster karo veri deposu ile iletişimi yetkilendirilmiştir. CBS istemci uygulamaları ile raster karo servisi ile yine yetkilendirilmiş HTTPS istekleriyle iletişim kurmaktadır.

4.4 Sunucusuz Mekânsal Zekâ Sistemi Tasarımı, Uygulaması ve Değerlendirmesi

Bulut bilişim teknolojilerinin gelişmesi yüksek hesaplama gücü gerektiren yapay zekâ uygulamalarının daha ulaşılabilir olmasını, kolay geliştirilmesini ve çalıştırılabilmesini sağlamaktadır. Bu gelişme farklı disiplinlerin konuya olan ilgisini de arttırmıştır. Yapay zekâ uygulamaları derin öğrenme ve makine öğrenmesi gibi alt kollara ayrılmaktadır. Bu alt uygulama alanları verinin bilgisayarlar tarafından analiz edilerek veri motiflerinin çıkarılmasını ve uygulamaların davranışlarının bu motiflere göre belirlenmesini sağlamaktadır. Merkezinde veri olan bir teknoloji ise mekânsal veriler olmadan düşünülemez.

Mekânsal bilişim teknolojileri çeşitli disiplinler arasında önemli gerçek dünya problemlerini analiz etmek, görselleştirmek ve anlamlandırmak amacıyla kullanılmaktadır (VoPham ve diğ, 2018). Mekânsal bilişim teknolojileri ve yapay zekâ uygulamalarının birlikte kullanıldığı uygulamalar mekânsal zekâ (GeoAI) uygulamaları olarak literatüre geçmiştir. Güney ve Çelik'e (2020) göre mekânsal zekâ şu şekilde tanımlanmıştır:

"Dünya üzerinde açık alanda ve/veya kapalı alanda gerçekleşen mekânsal veriyle ilişkili olguların ve insan faaliyetlerinin mekânsal modellemesinde, analizinde, görselleştirilmesinde, mekânsal problemlerin çözümünde ve mekânsal karar vermede yapay zekâ yöntemlerinin yukarıda ifade edilen ileri teknolojilerle birlikte toplum yararına kullanımı mekânsal zekâ olarak tanımlanabilir".

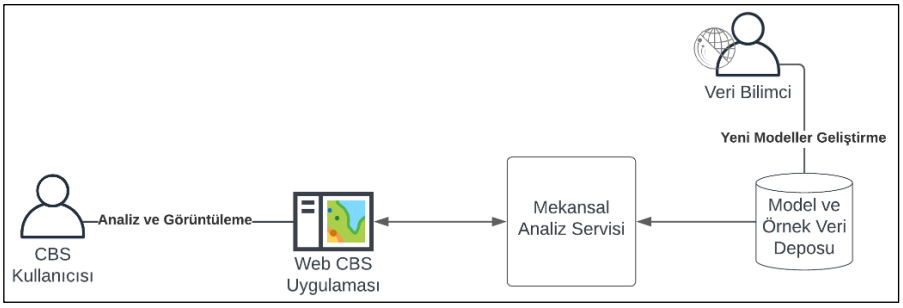
Mekânsal zekâ teknolojileri CBS uygulamaları içerisinde de kullanılmaktadır. Örneğin 2007 yılından beri CBS alanında yapılan SIGSPATAL konferansı 2017 yılından bu yana son beş yıldır da mekânsal zekâ üzerine sunulan bildirileri ayrı olarak yayınlamaktadır (URL-5).

Bu tez çalışmasında örnek bir mekânsal zekâ uygulamasının sistem tasarımı sunucusuz mimariye uygun şekilde tasarlanmıştır. Daha sonra bu tasarım bir bulut bilişim sağlayıcısı üzerinde uygulanarak değerlendirilmiştir.

4.4.1 Tasarım

Senaryo

Sistem mimarisi tasarımında kullanılan senaryoda CBS web kullanıcıları mekânsal analizler yapabilmek üzere makine öğrenmesi teknolojilerinden faydalanan bir mekânsal zekâ analiz sistemi kullanmaktadırlar (Şekil 4.11). Sistem servislerini OGC WPS üzerinden sunmaktadır. Böylece bu standardı destekleyen farklı CBS uygulamaları ile de kullanılabilmektedir. Bu senaryoda ayrıca veri bilimci makine öğrenmesine dayalı analizlerde kullanılan modelleri geliştirerek yeni verilerle devamlı olarak eğiterek sistemin güncelliğini ve devamlılığını sağlamaktadır.



Şekil 4.11 : Mekânsal analiz servisi sisteminin kullanım senaryosu.

Gereksinimler

Senaryoda belirtilen sistemin gereksinimleri belirlenmiştir ve bu gereksinimler sunucusuz mimari ile geliştirilen mekânsal analiz servisi tasarlanırken dikkate alınmıştır.

1. Mekânsal analiz servisi sistemi kapsamında geliştirilen uygulamalar sunucusuz hesaplama servisleri üzerinde çalışmalıdır.
2. Mekânsal analiz servisi birlikte çalışabilirliği artırabilmek için OGC'nin WPS standartlarında yayın yapabilmelidir.
3. Mekânsal analiz sonuçları kalıcı bir sunucusuz depolama alanı üzerinde geçici bir süre tutularak CBS kullanıcısı tarafından indirilebilmesi için hazır bekletilmelidir.
4. Eğitilmiş modeller ve örnek veriler bir sunucusuz depolama alanında tutulmalı ve mekânsal analiz servisi ve veri bilimci tarafından ulaşılabilir olmalıdır.
5. Veri bilimci yeni eğitilmiş modelleri sisteme yükleyebilmelidir.
6. Mekânsal analiz servisi HTTPS protokolü üzerinden yetkili kullanıcılar tarafından erişilebilmelidir.
7. Sistemdeki uygulamaların ve eğitilmiş modellerin sürümleri önce test sonra yayın ortamına alınmalıdır. Test ortamında hatası bulunan model veya uygulamanın yeni sürümü yayınlanmamalıdır.
8. Sistemdeki uygulamalar için geliştirilen kodlar kod deposunda saklanmalı ve sürüm yayınlama otomasyonu üzerinden yayınlanmalıdır.

Roller

Geliştirilen sistem mimarisinde CBS kullanıcısı, veri bilimci ve uygulama geliştirici olmak üzere üç kullanıcı rolü bulunmaktadır.

CBS kullanıcısı rolü sistem üzerinde CBS web uygulamasının kullanıcısıdır. Bu roldeki kullanıcı web CBS uygulaması üzerinden sistemde veri bilimci ve uygulama geliştirici tarafından hazırlanmış mekânsal analizleri çalıştırabilir. CBS kullanıcısı sadece kendisine yetki verilmiş mekânsal analizleri görebilir ve çalıştırabilir. CBS kullanıcısı OGC WPS standardında sunulan mekânsal analizi servisini dilerse bir başka masaüstü veya web CBS uygulamasında da kullanabilir. CBS kullanıcısı analiz sonucu ortaya çıkan vektör veya raster formattaki coğrafi veriyi indirebilir.

Uygulama geliştiricisi rolü mekânsal analiz servisinin geliştirilmesi ve devamlılığını sağlamakla sorumludur. Bu roldeki kullanıcı sunucusuz mimariye uygun uygulama geliştirir ve yayına alır. Sunucusuz mimarilerde altyapı yönetimi platform sağlayıcısı tarafından yapılmakta olduğu için uygulama geliştiricinin ileri düzey bulut bilişim altyapı yönetimi hakkında bilgi sahibi olması gerekmemektedir. Uygulama geliştirici platform üzerinde sağlanan sunucusuz hesaplama hizmetlerinin destek verdiği programlama dillerinden birini seçerek uygulamalarını geliştirebilir. Geliştirmeler sonucu üretilen kodlar kod deposu üzerinde arşivlenmelidir bu nedenle uygulama geliştiricin kod deposuna erişimi bulunmaktadır. Uygulama geliştirici arşivlenen kodların kod deposundan yayına kadar olan sürüm yayınlama süreçlerini kurgulayarak yayın otomasyonunu sağlar. Uygulama geliştirici sistemi test ve yayın ortamı olmak üzere iki farklı ortamda yayınlar. Uygulama geliştirici sistemdeki performans metrikleri ve günlük kayıtlarının tutulduğu bileşenin de kullanıcısıdır. Bu bileşen üzerinden sistemin performansını ve sağlığını takip eder.

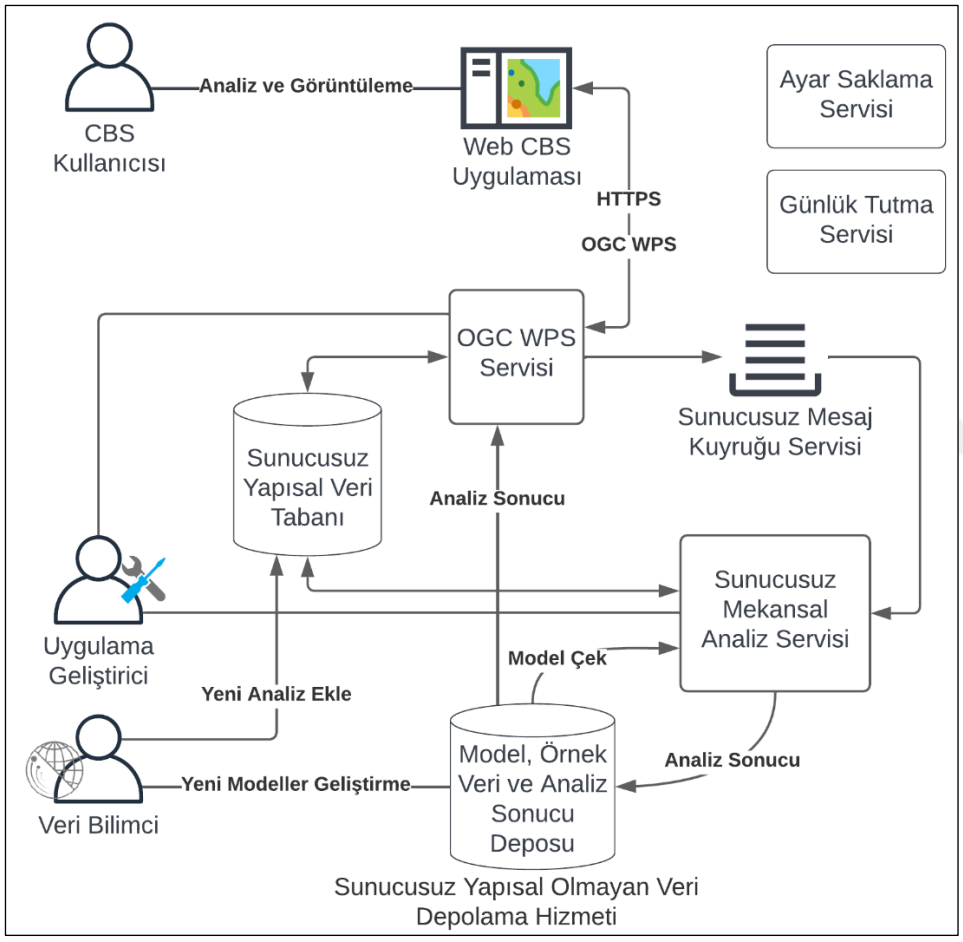
Veri bilimci rolündeki kullanıcı sistemdeki mekânsal analizlerin güncel ve yüksek doğrulukla çalışması için devamlı olarak güncellenen örnek verilerle modelleri geliştirir. Veri bilimci tarafından geliştirilen yeni modeller ve kodlar da kod deposu üzerinde arşivlenir. Veri bilimci ayrıca geliştirdiği modellerin doğruluğunun testi için ayrıca sürüm otomasyonu sistemine ek geliştirmeleri uygulama geliştirici ile yapabilir. Böylece doğruluk testleri de otomasyonunun bir parçası olarak yeni geliştirilen modeller yayın öncesi belirlenen ölçütlere göre otomatik olarak test edilmiş olur. Veri bilimci geliştirdiği yeni mekânsal veri analizlerini sisteme ekleyerek CBS kullanıcısı tarafından kullanılabilmesini sağlar. Bu amaçla yeni analizlere ilişkin dokümantasyon çalışmasını da yaparak CBS kullanıcıları ile paylaşır.

Sistem tasarımı ve bileşenleri

Sistem, verilen senaryo, kullanıcı rolleri ve gereksinimlere göre Şekil 4.12'de gösterilen şekilde tasarlanmıştır. Sistem tasarımı sunucusuz mimaride çalışacak şekilde geliştirilmiş ve bileşenleri açıklanmıştır.

Sistem tasarımında CBS kullanıcısının web CBS uygulaması üzerinden çalıştıracağı mekânsal veri analizleri mekânsal zekâ sistemine OGC WPS standardına uygun yayın yapan OGC WPS servisi bileşeni üzerinden iletilmektedir. Bu servis standartlara uygun bir şekilde istekleri alarak cevaplar üretir. Sistem mekânsal zekâ analizlerini asenkron olarak çalıştırmaktadır. Bu nedenle OGC WPS'in sunduğu senkron çalışma

modeline bu servis üzerinden destek verilmez. OGC WPS servisi aldığı analiz isteklerini sıraya koymak üzere sunucusuz mesaj kuyruğu servisine gönderir. Ayrıca servis aldığı isteğe ait bilgileri sunucusuz yapısal veri depolama alanına da görevlerin kaydedildiği tabloya görev takibi için kaydeder. Sunucusuz yapısal veri depolama alanındaki görevler tablosuna yapılan her kayıt için verilen görev numarası mekânsal veri analizi servisine iletilmek üzere mesaj kuyruğuna eklenen mesajların içine de eklenir. Sistemde üretilen analiz sonuçları bu görev numarası ile ilişkilendirilerek saklanır. Ayrıca bu görev numarası kullanıcı ile paylaşılarak kullanıcının analizi takip edebilmesi sağlanır.



Şekil 4.12 : Mekânsal zekâ sisteminin sunucusuz mimaride sistem tasarımı.

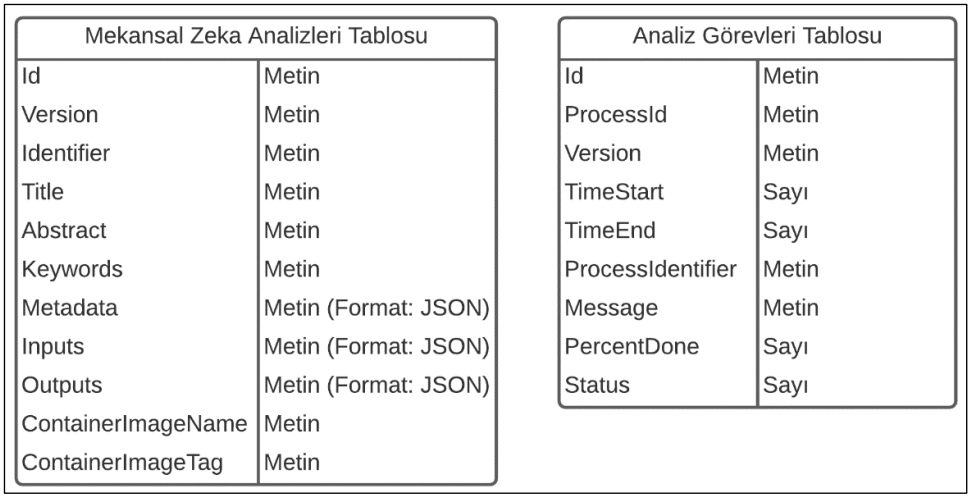
Sunucusuz mesaj kuyruğu servisi OGC WPS servisinden aldığı analiz görevlerini sırası önemli olmayacak bir şekilde toplayarak sunucusuz mekânsal analizi servisi tarafından çalıştırılmak üzere saklar. Bu mesaj kuyruğu gelen isteklerin mekânsal analiz servisinde meydana gelebilecek bir hata karşısında kaybolmasını da önler. Eğer bir görev çalıştırılırken hata oluşursa yeniden denenmek üzere bu mesaj kuyruğuna geri gönderilir. Mesaj kuyruğunun bir diğer önemli rolü ise mekânsal analiz servisinin

kuyrukta bekleyen görevlerin sayısına göre otomatik olarak ölçeklenebilmesini sağlamaktır. Mesaj kuyruğunun uzunluğu otomatik ölçeklendirme için ölçek katsayısı olarak kullanılabilir.

Sunucusuz mekânsal veri analizi servisinin görevi gelen analiz isteklerini mesaj kuyruğundan alıp çalıştırarak sonuç üretmektedir. Sunucusuz mekânsal veri analizi servisi birden fazla istek aynı anda geldiğinde analizleri paralel olarak çalıştırır. Grafik işlemcisinden faydalanmak ve uzun çalışma sürelerini yönetebilmek için servisin HoK modelindeki sunucusuz hesaplama hizmetleri ile çalışması gerekmektedir. Analizleri yapabilmek için önceden eğitilmiş modeller sunucusuz yapısal olmayan veri depolama alanından çekilerek kullanılır. Mekânsal veri analizi servisi işleme aldığı analizlerle ilgili durum bildirimlerini sunucusuz yapısal veri tabanındaki görev tablosunda kaydeder. Böylece CBS kullanıcısı OGC WPS servisine çalıştırdığı analiz hakkında durum bilgisi sorduğunda cevap alabilir. Mekânsal veri analizi servisi, analiz görevini başarı ile tamamladığında üretilen sonucu yapısal olmayan veri depolama servisi üzerinde kaydeder. Bununla birlikte analizin tamamlandığı bilgisi ve üretilen verinin sunucusuz depolama alanındaki kayıt konumu sunucusuz yapısal veri tabanındaki görev tablosunda ilgili kayda yazılır. Bu bilgi ile OGC WPS servisi, analiz sonucuna ulaşarak CBS kullanıcısına servis edebilir.

Sunucusuz yapısal veri tabanı sistemdeki mekânsal zekâ analizlerini ve CBS kullanıcısını tarafından çalıştırılan analiz görevlerine ait bilgileri saklamak için kullanılmıştır (Şekil 4.13).

Mekânsal zekâ analizleri tablosu sistem üzerindeki çalıştırılabilecek analizlerin tanımlandığı tablodur. Analiz görevleri tablosu ise çalışan analiz görevlerinin takibi için kullanılmaktadır. Mekânsal zekâ analizleri tablosu OGC WPS standardını gözetecek şekilde tasarlanmıştır. Bu veritabanına OGC WPS servisi, mekânsal analiz servisi ve veri bilimci ulaşabilir. Veri bilimci veri tabanına mekânsal zekâ analizlerini eklemek, çıkarmak ve güncellemek üzere bağlanır.



Şekil 4.13 : Mekânsal zekâ sistemi yapısal veri tabanı tabloları (Pakdil ve Çelik, 2021a).

Web CBS uygulaması bu mimari tasarımda OGC WPS istemcisi olarak konumlandırılmıştır. Ancak herhangi bir istemci uygulamasından HTTPS protokolü üzerinden gelen OGC WPS standardındaki istekler de yetkilendirme kontrolünde geçerek servise iletilebilir. Eğer gelen isteği yapan kullanıcının talep edilen mekânsal zekâ analizine erişim yetkisi yoksa istek reddedilir. Bu amaçla hangi mekânsal zekâ analizine hangi kullanıcı veya kullanıcı grupları için erişim yetkisi verileceği bilgisi başka bir yetkilendirme servisi veya bir veri tabanı üzerinde saklanabilir.

OGC WPS servisi ve sunucusuz mekânsal analiz servisinin geliştirilmesi ve bakımı uygulama geliştiricinin sorumluluğundadır. Bunun yanında sistemdeki bileşenlerin ayarlarının doğru şekilde yapılandırılması da uygulama geliştiricinin sorumluluğundadır. Bu amaçla uygulama geliştirici sistemde çeşitli testler yürüterek bu yapılandırmaların doğruluğunu test edebilir.

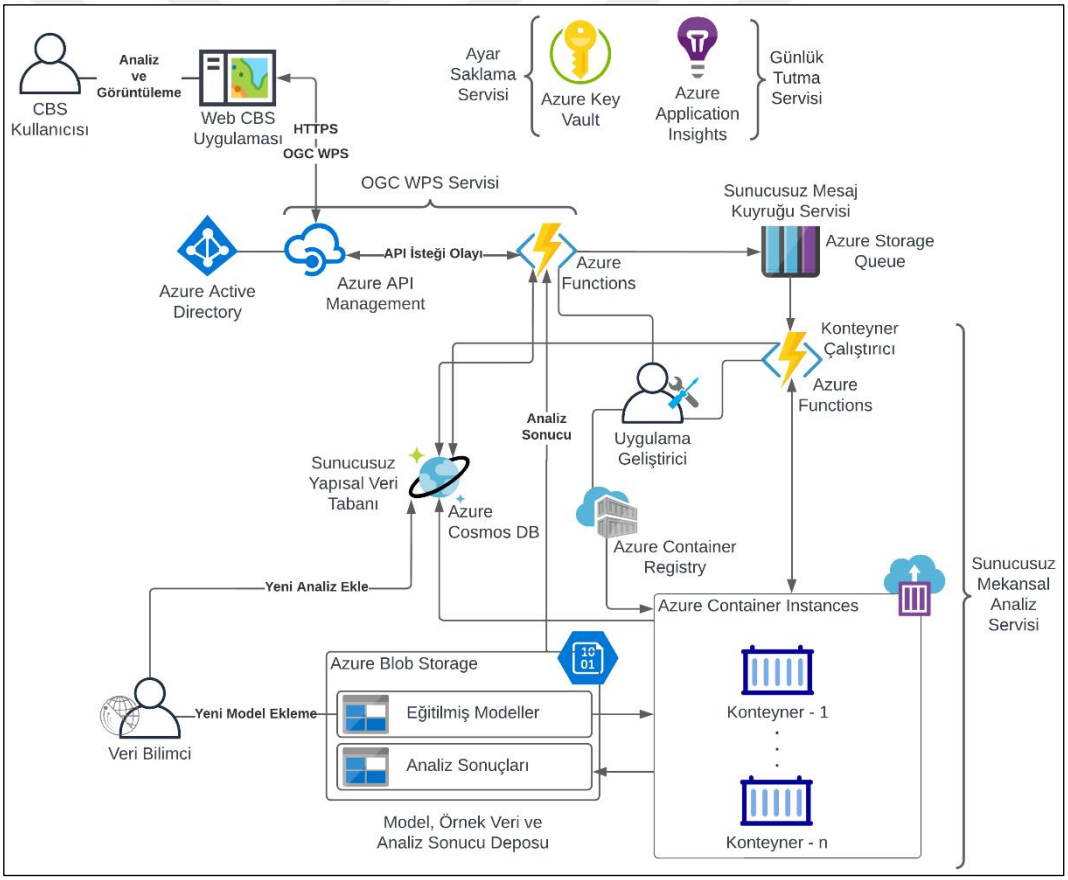
Günlük tutma servisi sistem tasarımındaki bileşenlerde meydana gelen hata ve diğer günlük kayıtlarını tutar. Bu hata kayıtları uygulama geliştiricinin uygulamalardaki hataların kök nedenlerini bularak çözmesine yardımcı olmaktadır. Bu servis ayrıca sistem bileşenlerine ait metrikleri de tutarak performans analizlerinin yapılmasını sağlamaktadır. Uygulama geliştirici bu bilgileri değerlendirerek uygulamaların iyileştirilmesi için kararlar alabilir. HoK modelinde çalışan mekânsal zekâ uygulamalarının günlük tutma servisi ile bağlantısı uygulama geliştirici tarafından kurulmalıdır.

