T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BULUT TEKNOLOJİSİ KULLANAN HASTA TAKİP HİZMETLERİNDE MİKROSERVİS TEMELLİ UÇ SİSTEM TASARIMI VE GELİŞTİRİLMESİ

Sinan TAŞLI

Yüksek Lisans Tezi

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

TEMMUZ 2022

T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Bilgisayar Mühendisliği Anabilim Dalı

Yüksek Lisans Tezi

BULUT TEKNOLOJİSİ KULLANAN HASTA TAKİP HİZMETLERİNDE MİKROSERVİS TEMELLİ UÇ SİSTEM TASARIMI VE GELİŞTİRİLMESİ

Tez Yazarı
Sinan TAŞLI

Danışman
Dr. Öğr. Üyesi Güngör YILDIRIM

TEMMUZ 2022
ELAZIĞ

T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Bilgisayar Mühendisliği Anabilim Dalı

Yüksek Lisans Tezi

Başlığı:	Bulut Teknolojisi Kullanan Hasta Takip Hizmetlerinde Mikroservis Temelli Uç Sistem Tasarımı ve Geliştirilmesi
Yazarı:	Sinan TAŞLI
İlk Teslim Tarihi:	03.06.2022
Savunma Tarihi:	01.07.2022

TEZ ONAYI

Fırat Üniversitesi Fen Bilimleri Enstitüsü tez yazım kurallarına göre hazırlanan bu tez aşağıda imzaları bulunan jüri üyeleri tarafından değerlendirilmiş ve akademik dinleyicilere açık yapılan savunma sonucunda OYBİRLİĞİ ile kabul edilmiştir.

		İmza	
Danışman:	Dr. Öğr. Üyesi Güngör YILDIRIM Fırat Üniversitesi, Mühendislik Fakültesi		Onayladım
Başkan:	Doç. Dr. İlhan AYDIN Fırat Üniversitesi, Mühendislik Fakültesi		Onayladım
Üye:	Dr. Öğr. Üyesi Alper KILIÇ Konya Teknik Üniversitesi, Mühendislik ve Doğa Bilimleri Fakültesi		Onayladım

Bu tez, Enstitü Yönetim Kurulunun ...../...