Ayar saklama servisi, OGC WPS servisi ve mekânsal veri analizi servisinin diğer sistem bileşenleri ile olan iletişimi için gerekli bağlantı bilgilerini tutar. Sürüm

yayınlama otomasyonu ayar saklama servisi ile entegre edilerek sistem bileşenleri arasındaki bağlantı parametreleri otomatik olarak güncellenebilir.

4.4.2 Uygulama

Mekânsal zekâ sistemi tasarımı Şekil 4.14'da gösterildiği şekilde Microsoft Azure bulut bilişim sağlayıcısına ait sunucusuz hizmetlerle uygulanmıştır. OGC WPS servisi için açık kaynak kodlu PyWPS yazılımı kullanılmıştır (De Sousa ve diğ, 2019; Pakdil ve Çelik, 2021a). Bu yazılım Python programlama dilinde geliştirilmiştir. PyWPS yazılımın sunucusuz mimaride çalıştırmak için Azure Functions sunucusuz hesaplama hizmeti Linux işletim sistemi tabanlı olarak kullanılmıştır. Bu konfigürasyonda Python uygulamaları Azure Functions tarafından çalıştırılabilmektedir. PyWPS yazılımı HoF modeli üzerinde olay güdümlü olarak çalıştırılabilmektedir (Pakdil ve Çelik, 2021a).



Şekil 4.14 : Mekânsal zekâ sisteminin Microsoft Azure bulut bilişim sağlayıcısı üzerindeki sunucusuz hizmetlerle uygulaması.

Web CBS ve masaüstü CBS istemcilerinden gelen OGC WPS standardındaki istekler sunucusuz API yönetimi servisi üzerinden geçerek OGC WPS servisi uygulamasına ulaşır. Microsoft Azure bulut bilişim sağlayıcısında sunucusuz API yönetim servisinin

adı Azure API Management olarak geçmektedir. Bu servis gelen HTTPS isteklerini Azure Functions olaylarına dönüştürerek fonksiyonunun olay güdümlü olarak çalışmasını sağlar. Ayrıca bu servis Azure Active Directory sunucusuz kimlik yönetim servisiyle birlikte çalışarak kullanıcı yetkilendirilmesinde de görev yapmaktadır.

Microsoft Azure bulut bilişim sağlayıcısındaki sunucusuz kullanıcı ve rol yönetimi servisi Azure Active Directory olarak adlandırılmaktadır. Kullanıcıların oturum açma, yetkilendirme ve rol yönetimi işlemleri Azure Active Directory üzerinden yönetilebilmektedir.

Sunucusuz mesaj kuyruğu servisi olarak Azure Storage Queue hizmeti kullanılmıştır. Bu servis Azure Functions servisi ile çalışabilmektedir ve yeni bir mesaj geldiğinde Azure Functions hizmetini olay güdümlü olarak tetikleyebilir. OGC WPS servisi gelen analiz isteklerini görev tanımına dönüştürerek mesaj kuyruğuna kaydeder. Kaydedilen her yeni mesaj Azure Functions üzerinde çalışan konteyner çalıştırıcı uygulamayı tetikler.

Konteyner çalıştırıcı uygulama mesaj kuyruğuna gelen mesajlarla olay güdümlü olarak çalışmaktadır. Bu uygulamanın görevi HoK modelinde hizmet veren Azure Container Instances servisi üzerinde mekânsal analizi görevini yeni bir konteyner başlatarak çalışmasını sağlamaktadır. Köprü görevi gören bu uygulama, normalde olay güdümlü olarak çalışamayan Azure Container Instances servisini dolaylı yoldan hem olay güdümlü hem de otomatik ölçeklenebilir şekilde çalıştırır. Sunucusuz mesaj kuyruğuna gelen her yeni mesaj konteyner çalıştırıcı uygulamayı tetiklemektedir. Bu sayede platform tarafından izin verilen eş zamanlı çalışabilen konteyner sayısı kadar mekânsal zekâ analizi görevi paralel olarak çalışabilir. Bu uygulamanın diğer görevi de çalışması sonlanan konteyneri kapatmaktır. Bunun için konteyner uygulaması tarafından işlem sonunda konteyner çalıştırıcı fonksiyondan kendisini kapatması için istek gönderilir. Bu istek konteyner çalıştırıcı fonksiyonun dahili HTTPS adresi üzerinden gönderilir. Bu nedenle bu fonksiyonun yalnız mesaj kuyruğu mesajlarına değil ayrıca HTTPS isteklerine olay güdümlü olarak çalışması gerekmektedir. Konteyner çalıştırıcı uygulama aynı zamanda her yeni gelen analiz isteğinde aktif çalışan analizlerin çalışma sürelerini kontrol eder. Eğer beklenenden daha uzun süre çalışır durumda kalmış bir analiz görevi varsa buna bağlı olan konteyneri de kapatabilir. Bunun için uygulama geliştirici veri bilimciyle birlikte zaman aşımı süresine birlikte karar verilir.

Mekânsal zekâ analizlerinin çalıştırılması için HoK modelinde hizmet veren Azure Container Insantances servisi seçilmiştir. Bu servis gelen her bir analiz görevini yeni bir konteyner üzerinde çalıştırmaktadır. Konteynerlerin imajları uygulama geliştirici tarafından veri bilimcinin yardımı ile hazırlanarak sunucusuz konteyner imaj deposu olan Azure Container Registry üzerinde depolanır. Hangi konteyner imajının hangi mekânsal zekâ analizini çalıştıracağı bilgisi sunucusuz yapısal veri tabanındaki mekânsal veri analizleri tablosunda belirtilir. Konteyner çalıştırıcı gelen analiz görevinde hangi imajın çalıştırılacağı bilgisini bu tablodan okur. Azure Container Instances hizmeti grafik işlemcisi desteği de sağladığından grafik işlemcisini yoğun kullanan yapay zekâ uygulamalarının daha hızlı çalışmasını sağlayabilmektedir.

Veri bilimci tarafından eğitilmiş modeller sunucusuz yapısal olamayan veri depolama servisi Azure Blob Storage üzerinde saklanmaktadır. Eğitilmiş veri setleri çalışmaya başlayan konteynerler tarafından çekilerek kullanılabilmektedir. Ayrıca bu depolama servisinde mekânsal zekâ analizleri sonucu üretilen ikili formattaki veriler de saklanmaktadır. CBS kullanıcısı analiz sonucunu indirmek istediğinde OGC WPS servisi bu depolama alanından analiz sonucunu çekerek istemciye gönderir.

Sunucusuz yapısal veri tabanı hizmeti olarak Azure Cosmos DB servisi kullanılmıştır. Bu serviste veriler belge tipinde kaydedilmiştir. Bu nedenle SQL API hizmeti tercih edilmiştir. Mekânsal veri analizleri ve analiz görevleri tablosu için iki farklı belge koleksiyonu oluşturulmuştur. Veri bilimci bu servise uygun istemci bir uygulama ile ulaşarak yeni mekânsal zekâ analizlerini ekleyebilir.

Günlük tutma servisi olarak Azure Application Insight servisi kullanılmıştır. Bu servis merkezi bir şekilde günlük kayıtlarını ve performans metriklerini tüm sistem bileşenlerinden toplayabilmektedir. Sistem bileşenlerinin ve uygulamaların günlük ve metrik toplama ayarları yapılarak günlük tutma servisinin çalışması sağlanmaktadır. Uygulama geliştirici bu servis üzerinden sistemi takip ederek hata düzeltme ve iyileştirme için gerekli aksiyonları alabilmektedir.

Ayar saklama servisi olarak Azure Key Vault servisi kullanılmıştır. Bu servis sistemdeki uygulamalar tarafından erişilerek sunucusuz yapısal veri depolama ve sunucusuz yapısal olmayan veri depolama alanlarına erişim için gerekli bağlantı parametrelerine ulaşmaktadırlar. Azure Key Vault için birçok programlama dilinde kütüphane bulunmaktadır.

4.4.3 Değerlendirme

Sunulan sistem tasarımı ve uygulama Çizelge 4.6'da gösterildiği gibi 12 Faktör yöntemine göre değerlendirilmiştir. Geliştirilen sistemde HoF ve HoK modelindeki sunucusuz hesaplama servisleri kullanılmıştır. Bu nedenle 12 Faktör yöntemindeki "Yönetim Süreçleri" hariç diğer prensipler uygulanabilmektedir ve bu prensipler üzerinden değerlendirilmiştir.

Çizelge 4.6 : Mekânsal zekâ sisteminin 12 Faktör yöntemine göre değerlendirilmesi.

Prensip Adı	Değerlendirme
Kod Deposu	Uygulama geliştirici ve veri bilimci geliştirdiği uygulamaları ve model algoritmalarını kod deposu üzerinde arşivlemektedir.
Bağımlılıklar	Bağımlılıklar HoK modeline çalışan mekânsal zekâ analizi servisi için değerlendirilmiştir. Sistemde bu servis üzerinde çalışacak konteyner imajları oluşturulurken çalışacak kodun ihtiyaç duyacağı kütüphanelerin yüklenmesi imaj dosyası tanımlanırken belirtilir. Örneğin uygulamada GeoPandas, TensorFlow gibi konteynerlerde doğrudan gelmeyebilen kütüphaneler yüklenmelidir.
Ayarlar	Bu sistemde uygulamaların diğer destek servislerine bağlanabilmesi için gereken parametrelerin saklanması için ayar servisi kullanılmıştır. Uygulamada ise Azure Key Vault servisi ile ayarlar saklanmıştır.
Destek Servisleri	Mekânsal zekâ analizi servisi ve OGC WPS servisi kapsamında geliştirilen uygulamalar destek servislerine ağ üzerinden bağlanmışlardır.
Derle, Yayınla ve Çalıştır	Uygulama geliştirici sürüm yayınlama süreçlerinin otomasyonundan sorumludur. Bununla birlikte bu otomasyonu veri bilimcide kullanarak geliştirdiği modelleri test ederek sisteme yüklemektedir.
Süreçler	Sistem tasarımı uygulanırken HoK modelinde çalışacak konteynerlerin durumsuz çalışması gerekmektedir. Bir analiz işlemi bir sonraki analiz işlemine durum aktarmamaktadır. Bu amaçla Bölüm 4.6'da geliştirilen mekânsal analiz iş akışı sistemi tasarımı kullanılabilir.
Port Bağlama	Sistem tasarımında HoK modeli üzerinde çalışan konteynerlere dışardan erişim olmadığı için port bağlama prensibi uygulanmamıştır.

Çizelge 4.6 (devam) : Mekânsal zekâ sisteminin 12 Faktör yöntemine göre değerlendirilmesi.

Prensip Adı	Değerlendirme
Eş Zamanlılık	Sistem tasarımındaki HoF modelinde çalışan bileşenlerin eş zamanlı çalışması platform tarafından yönetilmektedir. Sistemdeki HoK modelinde çalışan mekânsal zekâ servisi ise eş zamanlı olarak sunucusuz mesaj kuyruğu servisi yardımıyla eş zamanlı olarak çalıştırılabilir. Uygulamada mesaj kuyruğunu dinlemek için aracı başka bir fonksiyon kullanılarak mekânsal zekâ analizlerinin eş zamanlı çalışması sağlanmıştır.
Kullanılabilirlik	Uygulamada mekânsal zekâ analizleri uzun sürülebileceğinden asenkron çalışma modeli seçilmiştir. Bu sayede kullanıcıdan gelen bir istek mesaj kuyruğu üzerinde sıraya konduktan sonra görev numarası ile isteğe hemen cevap verilir. Ayrıca mekânsal zekâ analizlerini yapmak üzere geliştirilen konteyner imajları da çalışma zamanında işi tamamladıktan sonra kapatılmaları için konteyner çalıştırıcı fonksiyona istek yapmaktadırlar.
Geliştirme, Test ve Yayın Ortamlarının Benzerliği	Uygulama geliştirici sistem tasarımının uygulamasını test ve yayın ortamlarında birbirlerine benzeyecek şekilde yayınlar. Geliştirmeler uygulama geliştiricinin bilgisayarında yapılmaktadır. Benzer şekilde veri bilimci de model geliştirmelerini kendi bilgisayarında veya başka bir bulut bilişim servisi yardımıyla yapabilir. Yaptığı model çalışmalarını yine test ve yayın ortamlarına yükler.
Günlük Tutma	Sistem bileşenleri tarafından üretilen günlük kayıtları için günlük tutma servisi bileşeni eklenmiştir. Uygulamada ise Azure Application Insight günlük tutma servisi kullanılmıştır.

Sistem tasarımı ayrıca mimari değerlendirme ölçütlerine göre Çizelge 4.7'de değerlendirilmiştir.

Çizelge 4.7 : Mekânsal zekâ sisteminin bulut bilişim sağlayıcısının mimari değerlendirme ölçütlerine göre değerlendirilmesi.

Ölçüt Adı	Değerlendirme
Performans Verimliliği	Sistemdeki bileşenlerin performans verimlilikleri uygulama geliştiricinin sorumluluğundadır. Sistem bileşenleri için seçilen algoritma ve teknolojiler performans için belirleyici olmaktadır. Sistem tasarımında mekânsal zekâ analizlerinin hızlı şekilde çalışabilmesi için HoK modeli tercih edilmiştir. Ayrıca birden fazla isteği eş zamanlı çalıştırarak hızlı cevap verebilmek için de mesaj kuyruğu bileşeni kullanılmıştır.
Güvenilirlik	Sistem bileşenleri günlük tutma servisi ile izlenerek hatalar takip edilmektedir. Hata durumunda sistemin çalışmaya devam edebilmesi için HoF modelindeki uygulamalar istekleri yeniden çalıştırmaktadır. HoK modelinde çalışan uygulamalarda bir hata oluşması durumunda uygulamaların günlük tutma servisine hata kayıtlarını göndermesinin yapılandırılması uygulama geliştiricinin sorumluluğundadır.
Güvenlik	Sistemde kullanıcı ve bileşenlerin yetkileri detaylı bir şekilde tanımlanmıştır. CBS kullanıcısı bir istemci uygulama ile yetkilendirme servisinden geçerek analizleri çalıştırabilir. Veri bilimci ise platform üzerinde bulunan dahili yetkilendirme sistemi ile yapısal olmayan veri depolama alanı ve yapısal veri depolama alanına ulaşarak mekânsal zekâ analizlerini yönetebilir.

4.5 Sunucusuz Olay Güdümlü Mekânsal Veri İşleme Servisi Tasarımı, Uygulaması ve Değerlendirmesi: Kandilli Deprem Habercisi Örneği

Sunucusuz mimaride çalışan HoF modelindeki hesaplama hizmetleri olay güdümlü olarak çalışmaktadırlar. Genellikle diğer platforma servisleri veya HTTP/HTTPS üzerinden gelen isteklerle olayların kaynağı olmaktadır. Bu sistem tasarımında ise ani gelişen bir doğal afet olan depremlerin sunucusuz mimarideki sistemde olay kaynağı olarak kullanımı incelenmiştir.

Türkiye'de deprem olaylarını Afet ve Acil Durum Yönetimi Başkanlığı'na (AFAD) ve Kandilli Rasathanesi ve Deprem Araştırma Enstitüsü ülkede çeşitli yerlere kurdukları sismograf cihazları ile takip etmektedirler. Yaptıkları bu izleme çalışmalarının çıktılarını anlık olarak internet sitelerinden konum bilgisiyle birlikte yayınlamaktadırlar. Ancak depremlerin aniden meydana gelmesiyle birlikte depremi hisseden kişiler deprem büyüklüğü ve konumu hakkında bilgi almak için kurumların internet sitelerine hep birlikte girerek yüksek miktarda trafik oluşturmaktadırlar. Bu

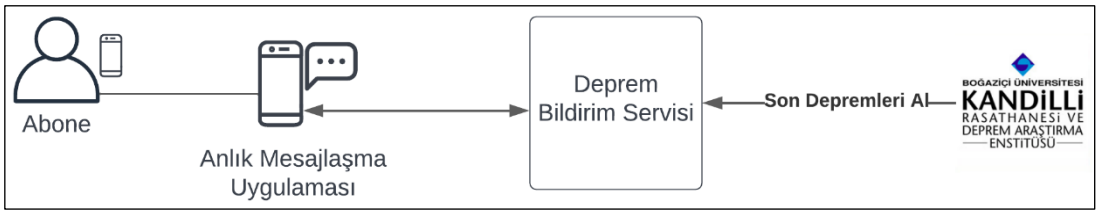
nedenle kurumların internet siteleri cevap veremez duruma gelmekte ve bu durum basında yer almaktadır (Top ve diğ, 2011; “4.2'lik deprem yetti: Kandilli'nin sitesi çöktü”, 2015).

Bu tez çalışmasında sunucusuz mimari kullanılarak meydana gelen depremleri kullanıcıların cep telefonlarına bildiren bir sistem tasarımı sunulmuştur. Bu sistem tasarımı ile deprem anında kurumların internet sitelerinin üzerindeki trafik yükü azaltılarak operasyonel kalmaları sağlanabilir. Bu amaçla sistem deprem bilgisi yayınlayan Kandilli Rasathanesi ve Deprem Araştırma Enstitüsü internet sitesini sık ve düzenli aralıklarla kontrol ederek yeni depremleri kullanıcılara iletmektedir. Bu tasarımın uygulaması açık kaynak kodlu olarak 2019 yılında yayınlanmıştır.

4.5.1 Tasarım

Senaryo

Sistem mimarisi tasarımında kullanılan senaryoda mobil kullanıcılar deprem bildirim servisine abone olarak Kandilli Rasathanesi tarafından yayınlanan son depremlerden anlık olarak haberdar olurlar. Bu senaryoda kullanıcılar bildirimleri alabilmek için sisteme özel ek bir mobil uygulama yüklemeden mobil uygulama mağazalarında mevcut olan popüler bir anlık mesajlaşma uygulamasını kullanarak bildirimleri alırlar.



Şekil 4.15 : Deprem bildirim sisteminin kullanım senaryosu.

Gereksinimler

Senaryoda belirtilen sistemin gereksinimleri belirlenmiştir ve bu gereksinimler sunucusuz mimari ile geliştirilen deprem bildirim servisi sistemi tasarlanırken dikkate alınmıştır.

1. Kullanıcılar sisteme abone olurken anlık mesajlaşma uygulamasını kullanmalıdırlar.
2. Anlık mesajlaşma uygulaması üzerinde sohbet botu özelliği bulunmalıdır. Kullanıcılar geliştirilecek bit sohbet botu ile deprem bildiri servisi ile etkileşim kurabilirler.

3. Anlık mesajlaşma uygulaması deprem bildirim servisi ile güvenli bir şekilde iletişim kurabilmelidir.
4. Deprem bildirim servisi de anlık mesajlaşma uygulamasının sunduğu web servisleri ile iletişim kurarak abone olan kullanıcılara deprem bildirimlerini gönderebilir.
5. Kullanıcılar isterlerse deprem büyüklüğüne veya konumuna göre koşul koyarak bu koşullara uyan depremlerin bildirimlerini alabilirler.
6. Kullanıcılara ait kişisel bilgiler sistem üzerinde tutulmamalıdır.
7. Sistemdeki uygulamaların sürümleri önce test sonra yayın ortamına alınmalıdır. Test ortamında hatası bulunan uygulamanın yeni sürümü yayınlanmamalıdır. Testler için anlık mesajlaşma uygulamasında test amaçlı sohbet botu ve aboneler oluşturulmalıdır.
8. Sistemdeki uygulamalar için geliştirilen kodlar kod deposunda saklanmalı ve sürüm yayınlama otomasyonu üzerinden yayınlanmalıdır.

Roller

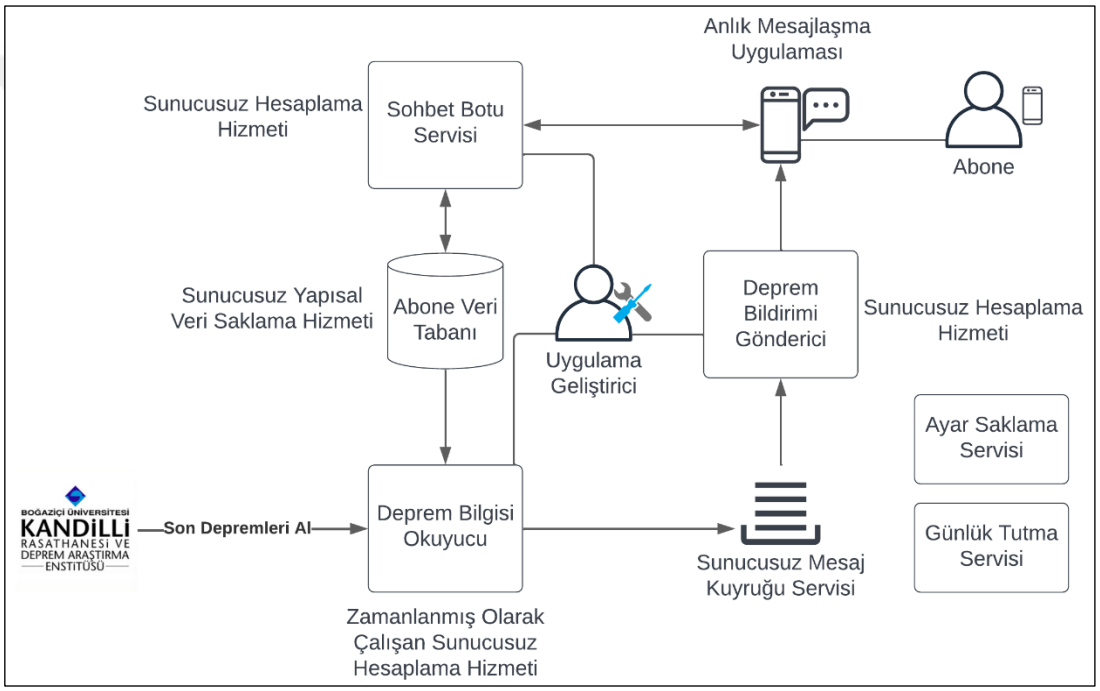
Geliştirilen sistem mimarisinde depremlerin takipçisi olan abone ve uygulama geliştirici olmak üzere iki kullanıcı rolü bulunmaktadır. Abone rolü sistem üzerinde cep telefonundaki anlık mesajlaşma uygulamasının kullanıcısıdır. Bu roldeki kullanıcı sistemle etkileşimini sohbet botu aracılığı ile yazılı mesajlaşarak veya konum bilgisi göndererek yapar. Kullanıcı sistemdeki aboneliğini istediği zaman sonlandırabilir. Kullanıcı istediği zaman sistemdeki deprem bildirim koşullarını güncelleyebilir, yeni koşul ekleyebilir veya koşul kaldırabilir. Hiçbir koşul koymayan aboneler kurum tarafından yayınlanan tüm deprem bilgilerini bildirim olarak alırlar.

Uygulama geliştiricisi rolü sistemdeki uygulamaların geliştirilmesi ve devamlılığını sağlamakla sorumludur. Bu roldeki kullanıcı sunucusuz mimariye uygun uygulama geliştirir ve yayına alır. Sunucusuz mimarilerde altyapı yönetimi platform sağlayıcısı tarafından yapılmakta olduğu için uygulama geliştiricinin ileri düzey bulut bilişim altyapı yönetimi bilmesi gerekmemektedir. Uygulama geliştirici platform üzerinde sağlanan sunucusuz hesaplama hizmetlerinin destek verdiği programlama dillerinden birini seçerek uygulamalarını geliştirebilir. Geliştirmeler sonucu üretilen kodlar kod deposu üzerinde arşivlenmelidir bu nedenle uygulama geliştiricin kod deposuna erişimi bulunmalıdır. Uygulama geliştirici arşivlenen kodların kod deposundan yayına kadar olan sürüm yayınlama süreçlerini kurgulayarak yayın otomasyonunu sağlar. Bu

kapsamda uygulama geliştirici sistem uygulamasını test ve yayın ortamı olmak üzere iki farklı şekilde yayınlar. Uygulama geliştirici sistemdeki performans metrikleri ve günlük kayıtlarının tutulduğu bileşenin de kullanıcısıdır. Bu bileşen üzerinden sistemin performansını ve sağlığını takip ederek uygulamalarda iyileştirmeler yapabilir.

Sistem tasarımı ve bileşenleri

Açıklanan senaryo, kullanıcı rolleri ve gereksinimlere göre Şekil 4.16'da gösterilen şekilde sistem tasarlanmıştır. Sistem tasarımı sunucusuz mimaride çalışacak şekilde geliştirilmiş ve bileşenleri açıklanmıştır.



Şekil 4.16 : Deprem bildirim sistemi tasarımı ve bileşenleri.

Sistem tasarımında deprem bildirimlerini alan abone ile anlık mesajlaşma uygulamasının sohbet botu etkileşim kurmaktadır. Anlık mesajlaşma uygulamasının sohbet botu uygulama geliştirici tarafından yapılandırılarak, kullanıcı etkileşimleri tanımlanır.

Sohbet botuna iletilen istekler anlık mesajlaşma uygulamasının arka plan servisleri üzerinden sistem mimarisindeki sohbet botu servisine gönderilir. Sohbet botu servisi gelen isteklere göre abone veri tabanıyla birlikte çalışarak abone bilgilerini günceller, yeni abone kaydeder, abone siler veya aboneye ait koşulları veri tabanı üzerinde günceller. Kullanıcının isteklerine göre yeni sorular sorulması gerektiğinde sohbet

botu servisi anlık mesajlaşma uygulamasının web servislerini çağırarak kullanıcı ile yazılı iletişim kurulmasını sağlar. Sohbet botu servisi anlık yükselen trafikleri kaldırabilmek için otomatik olarak ölçeklenebilir olmalıdır.

Abone veri tabanı sunucusuz yapısal veri depolama hizmeti üzerinde konumlandırılmaktadır. Bu servisin indeksleme ve nokta türündeki coğrafi konum bilgisi saklama ve sorgulama özellikleri sunar. Abone veri tabanı üzerinde aboneye ait sohbet kanalı bilgisi ve abonenin belirlediği deprem bilgisini alma koşulları tutulur. Abone hakkında kişisel veriler veri tabanında tutulmaz. Deprem büyüklüğü koşulu ve konum koşulu aboneler tablosunda tutulur. Deprem büyüklüğü koşulu sayı veri türünde olarak saklanır. Konum koşulu ise coğrafi nokta türünde saklanır. Veri tabanı olarak kullanılan sunucusuz yapısal veri depolama hizmeti eğer coğrafi veri tiplerini destekliyorsa konum alanı için nokta coğrafi verisi tutan veri tipi seçilir. Abone veri tabanı bildirim koşullarına ve abonenin sohbet kanalı bilgisine göre hız erişim için indekslenir. Böylece diğer bileşenlerin abone verisine erişimleri hızlı olması sağlanır.

Deprem bilgisi okuyucu bileşen belirlenen kısa aralıklarla Kandilli Rasathanesi'nin internet sitesinde yayınlanan son deprem bilgilerini okuyarak yeni deprem bilgisini kontrol eder. Eğer yeni deprem bilgisi eklenmişse bu yeni deprem bilgisine ait özniteliklerle abone veri tabanındaki koşullar karşılaştırarak hangi abonelere deprem bildirimi gideceği belirlenir. Yeni deprem bilgisini ve bu bilgiyi alacak abone bilgilerini sunucusuz mesaj kuyruğu servisine her abone için ayrı ayrı mesaj olarak gönderir.

Sunucusuz mesaj kuyruğu servisinde bildirim gönderilecek abone ve deprem bilgileri her bir bildirim başına bir mesaj olacak şekilde kaydedilir. Örneğin bir deprem bilgisi yüz aboneye gönderilecekse yüz ayrı mesaj kuyruğa yazılır. Bu mesaj kuyruğunun amacı deprem bildirim gönderici uygulamanın ölçeklenerek bildirimleri eş zamanlı olarak hızlı bir şekilde gönderilmesini sağlamaktır. Bu nedenle bildirim sayısı deprem bildirimi göndericinin otomatik ölçeklendirme katsayısı olarak kullanılmıştır.

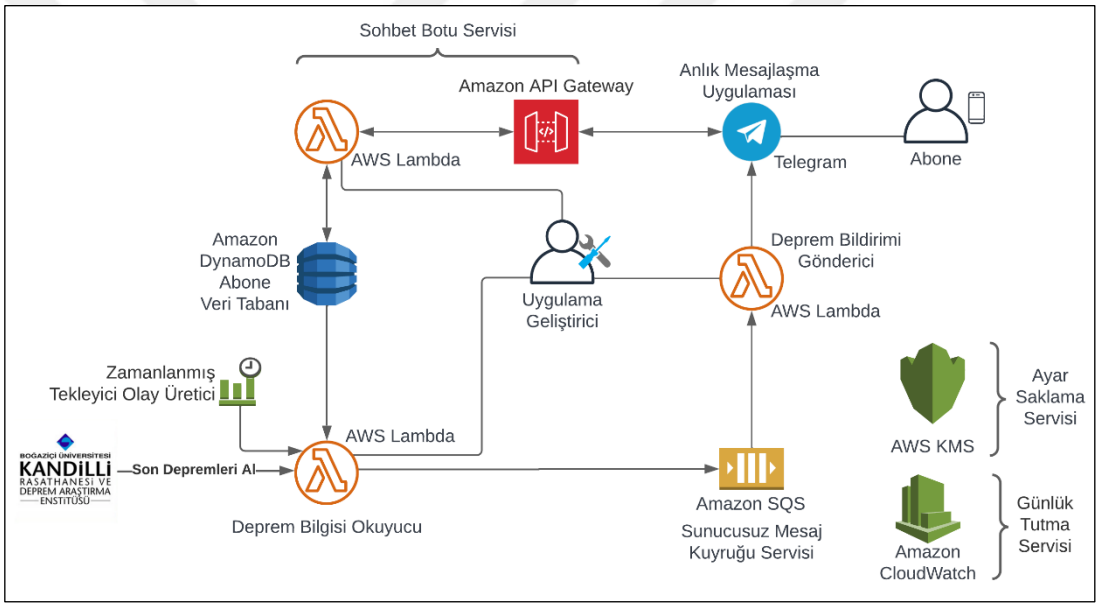
Günlük tutma servisi sistem tasarımındaki bileşenlerde meydana gelen hata ve diğer günlük kayıtlarını tutar. Uygulama geliştirici bu hata kayıtlarını okuyarak uygulamalardaki hataların kök nedenlerini bularak çözebilmektedir. Bu servis ayrıca sistem bileşenlerine ait metrikleri de tutarak performans analizlerinin yapılmasını

sağlamaktadır. Uygulama geliştirici bu performans bilgilerini değerlendirerek uygulamaların iyileştirilmesi için kararlar alabilmektedir.

Ayar saklama servisi, sohbet botu servisinin anlık mesajlaşma uygulaması ile olan iletişimi için gereken parametreleri ve abone veri tabanına bağlantı bilgilerini saklar. Deprem bilgisi okuyucu servisi için de benzer şekilde abone veri tabanına bağlantı parametrelerini, sunucusuz mesaj kuyruğu servisine bağlantı parametrelerini ve Kandilli Rasathanesi'ne ait internet sitesinin adresini saklar.

4.5.2 Uygulama

Deprem bildirim sistemi tasarımı Şekil 4.17'de gösterildiği şekilde AWS bulut bilişim sağlayıcısına ait sunucusuz hizmetlerle uygulanmıştır.



Şekil 4.17 : Deprem bildirim sistemi tasarımının AWS bulut bilişim sağlayıcısına ait sunucusuz hizmetlerle uygulaması (Pakdil ve Çelik, 2023, baskıda).

Anlık mesajlaşma uygulaması olarak Telegram uygulaması kullanılmıştır. Telegram uygulaması sunduğu API'larla sohbet botu geliştirilmesine destek vermektedir. Telegram üzerinde test ve yayın ortamları için iki farklı sohbeti botu oluşturulmuştur. Sohbet botu kullanıcıdan komutları "/" sembolü ile başlayan kelimelerle almaktadır. Çizelge 4.8'te verilen komutlar Türkçe karakter kullanmadan sohbet botuna tanımlanmıştır.

Çizelge 4.8 : Telegram deprem bildirim sohbet botunun komutları

Komut Adı	Açıklama
/basla	Deprem bildirimlerine aboneliği başlatır.
/siddet	Deprem bildirimlerine şiddet filtresi uygulanması için kullanılır.
/konumekle	Deprem bildirimlerine konum filtresinin nasıl uygulanacağını açıklar.
/konumsil	Deprem bildirimlerinde konum filtresini kaldırır.
/bitir	Deprem bildirimlerine aboneliği bitirir.

Sistem tasarımındaki tüm uygulamalar HoF modelindeki sunucusuz hesaplama servisi olan AWS Lambda üzerinde çalışacak şekilde geliştirilmiştir. Uygulamaların programlama dili olarak C# programlama dili ve .Net Core çatkısının 3.1 sürümü seçilmiştir.

Sohbet botu servisi iki bileşenden oluşmaktadır. İstemcilerden gelen istekleri alarak AWS Lambda servisine HTTP olayı olarak ileten Amazon API Gateway isimli sunucusuz hizmet modeli ile çalışan internet trafiği kontrol ve yönlendiricisi kullanılmıştır. Amazon API Gateway, Telegram sohbet botundan gönderilen komutları sohbet botu uygulamasının çalıştığı AWS Lambda servisine olay olarak göndermektedir. Sohbet botu servisi gelen isteklerdeki komutları çalıştırarak kullanıcılara ek sorular sorabilmektedir. Bunun için Telegram uygulaması ile çift yönlü iletişim kurmaktadır. Sohbet botu servisinin sağladığı internet servisleri açık olarak yayınlandığı için güvenlik nedeniyle abone verileri listeme ve sorgulama servisleri bu bileşende sunulmamalıdır.

Abone veri tabanı olarak Amazon DynamoDB sunucusuz yapısal veri depolama hizmeti kullanılmıştır. Bu veri tabanı anahtar-değer veri tabanı olarak çalışmaktadır. Aboneler tablosu şeması üç alandan oluşturulmuştur (Şekil 4.18). Burada “chatid” alanı Telegram üzerinde sohbet botu ile başlatılan sohbet oturumu için atanan benzersiz sayısal değerdir ve anahtar olarak kullanılmıştır. Bu değer ile Telegram üzerinden aynı kullanıcıyla yeniden iletişim kurulabilmektedir. Konum filtresine ait nokta bilgisi Amazon DynamoDB tarafından destek verilen GeoHash formatında “location” alanında saklanmıştır. Deprem şiddeti koşuluna ait şiddet bilgisi de “magnitude” alanında sayısal olarak saklanmıştır.

Aboneler Tablosu	
ChatId	Sayı
Location	Metin
Magnitude	Sayı

Şekil 4.18 : Aboneler tablosu şeması.

Deprem bilgisi okuyucu servis Kandilli Rasathanesi'nin internet sitesinden son deprem bilgilerini HTML (HyperText Markup Language) formatında okur. Bu uygulamayı çalıştıran AWS Lambda servisi Amazon EventBridge üzerinde oluşturulan zamanlayıcı tarafından her dakika çalışacak şekilde ayarlanmıştır. Amazon EventBridge, her dakika AWS Lambda servisindeki deprem bilgisi okuyucu uygulamayı çalıştıran tetikleyici olayları üretmektedir. Deprem bilgisi okuyucu servisi abone veri tabanına Amazon DynamoDB'ye bağlanarak yeni deprem bilgisini alması gereken aboneleri sorgular. Tespit edilen abonelerin her biri için bildirim gönderim parametreleri oluşturularak sunucusuz mesaj kuyruğu servisine JSON formatında mesaj olarak eklenir.

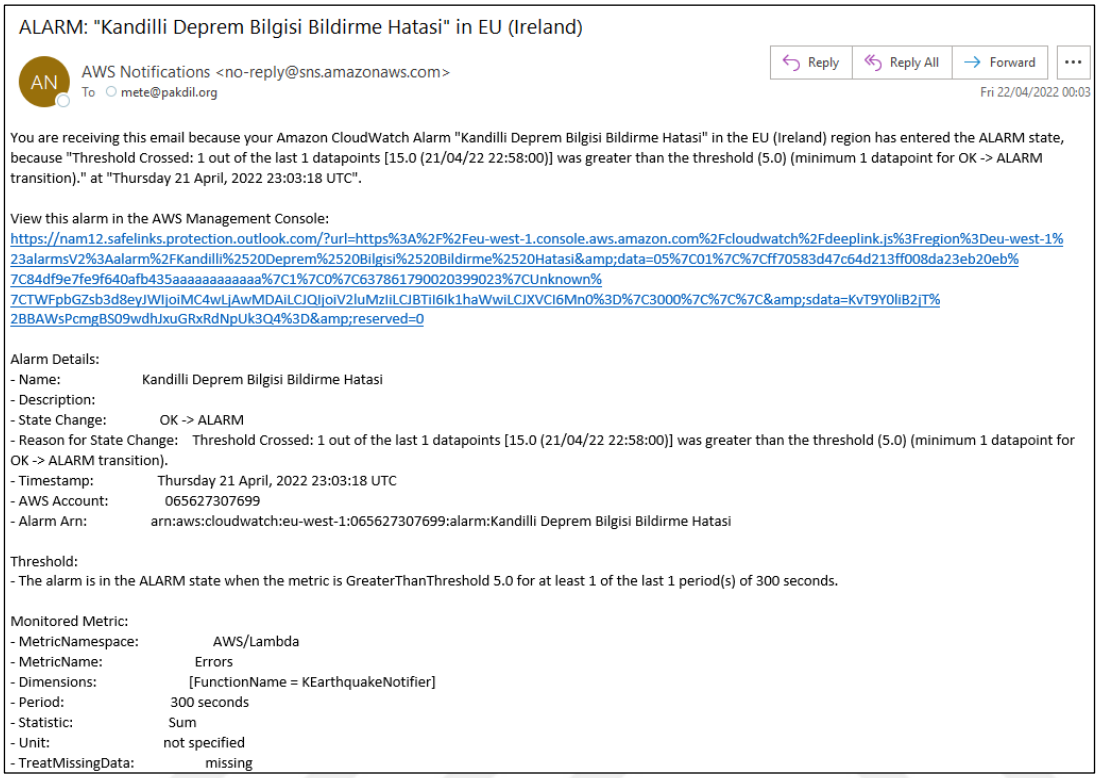
Sunucusuz mesaj kuyruğu servisi olarak Amazon SQS adlı servis kullanılmıştır. Amazon SQS servisi AWS Lambda servisi ile çalışabilmektedir. Bekleyen mesajlar işlenebilmek için deprem bildirimi gönderici servisini eş zamanlı çalışmak üzere tetiklemektedir.

Deprem bildirimi gönderici servisi AWS Lambda sunucusuz hesaplama servisi üzerinde çalışmaktadır. Sunucusuz mesaj kuyruğundan gelen her bir abonenin bildirim mesaj parametrelerini okuyarak Telegram mesajlaşma uygulamasına mesaj olarak iletir. Böylece abone olan kullanıcıların deprem hakkında en kısa sürede bilgilendirilmesi sağlanır.

Sistemdeki bileşenlerin birbirleri arasındaki iletişimi için yetkilendirmeler AWS IAM hizmeti üzerinden bileşenlere rol ataması yapılarak sağlanmıştır. Bunun yanında AWS KMS hizmeti ile Telegram uygulamasına bağlantı parametreleri uygulamalara güvenli bir şekilde servis edilmek için ayar saklama servisi olarak kullanılmıştır.

Amazon CloudWatch hizmeti ile tüm bileşenler takip edilerek ve hata günlükleri tutularak izlenmektedir. Sistemdeki uygulamalara ait metrikler sınır değerler belirlenerek alarmlar oluşturulmuştur. Sınır değerler aşıldığında uygulama geliştirici e-posta ile bilgilendirilerek sistemdeki sorunlardan anlık olarak haberdar olabilmektedir (Şekil 4.19).

Uygulamada kod deposu olarak GitHub kod paylaşım sitesi kullanılmıştır. Kodlar açık kaynak kodlu olarak paylaşılmıştır. Sürüm yayınlama otomasyonu için de Github Actions servisi kullanılmıştır (URL-6).


ALARM: "Kandilli Deprem Bilgisi Bildirme Hatasi" in EU (Ireland)

AN AWS Notifications <no-reply@sns.amazonaws.com>
To mete@pakdil.org

Reply | Reply All | Forward | ...
Fri 22/04/2022 00:03

You are receiving this email because your Amazon CloudWatch Alarm "Kandilli Deprem Bilgisi Bildirme Hatasi" in the EU (Ireland) region has entered the ALARM state, because "Threshold Crossed: 1 out of the last 1 datapoints [15.0 (21/04/22 22:58:00)] was greater than the threshold (5.0) (minimum 1 datapoint for OK -> ALARM transition)." at "Thursday 21 April, 2022 23:03:18 UTC".

View this alarm in the AWS Management Console:
https://nam12.safelinks.protection.outlook.com/?url=https%3A%2F%2Feu-west-1.console.aws.amazon.com%2Fcloudwatch%2Fdeeplink.js%3Fregion%3Deu-west-1%23alarmsV2%3Aalarm%2FKandilli%2520Deprem%2520Bilgisi%2520Bildirme%2520Hatasi&data=05%7C01%7C%7Cff70583d47c64d213ff008da23eb20eb%7C84df9e7fe9f640afb435aaaaaaaaaaaa%7C1%7C0%7C637861790020399023%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000%7C%7C%7C&sdata=KvT9Y0liB2jT%2BBAWsPcmgBS09wdhJxuGRxRdNpUk3Q4%3D&reserved=0

Alarm Details:
- Name: Kandilli Deprem Bilgisi Bildirme Hatasi
- Description:
- State Change: OK -> ALARM
- Reason for State Change: Threshold Crossed: 1 out of the last 1 datapoints [15.0 (21/04/22 22:58:00)] was greater than the threshold (5.0) (minimum 1 datapoint for OK -> ALARM transition).
- Timestamp: Thursday 21 April, 2022 23:03:18 UTC
- AWS Account: 065627307699
- Alarm Arn: arn:aws:cloudwatch:eu-west-1:065627307699:alarm:Kandilli Deprem Bilgisi Bildirme Hatasi

Threshold:
- The alarm is in the ALARM state when the metric is GreaterThanThreshold 5.0 for at least 1 of the last 1 period(s) of 300 seconds.

Monitored Metric:
- MetricNamespace: AWS/Lambda
- MetricName: Errors
- Dimensions: [FunctionName = KEarthquakeNotifier]
- Period: 300 seconds
- Statistic: Sum
- Unit: not specified
- TreatMissingData: missing

Şekil 4.19 : Amazon CloudWatch servisinden gelen deprem bildirimi gönderici bileşeni ile ilgili sorun olduğunu belirten alarm e-postası.

4.5.3 Değerlendirme

Uygulanan sistem tasarımı Çizelge 4.9'da gösterildiği gibi 12 Faktör yöntemine göre değerlendirilmiştir. Sunulan sistemde sadece HoF modelindeki sunucusuz hesaplama servisleri kullanılmıştır. Bu nedenle 12 Faktör yöntemindeki "Bağımlılıklar", "Süreçler", "Port Bağlama" ve "Yönetim Süreçleri" prensipleri doğrudan uygulanamadığı için değerlendirilmemiştir.

Çizelge 4.9 : Deprem bildirim sisteminin 12 Faktör yöntemine göre değerlendirilmesi.

Prensip Adı	Değerlendirme
Kod Deposu	Uygulama geliştirici geliştirdiği uygulamaları kod deposu üzerinde arşivlemekte ve açık kaynak kodlu olarak yayınlamaktadır.
Ayarlar	Sistem tasarımındaki uygulamaların gizli ayarları ayar saklama servisi üzerinde saklanmıştır. Gizli olmayan ayarlar ise uygulamaların çalıştığı konteynerlerin ortam değişkenleri üzerinde tutulmuştur. Sistem tasarımının uygulamasında ise AWS KMS servisi gizli ayarların güvenli bir şekilde saklanması için kullanılmıştır. Kandilli Rasathanesi'nin son depremler internet sitesi ise ortam değişkenlerinde saklanmıştır.
Destek Servisleri	Uygulamalar abone veri tabanına ağ üzerinden bağlanarak ulaşmıştırlar.
Derle, Yayınla ve Çalıştır	Sürüm yayınlama süreçlerinin otomasyonu uygulama geliştirici tarafından kurulmuştur. Sistem tasarımındaki uygulamaların bu otomasyonla yeni sürümleri yayınlanmaktadır.
Eş Zamanlılık	Uygulamalar HoF modelinde çalıştırılmak üzere geliştirildiği için eş zamanlı çalışma yönetimi bulut bilişim sağlayıcısı tarafından yapılmaktadır. Abonelere bildirim gönderiminin eş zamanlılığını sağlamak için bildirim mesajları sunucusuz mesaj kuyruğu üzerinden bildirim gönderici fonksiyona iletilmiştir.
Kullanılabilirlik	Sistem herhangi bir süreçte hata olması durumunda yeniden süreci tekrar edecek şekilde tasarlanmıştır. Deprem bilgisi okuyucu kurum sitesine ulaşamaması durumunda yeniden denemektedir. Deprem bildirimi gönderici hata durumunda gönderimi tekrarlamaktadır. Sistemdeki uygulamalar durumsuz çalışacak şekilde tasarlanmıştır. Kalıcı veriler abone veri tabanında tutulan verilerdir.
Geliştirme, Test ve Yayın Ortamlarının Benzerliği	Uygulama geliştirici sistem tasarımını test ve yayın ortamlarında birbirlerine benzeyecek şekilde yayınlar. Geliştirmeler uygulama geliştiricinin bilgisayarında yapılmaktadır.
Günlük Tutma	Sistem bileşenleri tarafından üretilen günlük kayıtları için günlük tutma servisi kullanılmıştır. Uygulamada ise Amazon CloudWatch sunucusuz günlük tutma servisi kullanılmıştır.

Sistem tasarımı ayrıca mimari değerlendirme ölçütlerine göre Çizelge 4.10'da değerlendirilmiştir.

Çizelge 4.10 : Vektör karo harita servisinin bulut bilişim sağlayıcısı mimari değerlendirme ölçütlerine göre değerlendirilmesi.

Ölçüt Adı	Değerlendirme
Performans Verimliliği	Sistemdeki uygulamalar yeni deprem bilgisi okunduktan sonra hızlı şekilde abonelere iletecek şekilde tasarlanmıştır. Bunu sağlamak için sistem tek amaca hizmet eden küçük bileşenlere bölünmüştür. Böylece HoF modelindeki sunucusuz hesaplama bileşenlerinin çalışmalarını hızlıca tamamlamaları beklenmektedir.
Güvenilirlik	Sistem bileşenleri günlük tutma servisi ile izlenerek hatalar takip edilmektedir. Bununla birlikte hatalara karşı süreçlerin kaybolmaması için ek önemler tasarlanmıştır.
Güvenlik	Sistemdeki servisler doğrudan kullanılamamaktadır. Kullanıcılar anlık mesajlaşma uygulaması üzerinden sistemle iletişim kurmaktadır. Sistemin dışarıyla iletişimi sohbet botuna ait arka plan servisleri üzerinden makine-makine iletişimi şeklindedir. Abone veri tabanı üzerinde kişisel veriler tutulmamaktadır.

Sistem AWS bulut bilişim sağlayıcısı üzerinde 29 Ekim 2019 tarihinden bu yana çalışmaktadır. Sistemde bu süre yapılan kod iyileştirmeleri dışında bulut bilişim altyapısının yönetilmesi gerekmemiştir. Güncel abone kullanıcı sayısı 8 Mayıs 2022 tarihi itibari ile 3025'dir. AWS bulut bilişim sağlayıcısı HoF modelindeki sunucusuz hesaplama hizmeti AWS Lambda için aylık bir milyonuncu çalışmasından sonra ücretlendirmeye başlamaktadır. Sistem bugüne kadar bu kullanım miktarına erişmediği için herhangi bir ekonomik maliyet oluşturmamıştır.

4.6 Sunucusuz Mekânsal Analiz İş Akışı Sistemi Tasarımı, Uygulaması ve Değerlendirmesi

Mekânsal bilgi teknolojileri büyük veri analizi projelerinin giderek önemli bir parçası hatta ana konusu olmaya başlamıştır. Sunucusuz mimarilerin bu kapsamda değerlendirilmesi ise teknik zorlukları nedeniyle henüz yeterince ilgi görememektedir (Baldini ve diğ, 2017). Her ne kadar sunucusuz mimarilerin mekânsal veri analizlerinde kullanılması literatürde çok fazla yer almasa da bu konu mekânsal olmayan veri analizi çalışmaları içerisinde giderek artan bir ilgi görmektedir.

Günümüzde halen klasik CBS araçlarıyla mekânsal veri analizi iş akışları tasarlanmakta ve bu iş akışları genelde kullanıcının kişisel bilgisayarı üzerinde çalıştırılmaktadır. Coğrafi veri kaynaklarının çoğalması ve üretilen veri miktarının büyümesi ile bu klasik yaklaşımlarla veri analizinin yapılması verinin büyüklüğü nedeniyle zorlaşmaktadır. Klasik yaklaşımlara alışmış bir kullanıcının doğrudan bulut bilişim teknolojileri üzerinde çalışan dağıtık karmaşık veri analizi sistemlerine geçişi de ileri düzey bulut bilişim eğitimi gerektirmekte ve zaman almaktadır. Bu geçişin kolaylaştırılması için bulut bilişim altyapısının yönetimini soyutlayan sunucusuz mimariler tercih edilebilir. Böylece kullanıcı daha az altyapı yönetimi ile uğraşırken daha çok veri analizini yapacak uygulamaya veya algoritmaya yoğunlaşabilir.

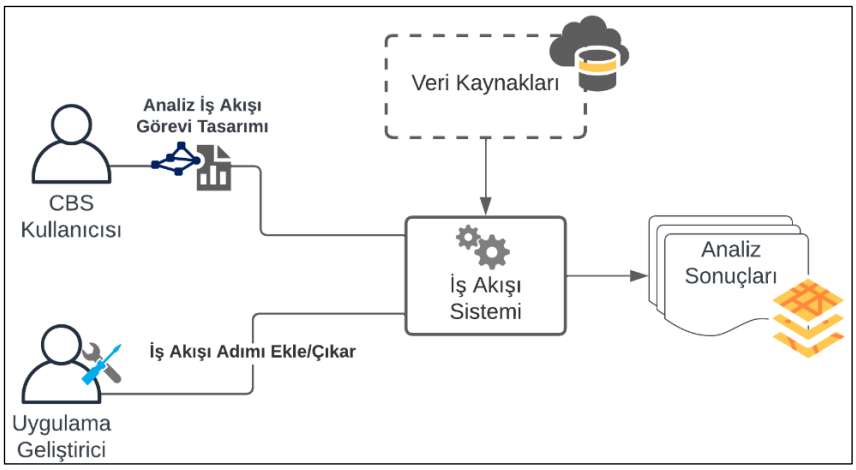
Bu tez çalışmasında sunucusuz mimariler üzerinde çalışan mekânsal veri analizi iş akışı sisteminin tasarımı ve uygulaması sunulmuştur (Pakdil ve Çelik, 2021b). Bu tasarım ile ileri düzey bulut bilişim altyapı yönetimi bilgisi olmayan kullanıcılar veri analizi görevlerini bulut bilişim platformlarında sunulan yüksek işlem gücü hacmi ve düşük çalıştırma maliyetlerinden faydalanarak çalıştırabilirler. Tasarımda sunulan mimari ile iş akışındaki adımlar sıralı veya paralel şekilde bir akış içerisinde birbirlerine durum aktararak çalışabilmektedir.

Sistem tasarımında kullanıcıların iş akışlarını tasarlayabilecekleri bir iş akışı modeli ve tanımı da sunulmuştur. Bu iş akışı tanımı hem makinelerin hem de insanların kolayca okuyup yazabileceği şekilde tasarlanmıştır. Böylece kullanıcılar karmaşık iş akışlarını kolayca yazarak çalıştırabilir, diğer kullanıcılarla paylaşabilir veya kod depolarında arşivleyebilir.

4.6.1 Tasarım

Senaryo

Sistem mimarisi tasarımında kullanılan senaryoda CBS kullanıcıları tasarladıkları mekânsal veri analizi iş akışlarını sisteme yükleyerek çalıştırırlar. İş akışı sistemi gelen iş akışlarını çeşitli kapalı veya açık coğrafi veya coğrafi olmayan veri kaynaklarından okuyarak adım adım çalıştırır. Üretilen analiz sonuçları da CBS kullanıcısının indirebileceği şekilde saklanır (Şekil 4.20).



Şekil 4.20 : Mekânsal analiz iş akışı sisteminin kullanım senaryosu.

Sistemdeki uygulama geliştiriciler de iş akışlarında kullanılabilecek iş akışı görevlerini sisteme ekleyerek CBS kullanıcılarının mekânsal analiz iş akışı tasarımlarında birer adım olarak kullanabilmeleri için yayınlarlar.

Gereksinimler

Mekânsal analiz iş akışı sistem mimarisinin tasarımında dikkate alınan gereksinimler aşağıdaki şekilde belirlenmiştir.

1. CBS kullanıcıları iş akışı tasarımlarını bir iş akışı tanımına göre yazabilmelidirler. Oluşturulan iş akışı tasarımları makine ve insan tarafından kolay okunabilir, arşivlenebilir ve paylaşılabilir bir formatta olmalıdır.
2. İş akışlarını çalıştırılmadan önce doğrulamalıdır. Doğrulama kuralları belgelenerek kullanıcı ile paylaşılmalıdır.
3. İş akışı doğrulama hataları kullanıcıya detaylı bir şekilde iş akışını çalıştırmadan önce bildirilmelidir.
4. Doğrulamadan geçen iş akışları kullanıcılar tarafından çalıştırılabilmek üzere saklanır.
5. İş akış tasarımları içerisinde paralel ve tekrarlı işlemler çalıştırılabilmelidir.
6. Tekrarlı işlemler eş zamanlı olarak çalıştırılabilmelidir. Örneğin, on kez tekrar edecek bir işlem eş zamanlı olarak paralel iki koldan çalışarak beş döngüde tamamlanabilmelidir.
7. İş akışlarındaki adımlar tasarımdaki sıraya göre çalıştırılmalıdır.
8. Bir iş akışı adımını çıktısı sonraki bir iş akışı adımına girdi olarak verilebilmelidir.
9. İş akışlarının girdi ve çıktıları olabilir.

10. Bir iş akışı girdisi işi akışındaki adımlara girdi olarak verilebilmelidir.
11. Bir iş akışı adımının çıktısı iş akışı adımlarının çıktılarından biri olmalıdır.
12. İş akışı analizinin çalışma sonuçları kalıcı bir depolama alanında saklanmalıdır.
13. İş akışı adımları çeşitli açık veya dahili veri kaynaklarını kullanabilmelidir. Bu nedenle sistemin internet erişimi olmalıdır.
14. İş akışı tasarımlarında kullanılabilecek yeni iş akışı görevleri sisteme eklenebilir olmalıdır.
15. İstemci uygulamalar OGC API Processes tanımına uygun istek gönderip cevap alabilmelidir.
16. Uygulama geliştiriciler yeni iş akışı görevleri ekleyip çıkarabilmelidir.
17. Uygulama geliştiriciler bir iş akışı görevini girdileri ve çıktıları olacak şekilde tanımlamalıdır. Girdi ve çıktıların türleri de tanımlamada belirtilmelidir. Bu tanımlamalar iş akışı sisteminin iş akışlarını doğrulamasında kullanılmalıdır.
18. Sistem sunucusuz mimariye uygun tasarlanmalı ve sunucusuz servislerle çalışabilmelerdir.
19. Sistemdeki uygulamaların sürümleri önce test sonra yayın ortamına alınmalıdır. Test ortamında hatası bulunan uygulamanın yeni sürümü yayınlanmamalıdır.
20. Sistemdeki uygulamalar için geliştirilen kodlar kod deposunda saklanmalı ve sürüm yayınlama otomasyonu üzerinden yayınlanmalıdır.

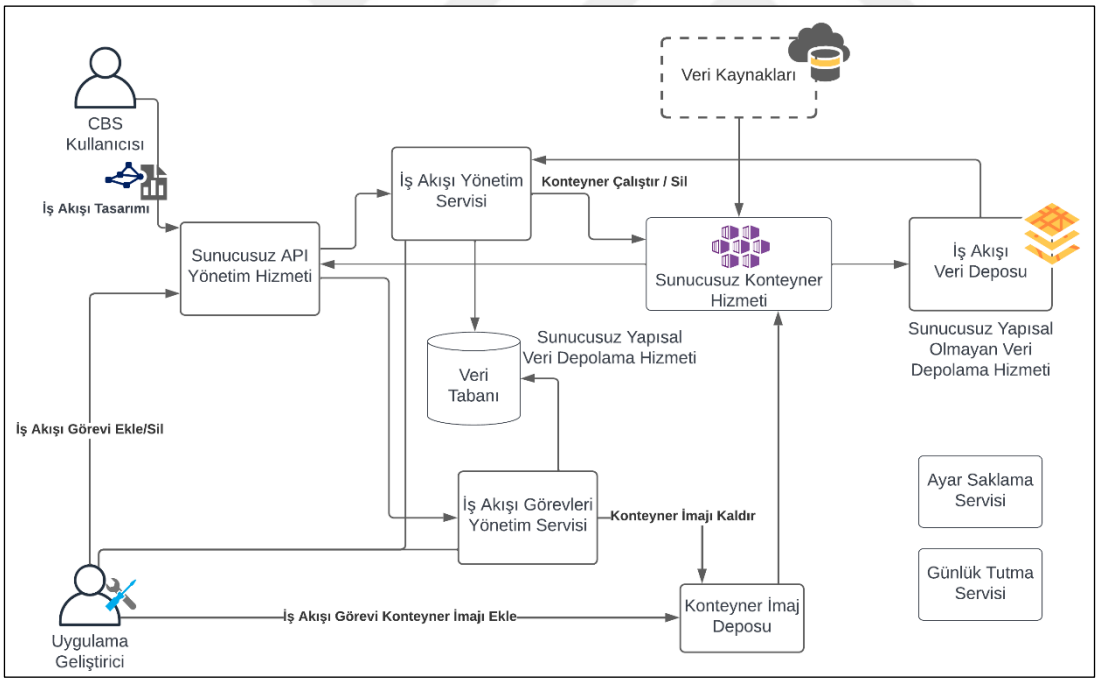
Roller

Uygulama geliştirici rolü sistemin takibi, sistemin bakımı ve yeni iş akışı görevlerinin eklenmesi ve güncellenmesinden sorumludur. Bu roldeki kullanıcılar sistemdeki izleme ve yönetim bileşenlerine erişim hakkına sahiptir. Uygulama geliştirici aynı zamanda yeni iş akışı görevlerinin kod geliştirmesini de yapar. Bu nedenle uygulama geliştirici bir iş akışı görevinin nasıl geliştirileceğinin eğitimini almış ve gerekli araç ve vasıflara sahip olmalıdır. Geliştirilen yeni iş akışı görevlerinin dokümantasyonunun yapılması da uygulama geliştiricinin sorumluluğundadır. Bu belgelendirme yapılırken iş akışı görevindeki girdilerin ve çıktıların, çalışan algoritmanın açıklamaları CBS kullanıcıları tarafından anlaşılabilecek şekilde sade ve kolay okunabilir olmalıdır. Tasarlanan sistem sunucusuz mimaride olduğundan altyapı yönetimi platform sağlayıcısı tarafından yapılacaktır, bu nedenle uygulama geliştiricinin ileri düzey bulut bilişim altyapı yönetimi bilmesi gerekmemektedir.

CBS kullanıcısı sistemde tanımlı olan iş akışı görevlerini kullanarak bir iş akışı tanımlar, sisteme ekler ve çalıştırır. Bu nedenle sistem tarafından sunulan iş akışı çalıştırma ve ekleme servislerine erişime sahip olmalıdır. Bir CBS kullanıcısı sisteme diğer CBS kullanıcıları tarafından eklenen iş akışlarını yeniden farklı girdilerle çalıştırabilir. CBS kullanıcısı iş akışı görevlerinde kullanılacak veri kaynaklarını belirler ve iş akışı görevlerinin bu veri kaynaklarıyla uyumluluğunu kontrol eder. Örneğin OGC WMS standardındaki bir harita servisi ile uyumlu olmayan bir iş akışı görevinin uyumlu olacak şekilde geliştirilmesi için uygulama geliştiriciyle birlikte çalışabilir. CBS kullanıcısı sistemdeki iş akışı görevlerini listeleyebilir, bir iş akışı görevine ait belgelendirmeye ulaşabilir.

Sistem tasarımı ve bileşenleri

Şekil 4.21'de sunulan sistem tasarımı Pakdil ve Çelik'in (2021) çalışmasından yararlanılarak tasarlanmış ve gösterilmiştir. Sunulan sistem tasarımındaki tüm bileşenler sunucusuz mimari ile çalışabilir.

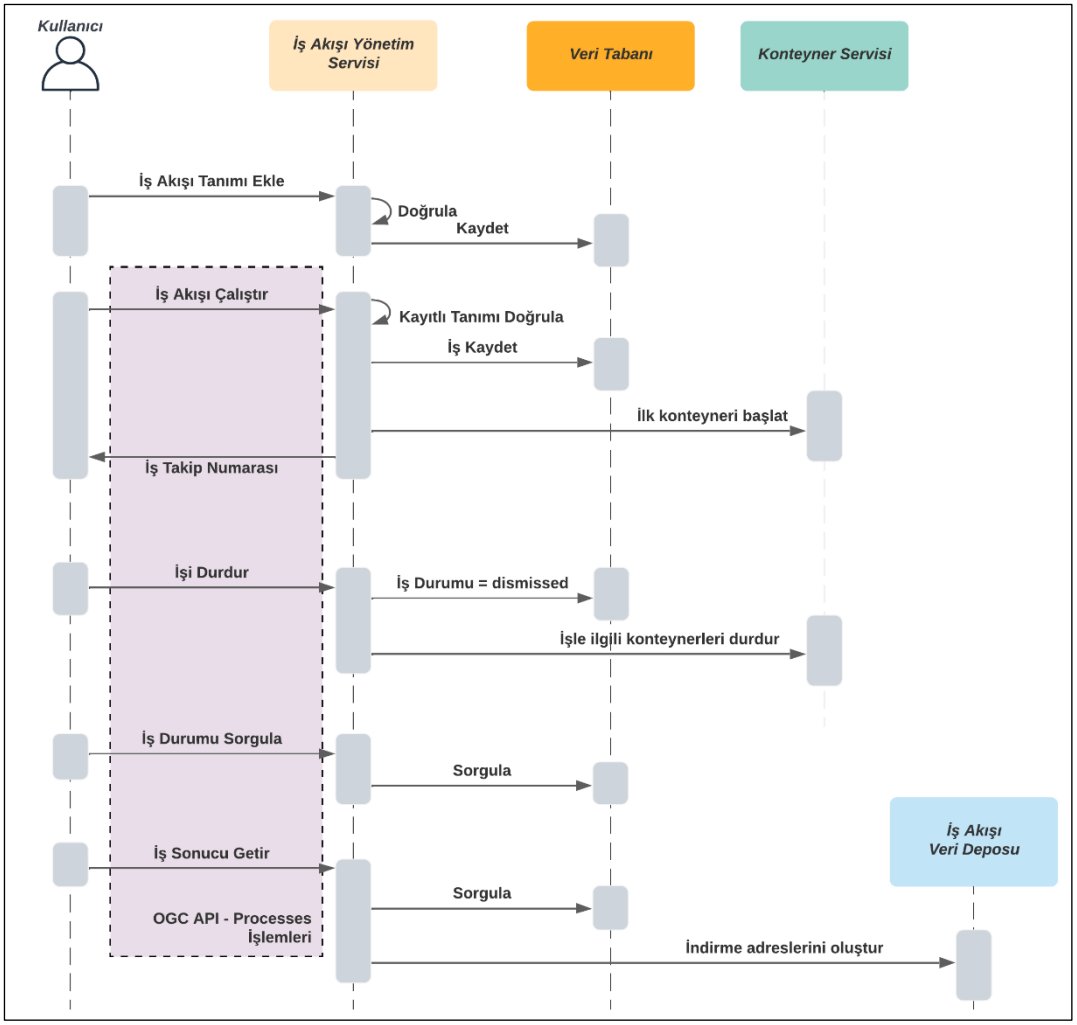


Şekil 4.21 : Mekânsal analiz iş akışı sisteminin tasarımı.

Sistemde kullanıcıların servislere erişim yapabilmesi için bir istemci uygulama konumlandırılmamıştır. Sistemde kullanıcıların kullanımına sunulan tüm servisler REST (Representational State Transfer) mimarisindedir. OGC API Processes standardı da REST mimarisine uygundur. Bu nedenle kullanıcıların bu mimariye

uygun seçebilecekleri herhangi bir REST istemci uygulama veya geliştirilebilecek bir grafik ara yüzü uygulaması sistemdeki servislerle çalışmak için kullanılabilir.

İş akışı yönetim servisi bileşeni iş akışlarının çalıştırılmasını ve takibini sağlayan servisler sunar. Bu servis kullanıcıdan gelen bir iş akışını doğrulayarak veri tabanına kaydeder. Kaydedilen iş akışının daha sonra çalıştırılabilmesini sağlar. Kullanıcı çalışan iş akışının durumunu bu servis üzerinden takip edebilir. Çalışan iş akışının ihtiyacı olan iş akışı görevlerine ait konteynerlerin yaşam döngülerini yönetir. Kullanıcıların çalışması tamamlanan iş akışının artifakt (artifact) ve parametre çıktılarına ulaşılabilmesini sağlar. Sunulan tasarımda iş akışlarının işleyecekleri veri miktarlarının büyüklüğü ve konteyner yaşam döngülerinin uzunluğu gibi nedenlerden iş akışlarının uzun süreler çalışabileceği dikkate alınmıştır. Bu nedenle iş akışı yönetim servisi bileşeninin sunduğu servisler asenkron istek-cevap modeline göre çalışmaktadır. Örneğin, bir iş akışını çalıştırma isteğini iş akışı yönetim servisi, veri tabanı ve konteyner servisiyle birlikte çalışarak işi başlatır. Başlayan iş akışına ait iş takip numarası istemciye geri döndürüldükten sonra iş akışı yönetim servisi çalışmasını tamamlar. Bu çalışma modeli sayesinde iş akışı yönetim servisi HoF modelindeki hesaplama hizmetlerinde kolaylıkla çalışabilir. Şekil 4.22'de bileşenler arasındaki etkileşim Pakdil ve Çelik'in (2021) çalışmasından yararlanılarak görselleştirilmiştir.



Şekil 4.22 : İş akışı yönetim servisi bileşeninin diğer bileşenlere çalışması ve OGC API Processes işlemlerinin kullanımı.

İş akışı yönetim servisi iş akışlarını her çalıştırmadan önce de doğrular. Bu sayede daha önce kaydedilmiş bir iş akışı tanımı çalışmaya başlamadan önce güncel iş akışı görevleri ile doğrulanır. Eğer iş akışı tanımı sistemde kayıtlı iş akışı görevleri ile artık uyumlu değilse çalıştırılmayarak kullanıcıya bilgi verilir.

İş akışı yönetim servisi iş akışı tanımı ekleme işlemi hariç OGC API Processes standartlarına uyumludur. OGC API Processes standardı mekânsal analiz işlemleri için OGC WPS 2.0'dan sonra modern web teknolojileri dikkate alınarak geliştirilmiştir (OGC, 2021). Bu standart 2021 yılının aralık ayında son haline kavuşmuştur. Henüz yeni olduğu için mevcut CBS istemcileri tarafından desteklenmemektedir ancak kısa zamanda geniş bir uygulama desteğine sahip olacağı düşünülmektedir. Bu standart hem senkron hem de asenkron mekânsal analiz işlemlerini tanımlamaktadır ancak tezde sunulan sistem tasarımı yukarıda açıklandığı üzere sadece asenkron çalışma modeline destek vermektedir.

İş akışı çalışması tamamlandığında üretilen parametre ve artifakt çıktılar veri tabanı ve iş akışı veri deposuna türüne göre kaydolur. Parametre çıktılar veri tabanına, artifakt çıktılar ise veri deposuna kaydolur. İş akışı sonucu eğer artifakt çıktı varsa bu çıktının geçici indirme adresi, eğer parametre çıktısı varsa analiz sonucunun cevabı içerisinde olacak şekilde kullanıcıya sağlanır.

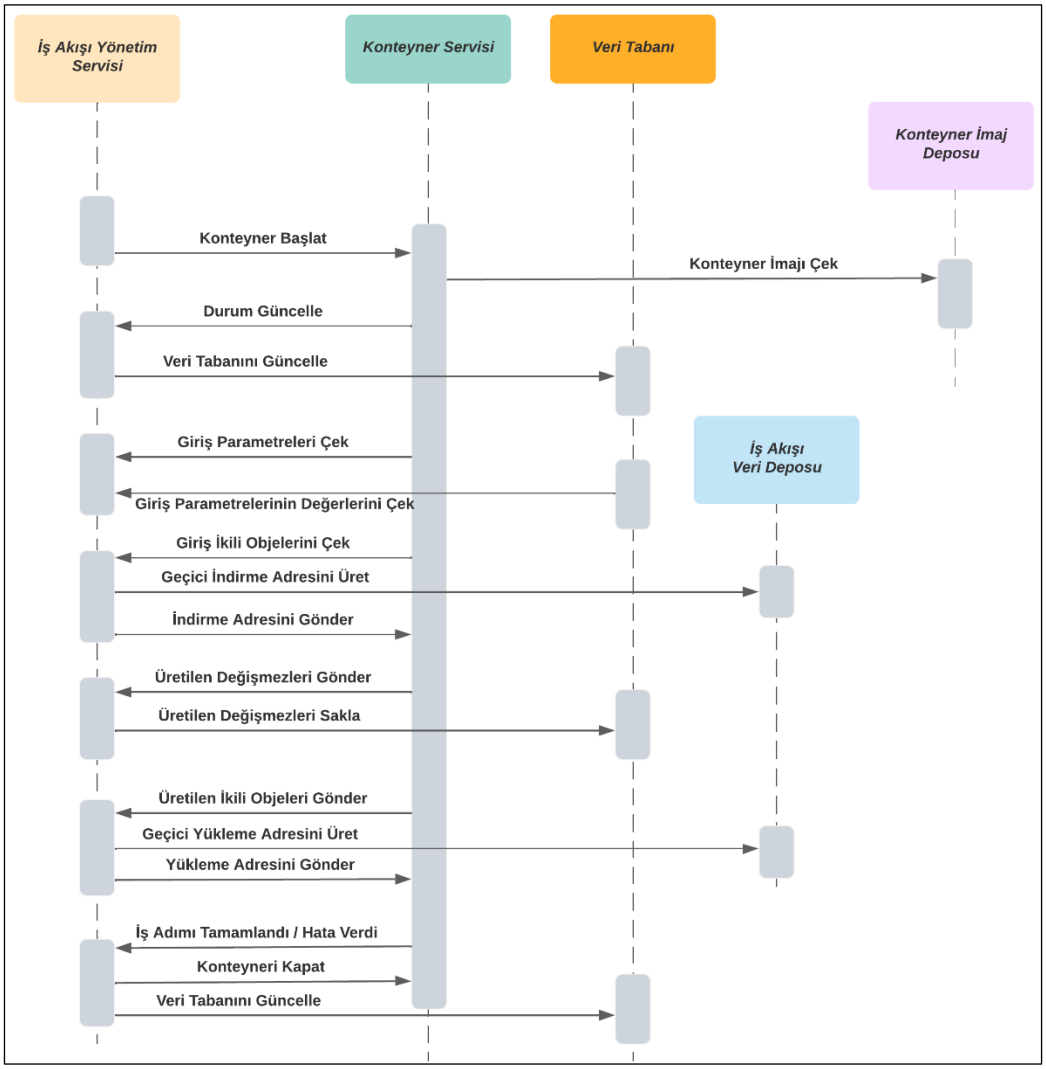
Konteyner servisi ve iş akışı yöntem sistemi birlikte çalışırlar. Çalışma esnasında iş akışı adımlarının kullandığı iş akışı görevlerine ait konteyner imajları konteyner servisi üzerinde çalıştırılır. Çalışmasını tamamlayan konteyner iş akışı yönetim sistemine bildirimde bulunarak kapatılmasını talep eder. İş akışı yönetim sistemi konteynerleri çalıştırırken konteynerin ortam değişkenlerini de günceller. Örneğin konteyner uygulaması iş akışı yönetim servisi ile hangi adres üzerinden haberleşeceğini ortam değişkenlerinden okur.

İş akışı adımları çalışma sonunda ürettikleri parametre ve artifakt türündeki veriler veri tabanı ve iş akışı veri deposuna kaydedilir. Bunun için konteynerdan iş akışı yönetim sistemine çıktının kaydı için istek yapılır. Aritfaktların veri deposuna yüklenmesi için yapısal olmayan veri depolama hizmetinin API servisi ile geçici yükleme adresleri üretilir. Parametre çıktılar ise iş akışı yönetim sistemi aracılığı ile veri tabanına kaydedilir.

Konteyner servisi iş akışı görevlerine ait konteyner imajlarının çalıştırıldığı bileşendir. Sunulan sistem tasarımında konteyner servisi HoK modelinde çalışan bir sunucusuz hizmet olarak seçilmiştir. Konteyner servisi üzerinde çalışan konteynerlerin gerektiğinde ulaşılmak istenen harici veri kaynaklarına erişim yetkisi olmalıdır. Bu nedenle ağ yapılandırması buna izin vermelidir. Konteyner servisi ile iş akışı yönetim servisinin birlikte çalışabilmesi için servis tarafından konteyner yaşam döngüsünün yönetilebileceği servisler API olarak sunulmalıdır.

İş akışı görevleri yönetim servisi bileşeni sistemdeki bir diğer REST mimarisindeki yönetim servisidir. Bu servis uygulama geliştirici rolündeki kullanıcıların kullanması için tasarlanmıştır. Servis üzerinden sisteme yeni iş görevleri eklenebilir, mevcut görevler güncellenebilir veya kaldırılabilir. İş akışı görevleri birer konteyner imajına bağlı olarak çalışırlar bu nedenle iş akışı görevi eklenmeden önce konteyner imajının varlığı konteyner imaj deposuna sorularak kontrol edilir. Silinen iş akışı görevine ait konteyner imajı da bu servis tarafından konteyner imaj deposundan silinir (Şekil 4.23).

İş akışı görevleri yönetim servisi HoF modelindeki hesaplama hizmetlerinde çalışabilecek şekilde tasarlanmıştır.



Şekil 4.23 : Konteynerlerin yaşam döngüsünün iş akışının çalışması ile ilişkisi (Pakdil ve Çelik, 2021b).

Konteyner imaj deposu iş akışı görevlerine ait konteyner imajlarının depolandığı bileşendir. Bu depo üzerindeki imajlar konteyner servisi bileşeni tarafından çekilerek çalıştırılırlar. Bu depoya aynı zamanda uygulama geliştiricilerin de erişim yetkisi vardır. Uygulama geliştirici, yeni konteyner imajlarını bu deponun sunduğu API'lar aracılığı veya sürüm yönetim sistemi yardımı ile yükleyebilir.

Sunucusuz API yönetim hizmeti iş akışı ve iş akışı görevleri servisini tek bir API adresi üzerinde toplamak ve güvenliklerini tek merkezden sağlamak üzere kullanılmıştır (Çizelge 4.11). Bu servis gelen HTTPS isteklerini arkasındaki servislere ileterek

cevapları istemcilere geri döndürür. Sunucusuz API yönetim hizmeti REST mimarisini ve servislere istekleri iletmede JSON formatını desteklemektedir.

Çizelge 4.11 : Sunucusuz API yönetimi servisinin adres tablosu ve karşılık gelen bileşenler (Pakdil ve Çelik, 2021b).

Adres	Açıklama	Bileşen
/processes/* /jobs/* /workflows/* /stepexecutions/*	OGC API Processes ve iş akışı yönetim	İş Akışı Yönetim Servisi
/tasks/*	İş akışı görevleri yönetimi	İş Akışı Görevleri Yöntem Servisi

Veri tabanı bileşeni için herhangi bir sunucusuz yapısal veya ilişkisel veri tabanı hizmeti seçilebilir. Uygulama tasarımlarının veri depolama ihtiyaçlarına göre veri depolama hizmeti türü seçilebilir. Burada dikkat edilmesi gereken bir husus da seçilen veri depolama hizmetinin uygulamanın geliştirildiği programlama dilindeki desteğinin olmasıdır.

Günlük tutma servisi sistem tasarımındaki bileşenlerde meydana gelen hata ve diğer günlük kayıtlarını tutar. Uygulama geliştirici bu servisten faydalanarak hata kayıtlarına erişebilir ve uygulamalardaki hataların kök nedenlerini bularak çözebilir. Bu servis ayrıca sistem bileşenlerine ait metrikleri de tutarak performans analizlerinin yapılmasını sağlamaktadır. Uygulama geliştirici bu bilgileri değerlendirerek uygulamaların iyileştirilmesi için kararlar ve aksiyonlar alabilir.

Ayar saklama servisi, iki yönetim servisinin veri tabanına, konteyner servisine ve iş akışı veri deposuna güvenli ulaşabilmesi için ayar parametrelerini saklar. Bu parametreler sürüm saklama otomasyonu üzerinden güncellenebilir.

Bu tez çalışmasında ayrıca iş akışı görev tanımı ve iş akışı tanımı modellenerek tasarlanmıştır. Bu tanımlar YAML (Yet Another Markup Language) dilinde geliştirilmiştir. Bu dilin seçilme nedeni ise daha okunabilir ve sade olmasıdır. Örneğin JSON dili ile karşılaştırıldığında YAML dilinde {}, [] ve "" işaretlerinin kullanılmadığı görülmektedir. Bu da YAML dilini insanlar için JSON'a göre daha kolay okunan ve yazılan bir dil yapmaktadır. Bunlarla birlikte iş akışı görevlerinde kullanılacak konteyner imajları için de bir baz konteyner imajı geliştirilmiştir. Baz konteyner imajının geliştirilmesi için yaygın olarak kullanılan Docker teknolojisi kullanılmıştır.

İş akışı görev tanımı

İş akışı görevleri iş akışları içerisinde birer adım olarak kullanılırlar. Bir iş akışı görevi bir konteyner imajı ile ilişkilidir. Bu konteyner imajı içerisinde iş akışı görevinde tanımlanan girdiler kullanılarak çalıştırılan yazılım kodu ile çıktılar üretilir. İş akışı görevleri iş akışlarının en küçük yapı taşıdır. Şekil 4.24'ta bu tez çalışmasında tanımlanan iş akışı görevi tanımı formatı gösterilmiştir.

```
Name: #Görev adı
Description: #Görev tanımı
Image: #Konteyner imajı adı
Inputs:
- Name: #Girdi adı
Type: #Girdi türü: artifact / parameter
Outputs:
- Name: #Çıktı adı
Type: #Çıktı türü: artifact / parameter
```

Şekil 4.24 : İş akışı görev tanımı formatı (Pakdil ve Çelik, 2021b).

Bir iş akışı görevini tanımlamak için kurallar belirlenmiştir. Bu kurallar Çizelge 4.12'de verilmiştir. İş akışı görevleri yönetim servisi yeni bir iş akışı görevini sisteme eklemeden önce bu kuralları kullanarak doğrulama işlemini gerçekleştirir.

Çizelge 4.12 : İş akışı görev tanımı doğrulama kuralları (Pakdil ve Çelik, 2021b).

Özellik Adı	Doğrulama Kural(lar)ı
Name	• Boş değer olamaz • İzin verilen karakterler; Alfa numerik, -, _ • Verilen değer başka bir iş akışı görevi tarafından kullanılamaz.
Description	Boş bırakılamaz
Image	• Boş değer verilemez • Konteyner imajı konteyner deposunda mevcut olmalı
Inputs[n]:Name Outputs[n]:Name	• Boş değer olamaz • İzin verilen karakterler; Alfa numerik, -, _ • Verilen değer iş akışı görevi tanımı içerisinde başka bir girdi veya çıktı tarafından kullanılamaz.
Inputs[n]:Type Outputs[n]:Type	• Boş değer olamaz • İzin verilen değerler; "artifact" veya "parameter"

İş akışı tanımı

İş akışı farklı iş akışı görevlerinin sırayla, paralel ve tekrarlı şekilde çalışmasını tanımlayan kompozisyonlardır. Şekil 4.25'te tezde sunulan iş akışı modeli tanımında kullanılan özellikler bir arada gösterilmiştir.

```yaml
Name: #İş akışı adı
Inputs:
  Parameters:
    - Name: #Parametre girdi adı
      Description: #Parametre girdi açıklaması
Outputs:
  Parameters:
    - Name: #Parametre çıktı adı
      Value: #Parametre çıktının referans değeri
  Artifacts:
    - Name: #Artifakt çıktı adı
      Value: #Artifakt çıktı referans değeri
Description: #İş akışı açıklaması
Steps:
- Id: #Adım adı
  Task: #İş akışı görev adı
  Inputs:
    Parameters:
    - Name: #Parametre girdi adı
      Value: #Parametre girdi değeri
      Description: #Parametre girdi açıklaması
    Artifacts:
    - Name: #Artifakt girdi adı
      Value: #Artifakt girdi referans değeri
      Description: #Artifakt girdi açıklaması
  Outputs:
    Artifacts:
    - Name: #Artifakt çıktı adı
    Parameters:
    - Name: #Parametre çıktı adı
- Id: #Adım adı
  Task: ForEach
  Iterate:
    Collection: #Referans değeri
    MaxConcurrency: #Her döngüde maksimum eş zamanlı hesaplama sayısı
    Steps: #Her döngüde çalıştırılacak adımlar
- Id: #Adım adı
  Task: Parallel
  Branches:
    - - #Bir dalda paralel olarak çalıştırılacak adımlar
      - ...
    - - #Bir dalda paralel olarak çalıştırılacak adımlar
      - ...
```

Şekil 4.25 : İş akışı tanımı formatı (Pakdil ve Çelik, 2021b).

İş akışı tanımı formatında kullanılan özelliklere ait kurallar Çizelge 4.13'te verilmiştir. Bu kurallar iş akışı yönetim servisi tarafından her çalışma öncesinde kullanılarak iş akışı tanımları doğrulanır.

Çizelge 4.13 : İş akışı tanımı doğrulama kuralları (Pakdil ve Çelik, 2021b).

Özellik Adı	Doğrulama Kural(lar)ı
Name	• Boş değer olamaz • İzin verilen karakterler; Alfa numerik, -, _ • Verilen değer başka bir iş akışı tarafından kullanılamaz
Inputs Outputs	Girdi ve çıktılar isteğe bağlıdır.
Inputs:Parameters[n]:Name	Boş bırakılamaz
Outputs:Parameters[n]:Name Outputs:Artifacts[n]:Name	Girdi ve çıktı isimleri boş değer olamaz
Outputs: Parameters [n]:Value Outputs: Artifacts [n]:Value	• Boş değer olamaz • Verilen değer referans formatında olmalıdır • Verilen değer iş akışı içerisindeki bir parametre veya artifakt adresi olmalıdır
Steps	İş akışında en az bir adım tanımlanmalıdır.
Steps[n]:Id	• Boş değer olamaz • İzin verilen karakterler; Alfa numerik, -, _ • Verilen değer iş akışı içerisinde başka bir adım tarafından kullanılamaz
Steps[n]:Task	• Boş değer olamaz • Bu değerlerden biri olabilir: o Kayıtlı iş akışı görevi adı o ForEach o Parallel
Steps[n]:Iterate	Bu özellik Steps[n]:Task değeri “ForEach” olarak verildiğinde kullanılabilir
Steps[n]:Iterate:Collection	• Boş değer olamaz • Referans değeri olmalıdır
Steps[n]:Iterate:MaxConcurreny	1 veya daha büyük bir değer olmalıdır.
Steps[n]:Iterate:Steps	• En az bir iş akışı adımı içermelidir • Verilen diğer iş adımı kuralları döngü içindeki kurallar için de geçerlidir

Çizelge 4.13 (devam) : İş akışı tanımı doğrulama kuralları (Pakdil ve Çelik, 2021b).

Özellik Adı	Doğrulama Kural(lar)ı
Steps[n]:Branches	• Bu özellik Steps[n]:Task değeri "Parallel" ise kullanılabilir • Çok boyutlu iş akışı adımları matrisi şekildedir • Her sütun paralel olarak çalışacak adımları içerir • En az bir iş adımları satırı olmalıdır • Verilen diğer iş adımı kuralları döngü içindeki kurallar için de geçerlidir
Steps[n]:Inputs	İsteğe bağlıdır. İş adımında girdi verilmeyecekse yok sayılır.
Steps[n]:Inputs:Parameters Steps[n]:Inputs:Artifacts	İsteğe bağlıdır. İş adımında parametre veya artifakt girdi verilmeyecekse yok sayılır.
Steps[n]:Inputs:Parameters[n]:Name Steps[n]:Inputs:Artifacts[n]:Name	• Boş değer olamaz • Verilen değer ilişkili iş akışı görevi parametre veya artifakt girdi isimlerinden biri olmalıdır
Steps[n]:Inputs:Parameters[n]: Value	• Boş değer olamaz • Verilen değer kalıp deyim (literal) veya referans değer olmalıdır
Steps[n]:Inputs:Artifacts[n]:Value	• Boş değer olamaz • Verilen değer referans değer olmalıdır
Steps[n]:Outputs	İsteğe bağlıdır. İş adımında çıktı yoksa yok sayılır
Steps[n]:Outputs:Parameters Steps[n]:Outputs:Artifacts	İsteğe bağlıdır. İş adımındaki çıktılar kullanılmayacaksa yok sayılır
Steps[n]:Outputs:Parameters[n]:Name Steps[n]:Outputs:Artifacts[n]:Name	• Boş değer olamaz • Verilen değer ilişkili iş akışı görevi parametre veya artifakt çıktı isimlerinden biri olmalıdır

Sunulan iş akışı tanımı sıralı, paralel ve tekrarlı çalışma biçimlerini desteklemektedir. Paralel ve tekrarlı çalışma biçimleri için "Parallel" ve "ForEach" iş akışı görevi isimleri sistem tarafından rezerve edilmiştir. Uygulama geliştirici kendi yükleyeceği iş akışı görevleri için bu isimleri kullanılamaz. Paralel ve tekrarlı iş adımları iç içe çalıştırılabilir. Örneğin bir paralel iş akışı dalı tekrarlı iş akışı adımı içerebilir.

Referans değerler çıktıların ve iş akışı girdilerinin iş akışı adımlarına girdi olarak aktarılabilmesi için kullanılır.

Çıktılara referans verilirken Denklem 4.1'e göre formüle edilerek yazılır.

$$\{\{step.[AdımAdı].[ÇıktıAdı]\}\} \tag{4.1}$$

İş akışı girdilerine referans verilirken Denklem 4.2'ye göre formüle edilerek yazılır.

$$\{\{input.[GirdiAdı]\}\} \tag{4.2}$$

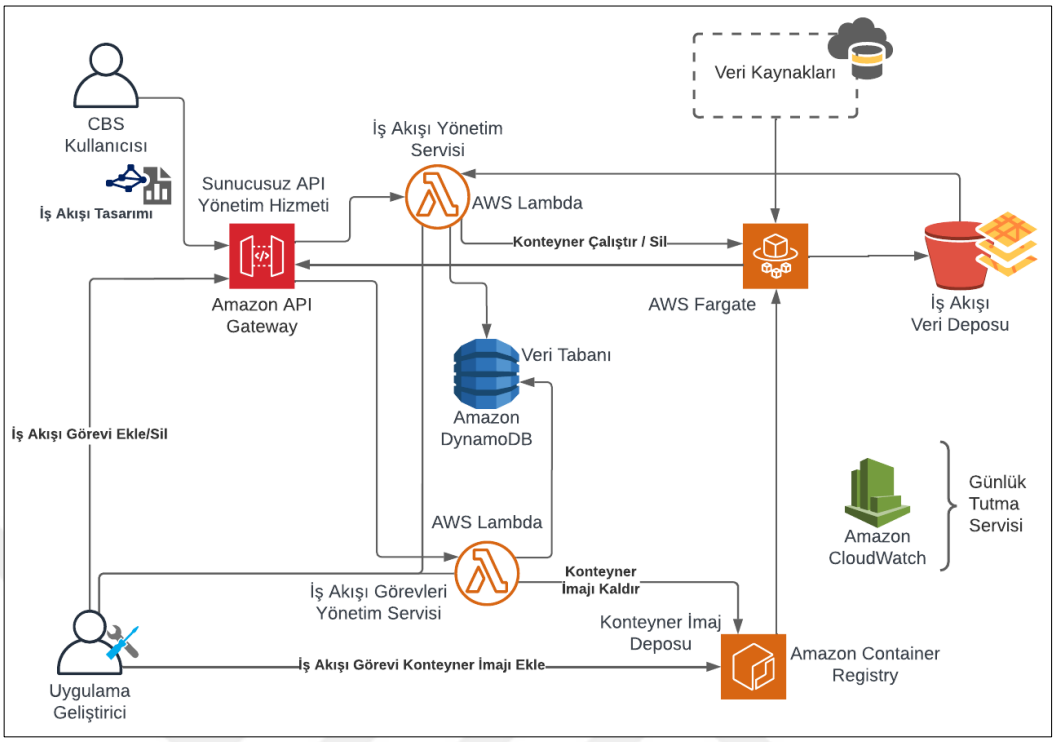
Baz konteyner imajı ve iş akışı konteyner imajının yazımı

İş akışı görevlerinde kullanılan konteyner imajları Docker konteyner imajı tanımına göre yazılmalıdır. Docker konteyner imajları kalıtım özelliklerini desteklemektedir (Cito ve diğ, 2017). Örneğin bir imaj başka bir veya birkaç imajdan türetilebilir. Bu çalışmada iş görevi tanımlarında kullanılacak konteyner imajları için bir baz konteyner imajı tasarlanmıştır (Şekil 4.26). Bu baz imaj için iş akışı yönetim servisi ve iş akışı veri deposu ile iletişim kuracak bir uygulama geliştirilmiştir. Bu uygulama iş akışı görevi için geliştirilen kodu çalıştırmadan önce kodun ihtiyacı olan girdileri çeker. Kodun çalışması sırasında üretilen günlük kayıtlarını da iş akışı yönetim sistemine iletir. Kodun çalışması tamamlandığında ise üretilen çıktıları iş akışı yönetim sistemi bileşeni ile çalışarak çıktı türüne göre veri tabanı ve iş akışı veri deposuna kaydeder.

```
FROM georchestrator:latest as function
FROM python:3.7

COPY --from=function /function_runner/function_runner /usr/bin/function_runner
RUN chmod +x /usr/bin/function_runner

....

ENV FUNCTION_COMMAND="[Çalıştırılacak Komut]"

CMD ["function_runner"]
```

Şekil 4.26 : İş akışı görevi konteyner imajı tanımı (Pakdil ve Çelik, 2021b).

4.6.2 Uygulama



Şekil 4.27 : Mekânsal analiz iş akışı sisteminin AWS bulut bilişim sağlayıcısı üzerinde uygulaması.

Mekânsal analiz iş akışı sistemi tasarımı Şekil 4.27'de gösterildiği gibi AWS bulut bilişim sağlayıcısına ait sunucusuz hizmetlerle uygulanmıştır (Pakdil ve Çelik, 2021b). Bu tez çalışmasında AWS bulut bilişim sisteminin seçilmesinin nedeni konteyner yönetimi aracının dahili otomatik ölçeklendirme ve Microsoft Azure'daki benzer servise göre daha fazla bellek sunmasıdır.

Sistem tasarımındaki iş akışı yönetim servisi ve iş akışı görevleri yönetim servisi HoF modelindeki sunucusuz hesaplama servisi olan AWS Lambda üzerinde çalışacak şekilde geliştirilmiştir. Uygulamaların programlama dili olarak C# programlama dili ve .Net Core çatkısı sürüm 3.1 seçilmiştir. Bu servislerin ikisi de HTTP isteği olaylarıyla tetiklenecek şekilde çalışmakta ve ölçeklenmektedir.

İstemcilerden gelen istekleri alarak AWS Lambda servisine HTTP isteği olayı olarak ileten Amazon API Gateway isimli servis sunucusuz API yönetim hizmeti olarak kullanılmıştır. Servislerin güvenliği ve kullanıcıların rollere göre yetkilendirilmesi için Amazon API Gateway ile AWS Cognito hizmeti birlikte kullanılabilir.

Veri tabanı hizmeti olarak Amazon DynamoDB servisi kullanılmıştır. Bu servis sunucusuz yapısal veri depolama hizmeti sunmaktadır. Anahtar-değer veri tabanı

olduğu için veri tabanı modellemesi de buna göre yapılmaktadır. Tezde sunulan sistemde OGC API Processes standartları kullanıldığı için bu standardın gerektirdiği bilgiler de saklanmalıdır.

İş akışı veri deposu olarak Amazon S3 sunucusuz yapısal olamayan veri depolama hizmeti seçilmiştir. Bu hizmet geçici URL (Uniform Resource Locator) adresleri oluşturulmasına olanak vererek objelerin yüklenmesine ve indirilmesine imkân vermektedir. Bu özellik sistem tasarımında konteynerler ve kullanıcılar ile güvenli veri alışverişini sağlamıştır. Bu hizmet üzerinde iş akışı görevlerinin çalışması sonucu oluşan artifakt çıktılar saklanmıştır.

Konteyner imaj deposu olarak Amazon Container Registry kullanılmıştır. Bu hizmet üzerinde iş akışı görevi konteyner imajları ve sisteme ait iş akışı görevi baz imajı saklanır. Saklanan imajlara verilen isimler aynı zamanda iş akışı görevi tanımlarında da saklanarak çalışma anında doğru imajın depodan çekilmesi sağlanmış olur.

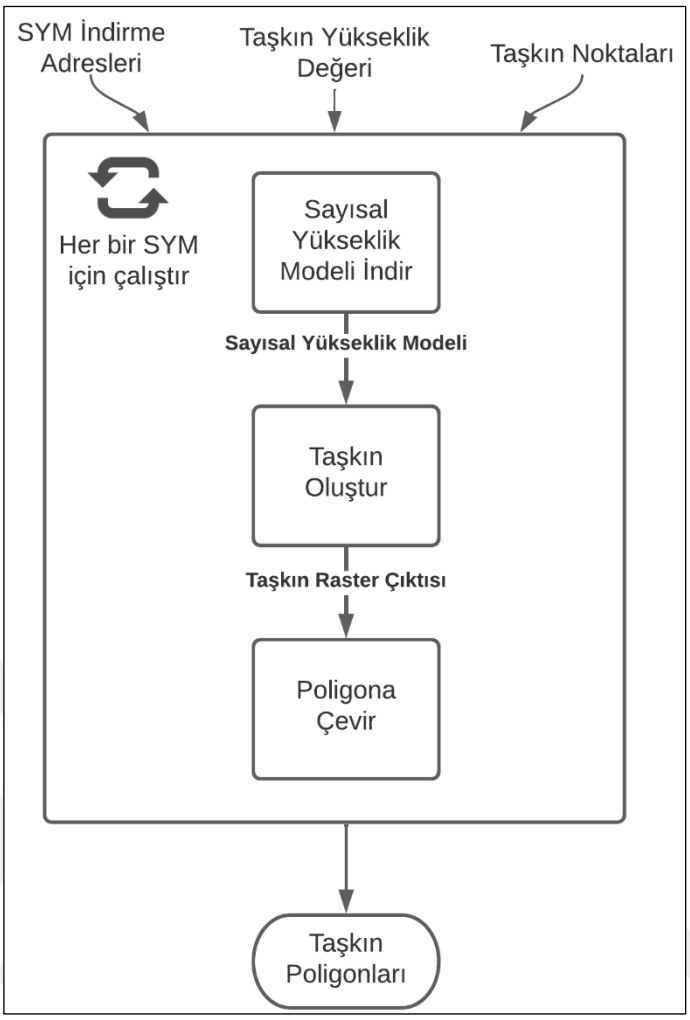
Konteyner servisi olarak Amazon Fargate kullanılmıştır. Bu hizmet HoK modelinde sunucusuz olarak Docker konteynerlerin çalıştırılmasını ve orkestrasyonunu sağlar. Bu hizmet ağ ayarları doğru yapılandırıldığında konteynerin harici veri kaynaklarına erişmesine de izin verebilmektedir.

Sistemdeki bileşenlerin birbirleri ile olan erişim yetkileri AWS IAM hizmeti üzerinden bileşenlere rol ataması yapılarak sağlanmıştır. Bunun yanında bileşenlerin ayar saklama servisi kullanarak bir destek servisi ile iletişim kurması gerekmemiştir.

Amazon CloudWatch hizmeti ile tüm bileşenler takip edilmektedir. Bu servis bileşenlerin hata günlüklerini merkezi bir yerde tutarak izlenebilmesini sağlamaktadır. Bunun yanında sistem bileşenlerine ait performans metrikleri de bu servis üzerinden takip edilebilmektedir.

Örnek mekânsal analiz iş akışları

Sistemin çalışmasının daha iyi anlaşılabilmesi için iki örnek uygulama sunulmuştur. İlk uygulamada sel taşkın analizi yapmak üzere Lawhead, 2019 tarafından geliştirilen mekânsal taşkın analizi bu sitem üzerinde iş akışına çevrilerek uygulanmıştır. Çalışmadaki mekânsal analiz ile sayısal yükseklik modelleri (SYM) kullanılarak basit bir taşkın analizi yapılmaktadır. Bu analiz Şekil 4.28'de gösterildiği gibi bir iş akışına çevrilmiştir.



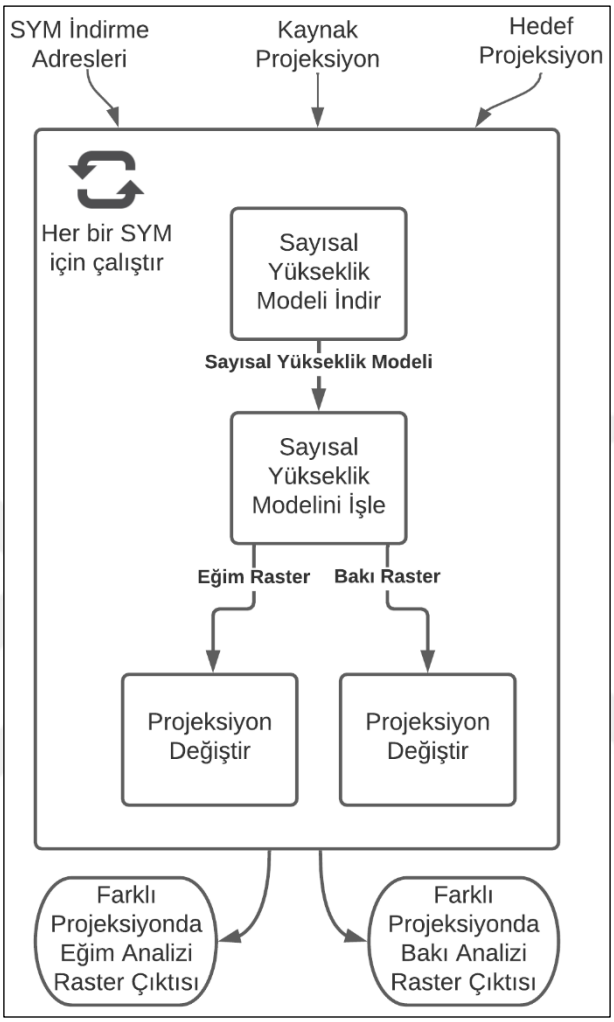
Şekil 4.28 : Mekânsal taşkın analizinin iş akış şeması (Pakdil ve Çelik, 2021b).

Çalışmada verilen Python dilindeki taşkın analizi kodu "FloodFill" isimli iş akışı görevine dönüştürülmüş ve konteyner imajı olarak yayınlanmıştır. Analizde girdi olarak kullanılacak sayısal yükseklik modellerini bir internet kaynağından indirmek üzere "DownloadUrl" isimli bir iş akışı görevi geliştirilerek yayınlanmıştır. Taşın analizi sonucu üretilen raster verinin vektör poligon verisine dönüştürülmesi adımı için de "Polygonize" isimli iş akışı görevi ve konteyner imajı yayınlanmıştır. Toplamda üç farklı görev için üç konteyner imajı oluşturulmuştur. Çalışmada girdi olarak kullanılan veriler ise Lawhead, 2019 çalışmasından alınmıştır. İş akışı Şekil 4.29'da gösterildiği gibi iş akışı tanımı kurallarına göre yazılmıştır.

```yaml
Name: TaskinAnalizi
Inputs:
  Parameters:
    - Name: dems
      Description: Sayisal yükseklik modellerinin adresleri
    - Name: inundation_level
      Description: Taşkın seviyesi yüksekliği
    - Name: seed_points
      Description: Taşkın noktaları
Outputs:
  Artifacts:
    - Name: flood_polygons
      Value: '{{step.PoligonaCevir.flood.shp}}'
Steps:
- Id: SayisalYükseklikModelleriniDon
  Task: ForEach
  Iterate:
    Collection: '{{input.dems}}'
    MaxConcurrency: 2
    Steps:
      - Id: SayisalYukseklikModeliIndir
        Task: DownloadUrl
        Inputs:
          Parameters:
          - Name: url
            Value: '{{item}}'
        Outputs:
          Artifacts:
          - Name: output
      - Id: TaskinOlustur
        Task: FloodFill
        Inputs:
          Parameters:
          - Name: inundation_level
            Value: '{{input.inundation_level}}'
          - Name: seed_points
            Value: '{{input.seed_points}}'
          Artifacts:
          - Name: terrain.asc
            Value: '{{step.SayisalYukseklikModeliIndir.output}}'
        Outputs:
          Artifacts:
          - Name: flood.asc
      - Id: PoligonaCevir
        Task: Polygonize
        Inputs:
        Parameters:
          - Name: rasterband
            Value: 1
          Artifacts:
          - Name: flood.asc
            Value: '{{step.TaskinOlustur.flood.asc}}'
        Outputs:
          Artifacts:
          - Name: flood.shp
```

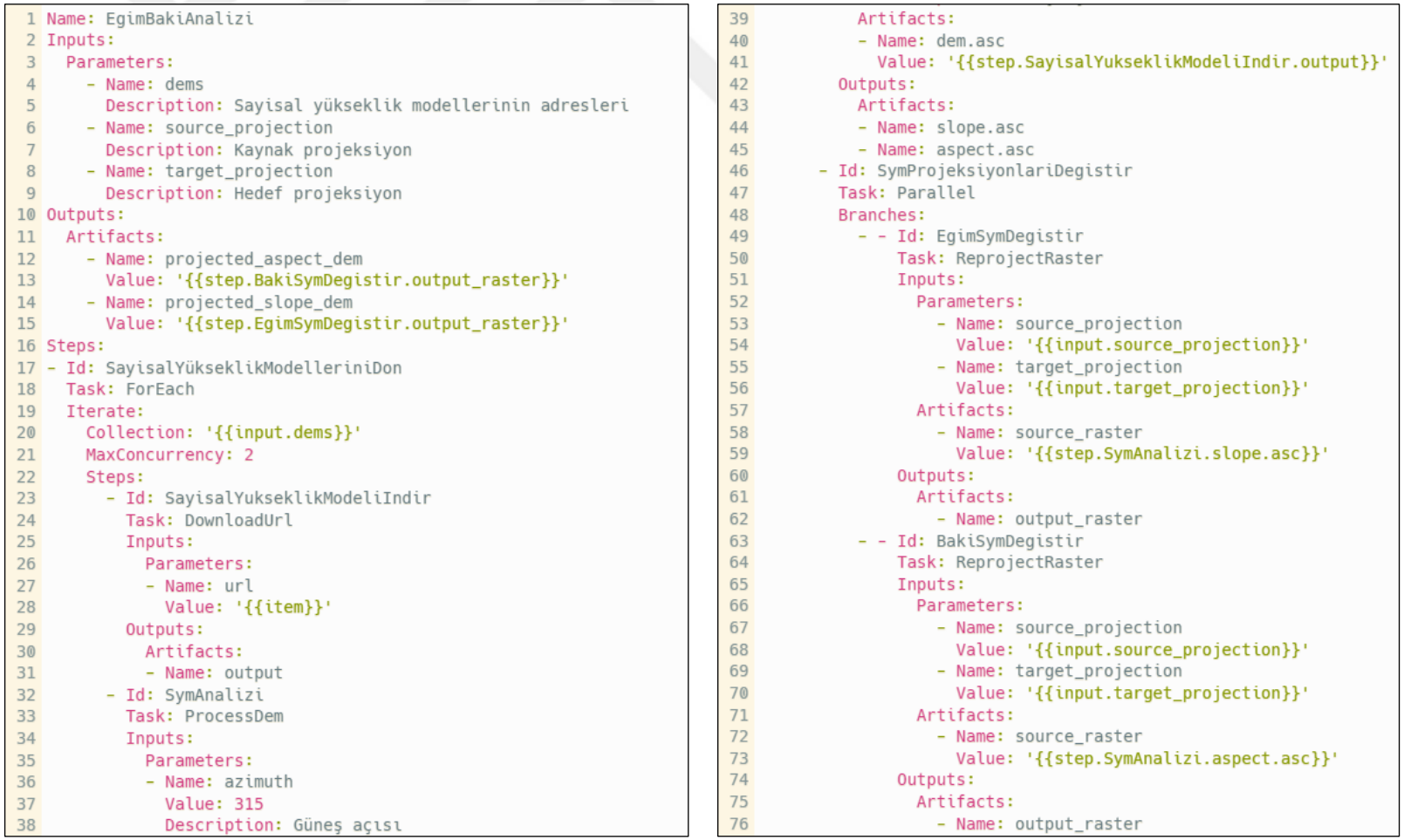
Şekil 4.29 : Taşkın analizinin iş akışı tanımına göre yazılımı (Pakdil ve Çelik, 2021b).

Bir diğer örnek uygulamada ise paralel ve döngü işlemlerinin birlikte raster analizi için kullanımı incelenmiştir. Böylece sistemdeki iki farklı türdeki dahili iş akışı görevi birlikte denenmiştir (Şekil 4.30).



Şekil 4.30 : Paralel ve tekrarlı iş akışı görevlerinin raster analizi için iş akışı içerisinde birlikte kullanımı (Pakdil ve Çelik, 2021b).

Bu iş akışında önceki örnekte kullanılan sayısal yükseklik modelini indirmek için kullanılan "DownloadUrl" isimli iş akış görevi yeniden başka bir iş akışında kullanılabilmiştir. Raster verilerine bakı ve eğim analizi yapabilmek için Lawhead, 2019 çalışmasında verilen örnek Python kodu bu çalışmada "ProcessDem" isimi iş akışı görevi ve konteyner imajı olarak yayınlanmıştır. Ayrıca iş akışına SYM verisinin projeksiyonunu değiştiren bir iş akışı görevi de yine Python dilinde geliştirilerek iş akışı görevi ve konteyner imajı olarak "ReprojectRaster" ismi ile yayınlanmıştır. Bu iş akışı görevi iş akışı içerisinde paralel olarak çalışacak şekilde kullanılmıştır. İş akışı Şekil 4.31'de gösterildiği gibi iş akışı tanımı kurallarına göre yazılmıştır.



```yaml
Name: EgimBakiAnalizi
Inputs:
  Parameters:
    - Name: dems
      Description: Sayisal yükseklik modellerinin adresleri
    - Name: source_projection
      Description: Kaynak projeksiyon
    - Name: target_projection
      Description: Hedef projeksiyon
Outputs:
  Artifacts:
    - Name: projected_aspect_dem
      Value: '{{step.BakiSymDegistir.output_raster}}'
    - Name: projected_slope_dem
      Value: '{{step.EgimSymDegistir.output_raster}}'
Steps:
- Id: SayisalYükseklikModelleriniDon
  Task: ForEach
  Iterate:
    Collection: '{{input.dems}}'
    MaxConcurrency: 2
    Steps:
      - Id: SayisalYukseklikModeliIndir
        Task: DownloadUrl
        Inputs:
          Parameters:
          - Name: url
            Value: '{{item}}'
        Outputs:
          Artifacts:
          - Name: output
      - Id: SymAnalizi
        Task: ProcessDem
        Inputs:
          Parameters:
          - Name: azimuth
            Value: 315
            Description: Güneş açısı
          Artifacts:
          - Name: dem.asc
            Value: '{{step.SayisalYukseklikModeliIndir.output}}'
        Outputs:
          Artifacts:
          - Name: slope.asc
          - Name: aspect.asc
      - Id: SymProjeksiyonlariDegistir
        Task: Parallel
        Branches:
          - - Id: EgimSymDegistir
              Task: ReprojectRaster
              Inputs:
                Parameters:
                  - Name: source_projection
                    Value: '{{input.source_projection}}'
                  - Name: target_projection
                    Value: '{{input.target_projection}}'
                Artifacts:
                  - Name: source_raster
                    Value: '{{step.SymAnalizi.slope.asc}}'
              Outputs:
                Artifacts:
                  - Name: output_raster
          - - Id: BakiSymDegistir
              Task: ReprojectRaster
              Inputs:
                Parameters:
                  - Name: source_projection
                    Value: '{{input.source_projection}}'
                  - Name: target_projection
                    Value: '{{input.target_projection}}'
                Artifacts:
                  - Name: source_raster
                    Value: '{{step.SymAnalizi.aspect.asc}}'
              Outputs:
                Artifacts:
                  - Name: output_raster
```

Şekil 4.31 : Sayısal yükseklik modellerinde eğim ve bakı analizi iş akışının geliştirilen tanımlama modeline göre yazımı (Pakdil ve Çelik, 2021b).

4.6.3 Değerlendirme

Mekânsal analiz iş akışı sistemi tasarımı ve uygulaması Çizelge 4.14'te gösterildiği gibi 12 Faktör yöntemine göre değerlendirilmiştir. Geliştirilen sistemde HoF ve HoK modelindeki sunucusuz hesaplama servisleri kullanılmıştır. Bu nedenle 12 Faktör yöntemindeki "Yönetim Süreçleri" hariç diğer prensipler doğrudan uygulanmıştır ve değerlendirilmiştir.

Çizelge 4.14 : Mekânsal analiz iş akışı sisteminin 12 Faktör yöntemine göre değerlendirilmesi.

Prensip Adı	Değerlendirme
Kod Deposu	Uygulama geliştirici geliştirdiği uygulamaları ve iş akış görevlerine ait kodları kod deposu üzerinde arşivlemektedir.
Bağımlılıklar	Bağımlılıklar prensibi HoK modeline çalışan iş akışı görevleri için değerlendirilebilir. Sistemde bu servis üzerinde çalışacak konteyner imajları oluşturulurken çalışacak kodun ihtiyaç duyacağı kütüphanelerin yüklenmesi imaj dosyası tanımlanırken belirtilir. Örneğin uygulamada GDAL, Rasterio gibi mekânsal veri kütüphaneleri kullanılacaksa bunların işletim sistemi üzerindeki bağımlılıkları imaja yüklenmelidir.
Ayarlar	Bu sistemde uygulamaların diğer destek servislerine bağlanabilmesi için gereken parametrelerin saklanması için ayar servisi kullanılmıştır. Uygulamada ise bir ayar servisine gerek olmadan AWS IAM servisinden bileşenler rol bazlı yetkilendirilerek birbirleri ile iletişim kurmuşlardır.
Destek Servisleri	Geliştirilen iş akışı yönetim servisi ve iş akışı görevleri yönetim servisi uygulaması veri tabanı ve iş akışı veri deposu destek servislerine ağ üzerinden bağlanmışlardır.
Derle, Yayınla ve Çalıştır	Uygulama geliştirici sürüm yayınlama süreçlerinin otomasyonundan sorumludur. Bu otomasyon aynı zamanda iş akışı görevlerinin konteyner imajlarının yayınlanması içinde kullanılabilir.
Süreçler	Sistem tasarımı uygulanırken HoK modelinde çalışacak konteynerlerin durumsuz çalışması gerekmektedir. Bir analiz işlemi bir sonraki analiz işlemine durum aktarmamaktadır.
Port Bağlama	Sistem tasarımında HoK modeli üzerinde çalışan konteynerlere dışardan erişim gerekmediği için port bağlama prensibinin uygulanması da gerekmemiştir.

Çizelge 4.14 (devam) : Mekânsal analiz iş akışı sisteminin 12 Faktör yöntemine göre değerlendirilmesi.

Prensip Adı	Değerlendirme
Eş Zamanlılık	Sistemdeki iş akışı yönetim servisi ve iş akışı görevleri yönetim servisi bileşenleri eş zamanlı olarak çalışmaya uygundur. Bu bileşenlerin eş zamanlı çalışmasına katkı sağlamak için sunucusuz API yönetim hizmeti servisi de kullanılmıştır. Sistemde kullanılan konteyner servisi ise aynı anda birden fazla iş akışı adımını çalıştırması gerekebileceğinden eş zamanlı çalışmaya uygundur. Uygulamada da AWS Fargate servisi bu husus dikkate alınarak seçilmiştir.
Kullanılabilirlik	Uygulamadaki yönetim servisi bileşenleri birer istek-cevap modeli ile çalışan uygulamalardır. Cevap üretildikten sonra çalışmaları sonlanır. Bu süreç ise HoF çalışma modelinde platform sağlayıcısı tarafından yönetilmektedir. Ayrıca iş akışı görevi uygulamaları da konteyner içerisinde görevlerini tamamladıktan sonra iş akışı yönetim servisi bileşenine kapanmalarının sağlanması için istek yollamaktadırlar.
Geliştirme, Test ve Yayın Ortamlarının Benzerliği	Uygulama geliştirici sistem tasarımını test ve yayın ortamlarında birbirlerine benzeyecek şekilde yayınlar. Geliştirmeler uygulama geliştiricinin bilgisayarında yapılmaktadır.
Günlük Tutma	Sistem bileşenleri tarafından üretilen günlük kayıtları için günlük tutma servisi bileşeni eklenmiştir. Uygulamada ise Amazon CloudWatch günlük tutma ve performans metriklerinin izlenmesi servisi kullanılmıştır.

Sistem tasarımı ayrıca mimari değerlendirme ölçütlerine göre Çizelge 4.15'de değerlendirilmiştir (Pakdil ve Çelik, 2021b).

Çizelge 4.15 : Mekânsal zekâ sisteminin bulut bilişim sağlayıcısı mimari değerlendirme ölçütlerine göre değerlendirilmesi.

Ölçüt Adı	Değerlendirme
Performans Verimliliği	Sistemdeki bileşenlerin performans verimlilikleri uygulama geliştiricinin sorumluluğundadır. Sistem bileşenlerinin geliştirilmesi için seçilen algoritma ve teknolojiler belirleyici olmaktadır. Aynı durum iş akışı görevlerinin gerçekleştirileceği konteyner uygulamaları için de geçerlidir. Bu durumlar haricinde sistemin uygulamasında performans verimliliğine dezavantaj olabilecek servisler kullanılmamıştır.
Güvenilirlik	Sistem bileşenleri günlük tutma servisi ile izlenerek hatalar takip edilmektedir. Kullanıcıdan gelen isteklerde yönetim servislerinde gerçekleşen hatalarda kullanıcıya istek cevabı içerisinde bilgi verilmektedir. İş akışı görevlerine ait uygulamalarda meydana gelen hatalar ise konteyner baz imajındaki uygulama üzerinden takip edilmekte ve iş akışı yönetim servisine bildirilmektedir.
Güvenlik	Sistemdeki kullanıcı ve bileşenlerin erişim yetkileri detaylı bir şekilde tanımlanmıştır. Örneğin CBS kullanıcısının iş akışları görevleri yönetim servisine erişim yetkisi bulunmamaktadır. Sistem bileşenlerinde benzer duruma örnek olarak; iş akışı yönetim servisinin konteyner imaj deposuna erişim yetkisi bulunmaması verilebilir.

5. SONUÇ VE ÖNERİLER

Sunucusuz bulut bilişim teknolojilerinde yaşanan gelişim coğrafi bilgi teknolojilerini de etkilemektedir. Bu etkileşimin sonuçları bu tez çalışmasında detaylı olarak farklı CBS senaryoları üzerinden incelenmiştir. Her bir senaryo için sistem tasarımları geliştirilerek benzer uygulamalar için öncü sistem mimarileri ortaya konmuştur.

Sunucusuz bilgi teknolojileri sunucusuz veri depolama ve hesaplama olarak iki ana başlık altında incelenmiştir. Her bir veri depolama ve hesaplama hizmet modeli özellikleri üzerinden açıklanmıştır. Açıklanan hizmet modelleri tezde sunulan sistem tasarımlarında kullanılmıştır. Özellikle bu hizmet modellerinin mekânsal bilişim yeteneklerini öne çıkaran tasarımlar geliştirilmiştir.

Sunucusuz veri depolama hizmetleri yapısal ve yapısal olmayan şeklinde iki grupta Bölüm 2.1'de incelenmiştir. Yapısal veri depolama hizmetlerinde çizge, belge, sütun ailesi, anahtar-değer gibi farklı veri yapıları olduğu görülmüştür. Bu veri yapıları için yaygın olarak kullanılan SQL yerine kendilerine ait sorgulama yöntemleri veya dilleri olduğu görülmüştür. Sunucusuz ilişkisel veri depolama hizmetleri yapısal veri depolama hizmetleri başlığı altında incelenmiştir. CBS uygulamalarında yaygın olarak kullanılan SQL dili ve ilişkisel veri modelleri için destek ilişkisel sunucusuz veri depolama hizmetleri tarafından sunulmaktadır. Yapısal veri depolama hizmetlerinin ilişkisel veri depolama hizmetlerine göre daha ölçeklenebilir olduğu ve bazı senaryolarda daha iyi performans verebileceği görülmüştür. Bu nedenle sunucusuz mimarideki CBS uygulamalarında eğer zorunlu değilse ilişkisel veri depolama yerine yapısal veri depolama hizmetlerinin tercih edilmesinin avantaj sağlayacağı düşünülmektedir.

Tez çalışmasında incelenen sunucusuz veri depolama hizmetlerinin mekânsal veri desteklerinin birbirlerinden farklı olduğu görülmüştür. Genel olarak OGC'nin koyduğu veri ve sorgulama standartları takip edilmiştir. Bulut bilişim sağlayıcıları üzerinde sunucusuz veri depolama hizmetlerini kullanarak bir CBS uygulaması geliştirilmek istendiğinde bu farklar incelenerek platform ve teknoloji seçimi yapılmalıdır.

Sunucusuz hesaplama hizmetleri fonksiyon (HoF) ve konteyner (HoK) olarak iki farklı model üzerinden incelenmiştir. HoF ve HoK modelinin birer konteyner yönetimi modeli olduğu belirtilmiştir. HoF modelinin HoK modeline göre daha fazla altyapı yönetimini soyutlayan model olduğu görülmüştür. Buna karşın işletim sistemi üzerinde çalışmadan önce kurulması gereken bağımlılıkları olan CBS uygulamalarının HoF modelinde çalışmaya uygun olmadığı değerlendirilmiştir. Bu nedenle HoK modelinin özel kurulum gerektiren mekânsal uygulamalarda kullanılması önerilmektedir.

HoF modelinde uygulamaların olay güdümlü olarak çalışabileceği belirtilmiştir. Olay üzerine tetiklenen HoF modelindeki uygulama çalışmadan önce soğuk başlatma veya sıcak başlatma süreçlerinden geçmektedir. Bu nedenle HoF modelindeki bir uygulama çok nadir çalışıyor ve çalışacağı zaman hemen cevap vermesi gerekiyorsa fonksiyonun belirli aralıklarla tetiklenerek sıcak tutulması gerekmektedir. Bu durumun maliyet optimizasyonuna etkisinin olumsuz yönde olacağı da dikkate alınmalıdır.

Tezde ortaya konan sistem tasarımlarında iki sunucusuz hesaplama modeli aralarındaki farklar dikkate alınarak seçilmiş ve kullanılmıştır. Kullanılma nedenleri tasarımlar açıklanırken irdelenerek gelecekteki çalışmalar için yönlendirici olması hedeflenmiştir.

Bu tez çalışmasında sunucusuz bulut bilişim sağlayıcılarının yaygınlıkları dikkate alınarak en çok kullanılan iki sağlayıcının sunucusuz mimarideki hizmetleri mekânsal bilişim bakış açısıyla incelenmiştir. İki sağlayıcının da Bölüm 2'de verilen sunucusuz mimarideki hizmet türlerinin tamamı için çözüm sunduğu görülmüştür. Verilen hizmetler karşılaştırıldığında birbirlerine göre farkları olduğu görülmüştür.

Sunucusuz yapısal veri depolama hizmetlerinde öne çıkan fark Microsoft Azure bulut bilişim sağlayıcısının daha çok farklı veri türünde hizmet sunması olmuştur. İlişkisel veri depolama alanında AWS bulut bilişim sağlayıcısının sunduğu PostgreSQL tabanlı sunucusuz ilişkisel veri depolama hizmeti PostGIS eklentisi ile daha fazla coğrafi veri türü desteği ve coğrafi sorgulama yetenekleri sunmaktadır. Sunucusuz yapısal olmayan veri depolama hizmeti her iki sağlayıcıda da benzer yeteneklerde ama farklı terminolojilerde sunulduğu görülmüştür. Sunucusuz HoF modelindeki hesaplama hizmetlerinde iki sağlayıcı arasındaki belirgin farklar ölçeklenebilirlik, bellek miktarı ve çalışma süresinde olduğu görülmüştür. Bir diğer önemli fark ise AWS firmasının

HoF modelindeki sunucusuz hesaplama hizmetinde Windows işletim sistemi desteğinin olmamasıdır.

İki bulut bilişim sağlayıcısının sunucusuz hizmetlerde uyguladıkları ekonomik modelin benzer olduğu görülmüştür. Her iki sağlayıcıda da kullanılan hesaplama ve veri depolama kapasitesinin birim ücreti ile bir zaman birimi üzerinden kullanım sürelerinin çarpımları ile çalışma ücretleri hesaplanmaktadır.

Tezde sunucusuz mimaride beş farklı sistem tasarımı ve uygulaması sunulmuştur. Her bir tasarım seçilen bir bulut bilişim sağlayıcısı üzerinde uygulanarak değerlendirilmiştir. Değerlendirmeler iki farklı metot ile yapılmıştır. Değerlendirme metotları Bölüm 4.1'de incelenerek açıklanmıştır. Literatürde iki metodun birden kullanıldığı sistem mimari değerlendirmesi yenidir. Tezde sunulan her bir tasarım bir CBS senaryosu ile desteklenmiştir. Senaryolar bir probleme çözüm üretmek ve tasarımın anlaşılmasını kolaylaştırmak için kurgulanmıştır. Bu senaryolar sistem tasarımlarının karmaşık olmaması için mümkün olduğunca genel amaçlı ve basit tutulmuştur.

CBS uygulamalarında sıklıkla kullanılan vektör karo ve raster karo servislerinin sistem tasarımları sunucusuz mimariye dayalı olarak Bölüm 4.2 ve 4.3'te incelenmiştir. İki mimari tasarımda uygulamalar için HoF modelindeki sunucusuz hesaplama hizmetleri yeterli olmuştur. İki tasarımın uygulamasında da mevcut açık kaynak kodlu vektör karo ve raster karo sunucusu yazılımları kullanılmıştır. Bu iki sistem tasarımı ile açık kaynak kodlu hâlihazırdaki uygulamaların CBS problemlerine sunucusuz mimariler üzerinde çalışarak çözüm üretebileceği gösterilmiştir.

Mekânsal zekâ uygulamalarında kullanılan makine öğrenmesi veya derin öğrenme kütüphaneleri grafik işlemcisi yardımıyla daha hızlı veri işleyebilmektedirler. Tezde incelenen iki farklı bulut bilişim sağlayıcısının sunduğu HoF modelindeki hesaplama hizmetlerinde grafik işlemcisi desteği görülmemiştir. Buna karşın tezde incelenen bulut bilişim sağlayıcılarının sunduğu HoK modelindeki sunucusuz hesaplama hizmetlerinde Azure Container Instances servisinin bu desteği verdiği görülmüştür. Bu nedenle mekânsal zekâ konusunda çalışma yapmak isteyen kullanıcının bu konuya dikkat ederek platform seçimi yapması önerilmektedir. Gelecekte grafik işlemcisi desteğinin sunucusuz mimarideki hesaplama hizmetlerinde giderek artacağı düşünülmektedir. Bu desteğin artmasıyla birlikte gelecekte Bölüm 4.4'te sunulan

sistem tasarımının tamamen HoF modelindeki sunucusuz hesaplama hizmetlerinde çalışabilmesinin mümkün olabileceği düşünülmektedir. Geliştirilen mekânsal zekâ uygulaması OGC WPS 2.0 standardına göre servis verecek şekilde tasarlanmıştır. Bu standardın senkron çalışma yönteminin sunucusuz mimariler için uygun olmadığı görülmüştür.

Sunucusuz hesaplama hizmetlerinin olay güdümlü olarak çalışması Bölüm 4.5'te sunulan sistem tasarımı ile incelenmiştir. Sunulan sistem tasarımı olay kaynağı olarak bir doğal afeti ele almıştır. Sunulan sistem tasarımının senaryosunda iki sonuç hedeflenmiştir. Birincisi deprem bilgilerinin kullanıcılara hızlı, ölçeklenebilir ve kolay bir şekilde ulaştırılmasıdır. İkincisi ise deprem bilgilerini yayınlayan kurumun internet sitesi üzerindeki trafik yükünün azaltılarak hizmet vermeye devam edebilmesinin sağlanmasıdır. Çalışma yaklaşık üç sene boyunca bu amaçlar için bakım gerektirmeden çalışmıştır. Sistemin bu süre boyunca hiçbir maliyet oluşturmadığı da görülmüştür. Bu sistemin uygulaması aynı zamanda kaynak kodları açık bir şekilde yayınlanmıştır. Bu çalışma ile sunucusuz mimarilerin toplum yararına nasıl mekânsal bilişim teknolojileriyle birlikte kullanılabileceği de gösterilmiştir.

Mekânsal analizlerin bulut bilişim üzerinde sunucusuz mimarilerle iş akışı şeklinde çalıştırılması için Bölüm 4.6'da sistem tasarımı ve uygulaması sunulmuştur. Sunulan tasarımın CBS kullanıcıları tarafından kullanılabilmesi için ve uygulama geliştiriciler tarafından genişletilebilmesi için iş akışı ve iş akışı görevi tanımlama modelleri de geliştirilerek sunulmuştur. Bu yeni tanım modelleri ayrıca CBS kullanıcısına paralel ve tekrarlı işlemler içeren analiz iş akışlarını tanımlayarak çalıştırabilmesini sağlamaktadır. Bunun sağlanabilmesi için konteyner orkestrasyonu uygulaması ve iş akışı görevlerinde kullanılmak üzere baz konteyner imajı da geliştirilmiştir. Sistem tasarımı iki farklı iş akışı örneği üzerinden test edilmiştir.

Tez çalışmasında mekânsal bir iş akışı sistemi tasarımında ve uygulamasında OGC API Processes literatürde ilk kez kullanılmıştır. Bu standart OGC tarafından WPS standardının devamı olarak modern web teknolojilerine ve mimarilerine uyumlu olarak yeninden geliştirilmiştir. Bölüm 4.4'teki tasarımda kullanılan OGC WPS standardı daha eski olması nedeniyle CBS uygulamaları tarafında verilen destek daha fazladır. Ancak OGC WPS ile modern web CBS uygulamaları OGC API Process'e göre daha zor geliştirilecektir, çünkü yeni nesil web uygulama çatkılarının çoğunluğu

REST mimarisi ve JSON iletişim dili dışındaki iletişim mimarileri ve dilleri ile çalışmaya doğrudan destek vermemektedirler.

Tezde sunulan tasarımlardaki tüm kullanıcı rollerinin bulut bilişim altyapıları konusunda ileri düzey bilgi sahibi olmaları gerekmemektedir. Bu da tasarımların kullanılabilirliğini arttırmakta ve bulut bilişim teknolojilerini ulaşılabilir kılmaktadır. Tezin bu özelliği ile mekânsal bilişim konusunda çalışan araştırmacılar ve uzmanlar arasında bulut bilişim teknolojileri farkındalığını ve bu teknolojilere ilgiyi arttıracağı da düşünülmüştür.

Tez çalışmasında sunulan sistem tasarımları iki farklı bulut bilişim sağlayıcısı üzerinde uygulanmıştır. Exposito Jimenez ve Zeinzer (2018) çalışmalarında bulut bilişim pazarındaki diğer aktörlerin sunduğu sunucusuz mimarideki hizmetlerin birbirlerine çok benzer olduklarını belirtmiştir. Bu nedenle bu tez çalışmasındaki sistem tasarımları her ne kadar sadece iki bulut bilişim sağlayıcısının sunduğu sunucusuz hizmetler üzerinde uygulanmış olsa da diğer bulut bilişim sağlayıcıları ile de uygulanabilir olması beklenmektedir.

Köse (2020) çalışmasında sunucusuz mimarilerin yerinde bilişim altyapıları üzerinde de uygulanabilir olduğunu belirtmiştir. Sunulan sistem tasarımları yerinde bilişim altyapıları üzerinde de uygulanabilir. Böylece verinin bulut bilişim sağlayıcısı üzerine taşınmasının mümkün olmadığı durumlarda da sunucusuz mimarilerden yararlanılabileceği düşünülmektedir.

Sunduğu yenilikçi sistem tasarımları, karşılaştırmalı incelemeler ve uygulamalar açısından bu tez kapsamındaki bu çalışmanın amacı; mekânsal bilişim dünyasını sunucusuz bulut bilişim teknolojileri konusundaki bilinci arttırmak ve bu konuda yeni kapılar açmak üzere bir kaynak olmaktır. Bununla beraber literatürde sunucusuz bulut bilişim sistem tasarımlarının ve uygulamaların nasıl değerlendirileceği konusunda yol haritası da verilmiştir. Tüm bunlara ek olarak CBS teknolojileri, mekânsal zekâ, büyük veri gibi bilişim dünyasında her geçen gün popülerliği artan teknoloji ve oluşumların sunucusuz mimarilerle etkileşimi incelenerek gelecekte katkı sağlayacağı düşünülen bir referans kaynak oluşturulmuştur.

KAYNAKLAR

Adzic, G. ve Chatley, R. (2017). Serverless computing: economic and architectural impact. *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, Paderborn Almanya. s884–889.

Amazon (2022a). Amazon Around the Globe. https://sustainability.about amazon.com/about/around-the-globe, erişim tarihi 09.05.2022.

Amazon (2022b). Invoking an AWS Lambda function from an Aurora PostgreSQL DB cluster. https://docs.aws.amazon.com/AmazonRDS/latest/Aurora UserGuide/PostgreSQL-Lambda.html, erişim tarihi 09.05.2022.

Anand, S., Johnson, A., Mathikshara, P. ve Karthik, R. (2019). Real-time GPS tracking using serverless architecture and ARM processor. *11th International Conference on Communication Systems & Networks (COMSNETS)*, Bengaluru, Hindistan, s541–543.

Antoniou, V., Morley, J., ve Haklay, M. (2009). Tiled Vectors: A Method for Vector Transmission over the Web. *Web and Wireless Geographical Information Systems*, Berlin, Heidelberg. 56–71.

Atik, M.E., Güngör, Ö., Keskin, E. ve Duran, Z. (2022). Fotogrametrik nokta bulutu verisinin makine öğrenmesi ile sınıflandırılması. *Jeodezi ve Jeoinformasyon Dergisi, 9,* 137–149.

Astrova, I., Koschel, A., Eickemeyer, C., Kersten, J., ve Offel, N. (2017). DBaaS comparison: Amazon vs. Microsoft. *International Conference on Information Society (i-Society)*, Dublin. s15–21.

Baldini, I., Castro, P., Chang, K., Cheng, P., Fink, S., Ishakian, V., …., Sutter, P. (2017). Serverless Computing: Current Trends and Open Problems, *Research Advances in Cloud Computing,* 1–20.

Baralis, E., Dalla Valle, A., Garza, P., Rossi, C., ve Scullino, F. (2017). SQL versus NoSQL databases for geospatial applications. *2017 IEEE International Conference on Big Data (Big Data)*, Boston, MA. s3388–3397.

Barbieri, L. ve Bonanni M. (2019). *Mastering Azure Serverless Computing*. Packt Publishing, Birmingham, Birleşik Krallık.

Bebortta, S., Das, S.K., Kandpal, M., Barik, R.K. ve Dubey, H. (2020). Geospatial Serverless Computing: Architectures, Tools and Future Directions. *ISPRS International Journal of Geo-Information, 9,* 311.

Bertolotto, M. ve Egenhofer, M.J. (2001). Progressive Transmission of Vector Map Data over the World Wide Web. *GeoInformatica, 5,* 345–373.

Beswick, J. (2020) Implementing geohashing at scale in serverless web applications. *AWS Compute Blog*, https://aws.amazon.com/blogs/compute/imple

menting-geohashing-at-scale-in-serverless-web-applications/, erişim tarihi 08.05.2022.

Bitner, D. (2019). Lambda MapProxy. *GitHub*, https://github.com/bitner/lambda-mapproxy/, erişim tarihi 08.05.2022.

Blower, J.D. (2010). GIS in the cloud: implementing a web map service on Google App Engine. *Proceedings of the 1st International Conference and Exhibition on Computing for Geospatial Research & Application - COM.Geo '10,* Washington, D.C., 1.

Castro, P., Ishakian, V., Muthusamy, V., ve Slominski, A. (2019). The rise of serverless computing. *Communications of the ACM, 62,* 44–54.

Chapin, J. ve Roberts, M. (2020). *Programming AWS Lambda*. O'Reilly Media, Inc, Amerika Birleşik Devletleri.

Cito, J., Schermann, G., Wittern, J.E., Leitner, P., Zumberi, S., ve Gall, H.C. (2017). An Empirical Analysis of the Docker Container Ecosystem on GitHub. *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, Buenos Aires, Arjantin. s323–333.

Cumhuriyet Gazetesi (2015) "4.2'lik deprem yetti: Kandilli'nin sitesi çöktü.". *Cumhuriyet Gazetesi*, 16 Kasım 2015, Erişim adresi https://www.cumhuriyet.com.tr/haber/42lik-deprem-yetti-kandillinin-sitesi-coktu-420795

Daher, Z., ve Hajjdiab, H. (2018). Cloud Storage Comparative Analysis Amazon Simple Storage vs. Microsoft Azure Blob Storage. *International Journal of Machine Learning and Computing*, *8*, 85–89.

De Beukelaar, I.T.Y. (2018). *Cartographic implications of Vector Tile technology*. (Master Tezi). Utrecht Üniversitesi, Hollanda.

De Sousa, L.M., De Jesus, J.M., Čepicky, J., Kralidis, A.T., Huard, D., Ehbrecht, C., ..., Eberle, J. (2019). PyWPS: overview, new features in version 4 and existing implementations. *Open Geospatial Data, Software and Standards*, *4*, 13.

Döş, M.E. ve Uysal, M. (2019). Uzaktan algılama verilerinin derin öğrenme algoritmaları ile sınıflandırılması *Türkiye Uzaktan Algılama Dergisi, 1,* 28-34.

Exposito Jimenez, V.J. ve Zeiner, H. (2018). Serverless Cloud Computing: A Comparison Between "Function as a Service" Platforms. *Computer Science & Information Technology, Academy & Industry Research Collaboration Center (AIRCC)*, s15–22.

Ferreira, D. R. G. (2014) *Using Neo4J for Geospatial Data Storage and Integration* (Master Tezi). Madeira Üniversitesi, Portekiz.

Fitzsimmons, S. (2017). Deploying Tegola on AWS Lambda. *Medium*. https://medium.com/@mojodna/deploying-tegola-on-aws-lambda-a7f74ec8f0df/, erişim tarihi 08.05.2022.

Gandhi, U. (2021) Web Mapping with QGIS2Web (QGIS3). *QGIS Tutorials and Tips,*

http://www.qgistutorials.com/uk/docs/3/web_mapping_with_qgis2web.html, erişim tarihi 08.05.2022.

Google Trends (2022) https://trends.google.com/, erişim tarihi 08.05.2022.

Gulabani, S. (2015). *Amazon S3 Essentials*. Packt Publishing, Birmingham, Birleşik Krallık.

Güney C. ve Çelik R.N. (2020). Harita ve Kadastro Mühendisliğinin Dijital Ekosistemde Hayatta Kalabilmesi için Paradigma Değişimi: Mekânsal Zekâ, *17. Türkiye Harita Bilimsel ve Teknik Kurultayı*, İstanbul, Türkiye. https://www.hkmo.org.tr/resimler/ekler/3c2e21d6dfb7bea_ek.pdf

Grolinger, K., Higashino, W.A., Tiwari, A., ve Capretz, M.A. (2013). Data management in cloud environments: NoSQL and NewSQL data stores. *Journal of Cloud Computing: Advances, Systems and Applications*, *2*, 22.

Hashimoto, N. (2015). *Amazon S3 Cookbook.* Packt Publishing, Birmingham, Birleşik Krallık.

Hecht, R. ve Jablonski, S. (2011). NoSQL evaluation: A use case oriented survey. *2011 International Conference on Cloud and Service Computing*, Hong Kong, Çin. s336–341.

Ifrah, S. (2020). *Getting Started with Containers in Azure: Deploy, Manage, and Secure Containerized Applications*. Apress, Berkeley, CA, Amerika Birleşik Devletleri.

Jain, P., Munjal, Y., Gera, J. ve Gupta, P. (2020). Performance Analysis of Various Server Hosting Techniques. *Procedia Computer Science, 173,* 70–77.

Jackson, D. ve Clynch, G. (2018). An Investigation of the Impact of Language Runtime on the Performance and Cost of Serverless Functions. *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, Zurih. s154–160.

Jangda, A., Pinckney, D., Brun, Y., ve Guha, A. (2019). Formal foundations of serverless computing. *Proceedings of the ACM on Programming Languages*, *3*, 1–26.

Ji, X., Chen, B., Huang, Z., Sui, Z., ve Fang, Y. (2012). On the use of cloud computing for geospatial workflow applications. *20th International Conference on Geoinformatics*, Hong Kong, Çin, s1–6.

Jonas, E., Schleier-Smith, J., Sreekanti, V., Tsai, C.-C., Khandelwal, A., Pu, Q., …, Patterson, D. A. (2019). *Cloud Programming Simplified: A Berkeley View on Serverless Computing*. (Rapor No: UCB/EECS-2019-3). UC Berkeley

Kalid, S., Syed, A., Mohammad, A., ve Halgamuge, M.N. (2017). Big-data NoSQL databases: A comparison and analysis of "Big-Table", "DynamoDB", and "Cassandra.". *2nd International Conference on Big Data Analysis*, Beijing, Çin, s89–93.

Kamel Boulos, M.N., Peng, G., ve VoPham, T. (2019). An overview of GeoAI applications in health and healthcare. I*nternational Journal of Health Geographics*, *18*, 7.

Kapadia, A., Rajana, K. ve Varma, S. (2015). *OpenStack Object Storage (Swift) Essentials.* Packt Packt Publishing, Birmingha, Birleşik Krallık.

Kerkhove, T. (2021). Autoscaling Azure Container Instances with Azure Serverless. *Tom Kerkhove Blog*. https://blog.tomkerkhove.be/2021/01/02/autoscaling-azure-container-instances-with-azure-serverless/, erişim tarihi 08.05.2022.

Kiener, M., Chadha, M., ve Gerndt, M. (2021). Towards Demystifying Intra-Function Parallelism in Serverless Computing. *Proceedings of the Seventh International Workshop on Serverless Computing (WoSC7) 2021*, Sanal Etkinlik, Kanada, s42–49.

Kim, Y. ve Lin, J. (2018). Serverless Data Analytics with Flint. *11th International Conference on Cloud Computing (CLOUD)*, San Francisco, CA, 451–455.

Kleppmann, M. (2017). *Designing data-intensive applications: the big ideas behind reliable, scalable, and maintainable systems*. O'Reilly Media, Boston.

Köse, M. (2020). *Bilişim Altyapısı Üzerine Sunucusuz Mimari Platformu İnşa Etme*. (Master Tezi). Süleyman Demirel Üniversitesi, Fen Bilimleri Enstitüsü, Isparta.

Krämer, M., Würz, H.M., ve Altenhofen, C. (2021). Executing cyclic scientific workflows in the cloud. *Journal of Cloud Computing: Advances, Systems, 10*, 25.

Lawhead, J. (2019). *Learning geospatial analysis with Python.* Üçüncü Baskı. Packt Publishing, Birmingham, Birleşik Krallık.

Lee, H., Satyam, K. ve Fox, G. (2018). Evaluation of Production Serverless Computing Environments. *11th International Conference on Cloud Computing (CLOUD)*, IEEE, San Francisco, CA. 442–450.

Léger, Y. ve Broshar A. (2021) FaaS vs CaaS: Comparing Use Cases and Responsibilities. https://www.koyeb.com/blog/faas-vs-caas-comparing-use-cases-and-responsibilities, erişim tarihi 08.05.2022.

Li, L., Hu, W., Zhu, H., Li, Y., ve Zhang, H. (2017). Tiled vector data model for the geographical features of symbolized maps. *PLoS ONE*, *12*, e0176387.

Li, S., Dragicevic, S., Castro, F.A., Sester, M., Winter, S., Coltekin, A., …, Cheng, T. (2016). Geospatial big data handling theory and methods: A review and research challenges. *ISPRS Journal of Photogrammetry and Remote Sensing, 115*, 119–133.

Lopez, E., Béjar, R., Barrera, J., Lopez-Pellicer, F. J., Rodríguez, A. F. ve Abad P. (2017). *Support for vector tiles in INSPIRE view services* [PowerPoint sunumu]. erişim adresi https://inspire.ec.europa.eu/sites/default/files/presentations/INSPIRE2017_VectorTiles.pdf

Lynn, T., Rosati, P., Lejeune, A., ve Emeakaroha, V. (2017). A Preliminary Review of Enterprise Serverless Cloud Computing (Function-as-a-Service) Platforms. *2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, Hong Kong. s162–169.

Maissen, P., Felber, P., Kropf, P., ve Schiavoni, V. (2020). FaaSdom: a benchmark suite for serverless computing. *Proceedings of the 14th ACM International Conference on Distributed and Event-Based Systems*, Montreal Quebec Canada. s73–84.

Malawski, M., Gajek, A., Zima, A., Balis, B. ve Figiela, K. (2020). Serverless execution of scientific workflows: Experiments with HyperFlow, AWS Lambda and Google Cloud Functions. *Future Generation Computer Systems, 110*, 502–514.

MapServer (2022). https://hub.docker.com/r/mapserver/mapserver, erişim tarihi 08.05.2022.

Marroquín, R., Müller, I., Makreshanski, D., ve Alonso, G. (2018). Pay One, Get Hundreds for Free: Reducing Cloud Costs through Shared Query Execution. *Proceedings of the ACM Symposium on Cloud Computing*, Carlsbad CA, Amerika Birleşik Devletleri. s439–450.

Maslov, V. ve Petrashenko, A. (2021). Distributed Serverless Computing Orchestration Based on Finite Automaton. *Advances in Computer Science for Engineering and Education IV*, Springer International Publishing, Cham. 290–303.

Mete, M.O. ve Yomralıoğlu, T. (2021a). Implementation of serverless cloud GIS platform for land valuation. *International Journal of Digital Earth, 14*, 836–850.

Mete, M.O. ve Yomralıoğlu, T. (2021b). Açık Kaynaklı Bulut CBS Yardımıyla Kitlesel Taşınmaz Değerleme Uygulaması. *Harita Dergisi, 165*, 28-42.

Microsoft (2022a). Index geospatial data with Azure Cosmos DB. https://docs .microsoft.com/en-us/azure/cosmos-db/sql/sql-query-geospatial-index, erişim tarihi 09.05.2022.

Microsoft (2022b). Querying geospatial data with Azure Cosmos DB. https://docs .microsoft.com/en-us/azure/cosmos-db/sql/sql-query-geospatial-query, erişim tarihi 09.05.2022.

Microsoft (2022c). OGC Static Geometry Methods. https://docs.microsoft.com/en-us/sql/t-sql/spatial-geometry/ogc-static-geometry-methods, erişim tarihi 09.05.2022.

Microsoft (2022d). Tutorial: Implement IoT spatial analytics by using Azure Maps. https://docs.microsoft.com/en-gb/azure/azure-maps/tutorial-iot-hub-maps, erişim tarihi 09.05.2022.

Microsoft (2022e). Azure Functions hosting options. https://docs.microsoft.com/en-us/azure/azure-functions/functions -scale, erişim tarihi 09.05.2022.

Nadon, J. (2017). *Website Hosting and Migration with Amazon Web Services*. Apress, Berkeley, CA.

Netek, R., Masopust, J., Pavlicek, F., ve Pechanec, V. (2020). Performance Testing on Vector vs. Raster Map Tiles—Comparative Study on Load Metrics. *ISPRS International Journal of Geo-Information, 9,* 101.

Nickoloff, J., Kuenzli, S., ve Fisher, B. (2019). *Docker in action*. İkinci Baskı. Manning Publications, Shelter Island, NY, Amerika Birleşik Devletleri.

Obey, J. (2022) Best practices for building serverless applications that follow AWS's Well-Architected Framework. *Datadog. https://www.datadoghq.com/blog/well-architected-serverless-applications-best-practices,* erişim tarihi 08.05.2022.

OGC (2010). OpenGIS Web Map Tile Service Implementation Standard. https://www.ogc.org/standards/wmts, erişim tarihi 09.05.2022.

OGC (2006). Web Map Service. https://www.ogc.org/standards/wms, erişim tarihi 09.05.2022.

OGC (2018). Vector Tiles Engineering Report. https://docs.ogc.org/per/17-041.html, erişim tarihi 09.05.2022.

OGC (2021). OGC API - Processes - Part 1: Core. https://docs.ogc.org/is/18-062r2/18-062r2.html, erişim tarihi 09.05.2022.

Pakdil, M.E. ve Çelik, R.N. (2021a). Design of a Serverless OGC WPS Based Geoprocessing Service Solution. *6th International Conference on Smart City Applications*, Karabük Üniversitesi, s425–430.

Pakdil, M.E. ve Çelik, R.N. (2021b). Serverless Geospatial Data Processing Workflow System Design. *ISPRS International Journal of Geo-Information, 11,* 20.

Pakdil, M. E. ve Çelik, R. N. (2023, baskıda). Bulut bilişimde sunucusuz mimariler ile coğrafi bilgi teknolojilerinin kullanımı üzerine bir inceleme. *Jeodezi ve Jeoinformasyon Dergisi, 10(1),* 1-15

Patterson, S. (2019). *Learn AWS serverless computing*. Packt Publishing, Birmingham, Birleşik Krallık.

Piancazzo, D. (2022). *A comparison between azure cosmos DB and elasticsearch : A case study on cloud databases* (Lisans Bitirme Çalışması). http://urn.kb.se/resolve?urn=urn:nbn:se:mdh:diva-57444

Poccia, D. (2020). New for AWS Lambda – Container Image Support. *AWS News Blog*. https://aws.amazon.com/blogs/aws/new-for-aws-lambda-container-image-support/, erişim tarihi 08.05.2022.

Pothecary, R. (2021). *Running Microsoft Workloads on AWS: Active Directory, Databases, Development, and More.* Apress, Berkeley, CA, Amerika Birleşik Devletleri.

Richter, F. (2022). Amazon Leads $180-Billion Cloud Market, *statista*, https://www.statista.com/chart/18819/worldwide-market-share-of-leading-cloud-infrastructure-service-providers/, erişim tarihi 08.05.2022.[1]

Sahay, R. (2020). *Microsoft Azure Architect Technologies Study Companion: Hands-on Preparation and Practice for Exam AZ-300 and AZ-303*. Apress, Berkeley, CA, Amerika Birleşik Devletleri.

Sampé, J., Sánchez-Artigas, M., García-López, P., ve París, G. (2017). Data-driven serverless functions for object storage. *Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference*, ACM, Las Vegas Nevada. s121–133.

Sánchez, J., Nothstein, M., Neic, A., Huang, Y.-L., J. Prassl, A., Klar, J., …, Axel, L. (2020). openCARP: An Open Sustainable Framework for In-Silico Cardiac Electrophysiology Research. *2020 Computing in Cardiology*, 1-4.

Satapathi, A. ve Mishra, A. (2021). *Hands-on Azure Functions with C#: Build Function as a Service (FaaS) Solutions*. Apress, Berkeley, CA, Amerika Birleşik Devletleri.

Sbarski, P., Cui, Y., ve Nair, A. (2022). *Serverless architectures on AWS*. İkinci Baskı. Manning Publications, Shelter Island, NY, Amerika Birleşik Devletleri.

Siddiqui, T., Siddiqui, S.A., ve Khan, N.A. (2019). Comprehensive Analysis of Container Technology. *4th International Conference on Information Systems and Computer Networks (ISCON)*, Mathura, Hindistan. s218–223.

Solanki, J. (2021). Serverless Database – Everything you Need to Know, *Simform Blog*, https://www.simform.com/blog/serverless-databases/, erişim tarihi 08.05.2022.

Soueidi, C. (2015). *Microsoft Azure Storage Essentials*. Packt Publishing, Birmingham, Birleşik Krallık.

Souissia, N., ve Mainguenaudb, M. (2014). Database Server Models for WMS and WFS Geographic Web Services. *Geographical Information Systems: Trends and Technologies*, s142.

Tomey, S. (2017). Geographic Spatial Analysis with Azure Data Lake Analytics (ADLA). *Adatis.* https://adatis.co.uk/geographic-spatial-analysis-with-azure-data-lake-analytics-adla/, erişim tarihi 08.05.2022.

Top, T., Yıldırım, T., Gültekin, T., Kuşdemir, Y., Yenen, M., Koşak, E., Kahraman, F. (2011) 5.9'da bile çuvalladık. *Hürriyet Gazetesi*. Erişim adresi https://www.hurriyet.com.tr/gundem/5-9da-bile-cuvalladik-17838905

URL-1 <https://registry.opendata.aws/>, erişim tarihi 08.05.2022.

URL-2 <https://github.com/aws/containers-roadmap/issues/88>, erişim tarihi 08.05.2022.

URL-3 < https://tegola.io>, erişim tarihi 08.05.2022.

URL-4 <https://mapproxy.org/>, erişim tarihi 08.05.2022.

URL-5 <https://dl.acm.org/sig/sigspatial>, erişim tarihi 08.05.2022.

URL-6 <https://github.com/Geomates/KandilliEarthquakeNotifier>, erişim tarihi 08.05.2022.

Vemula, R. (2019). *Integrating Serverless Architecture: Using Azure Functions, Cosmos DB, and SignalR Service*. Apress, Berkeley, CA, Amerika Birleşik Devletleri.

Vohra, D. (2018). *Amazon Fargate Quick Start Guide: Learn How to Use AWS Fargate to Run Containers with Ease*. Packt Publishing, Birmingham, Birleşik Krallık.

VoPham, T., Hart, J.E., Laden, F., ve Chiang, Y.-Y. (2018). Emerging trends in geospatial artificial intelligence (geoAI): potential applications for environmental epidemiology. *Environ Health, 17*, 40.

Weaver, E. (2021): AWS Aurora Serverless v2: Architecture, Features, Pricing, and Comparison with Fauna. *Fauna*. https://fauna.com/blog/compare-aws-aurora-serverless-v2-architecture-features-pricing-vs-fauna, erişim tarihi 08.05.2022.

Wiggins, A. (2012). *The Twelve-Factor App*. https://12factor.net, erişim tarihi 08.05.2022.

Wittig, M., Wittig, A., ve Whaley, B. (2018). *Amazon Web Services in action*. İkinci Baskı. Manning Publications, Shelter Island, NY, Amerika Birleşik Devletleri.

Yusuf, S. (2021): CaaS Services Through AWS, Azure, and Google Cloud. *Thundra Blog*. https://blog.thundra.io/caas-services-through-aws-azure-and-google-cloud, erişim tarihi 08.05.2022.

Zois T. (2021). *The Evolution of Serverless Services*. doi: 10.13140/RG.2.2.33924.86407.

ÖZGEÇMİŞ

Ad-Soyad **:** Mete Ercan PAKDİL

ÖĞRENİM DURUMU:

- **Lisans** **:** 2011, İstanbul Teknik Üniversitesi, İnşaat Fakültesi, Geomatik Mühendisliği Bölüm
- **Yüksek lisans** **:** 2014, İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Geomatik Mühendisliği Yüksek Lisans Programı

MESLEKİ DENEYİM VE ÖDÜLLER:

- 2022 yılında ISPRS International Journal of Geo-Information dergisine yazdığı "Serverless Geospatial Data Processing Workflow System Design " başlıklı makale ile Mott MacDonald MMDV (Mott MacDonald Digital Ventures) bölgesi "En İyi Makale (Best Technical Paper)" ödülünü aldı.
- 2019 Kasım ayından bu yana halen İngiltere'de Mott MacDonald firmasında Yazılım Uzmanı ve Yönetici olarak çalışmaktadır.
- 2019 Ekim-Kasım ayları arasında İngiltere'de DCSL Yazılım firmasında Yazılım Mimarı olarak görev yaptı.
- 2017 -2019 yılları arasında Hollanda'da Exact firmasında Uzman Yazılım Mühendisi ve Mimar olarak görev yaptı.
- 2017 yılında Pegasus Havayolları'nda geliştirilen uçuş haritaları üretim ve dağıtım sistemi ile ESRI CBS Özel Başarı (Special Achievement in GIS) ödülü aldı.
- 2014 yılında İstanbul Teknik Üniversitesi'nde Geomatik Mühendisliği bölümünde yüksek lisansını tamamladı.
- 2013 yılında Bilim, Sanayi ve Teknoloji Bakanlığı tarafından yürütülen SAN-TEZ Programı kapsamında "Ulusal veUluslararası Sayısal Navigasyon Bilgisi ve Uçuş Haritalarının Üretim ve Yönetim Sisteminin Tasarımı, Oluşturulması. I. Üretim Fazı:Kalkış Haritalarının Üretimi" başlıklı proje ile yüksek lisans tezi projesi için destek kazandı.
- 2012 -2017 yılları arasında Türkiye'de Pegasus Hava Taşımacılığı firmasında Navigasyon Lideri olarak görev yaptı.
- 2011 -2012 yılları arasında Türkiye'de Universal Bilgi Teknolojileri firmasında Yazılım Uzmanı olarak görev yaptı.
- 2011 yılında Haziran – Eylül ayları arasında Amerika Birleşik Devletleri'nde ESRI firmasının genel merkezinde dört ay staj yapmak üzere seçildi.
- 2011 yılında İstanbul Teknik Üniversitesi'nde Geomatik Mühendisliği bölümünde lisansını tamamladı.

- 2010 yılında TÜBİTAK'ın 2209 Üniversite Öğrencileri Yurt İçi / Yurt Dışı Araştırma Projeleri Destekleme Programı'nda "İnternet Tabanlı Bir Uzaktan Eğitim Laboratuvarı: Geomatik Uygulamalar" başlıklı proje ile lisans bitirme ödevi için destek kazandı.

DOKTORA TEZİNDEN TÜRETİLEN YAYINLAR, SUNUMLAR VE PATENTLER:

- **Pakdil, M.E.** ve Çelik, R.N. (2021a). Design of a Serverless OGC WPS Based Geoprocessing Service Solution. *6th International Conference on Smart City Applications,* Karabük Üniversitesi, s425–430**.**
- **Pakdil, M.E.** ve Çelik, R.N. (2021b). Serverless Geospatial Data Processing Workflow System Design. *ISPRS International Journal of Geo-Information, 11,* 20.
- **Pakdil, M. E.** ve Çelik, R. N. (2023, baskıda). Bulut bilişimde sunucusuz mimariler ile coğrafi bilgi teknolojilerinin kullanımı üzerine bir inceleme. *Jeodezi ve Jeoinformasyon Dergisi*, *10(1),* 1-15

DİĞER YAYINLAR, SUNUMLAR VE PATENTLER:

- Bıçakçı, Y. S., Sarıca, B., **Pakdil, M. E.,** Yazırlı, B., ve Demirel, H. (2017). Spatio-temporal analyses to estimate speed information for transport forecasting and scenario testing. *Fresenius Environmental Bulletin, 26(1),* 100-106.
- **Pakdil, M.E.,** Çelik, R.N., Kaya, Ö., Konak, Y.C., ve Güney, C. (2015). Smart Aeronautical Chart Management System Design. *2015 Joint International Geoinformation Conference.* s83–87.
- Keskin, M., Çelik, B., Doğru, A. Ö., ve **Pakdil, M. E.** (2015). A Comparison of Space-time 2D and 3D Geovisualization. *27th International Cartographic Conference, Rio de Janeiro,* Brezilya. s23-28.
- Kıvılcım, C.Ö., Sterenczak, K., Kanjir, U., Şengül, A., Stavbar, G., **Pakdil, M.E.,** Lobo, E., Oo, K. S. (2012). ISPRS Student Consortium: The Network of Youth in Geoinformation Society. *XXII ISPRS Congress*.
- Kıvılcım, C. Ö., **Pakdil, M. E.,** ve Şengül, A. (2010). Role of internet as a communication platform and ISPRS Student Consortium web site. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 38,* s38-41.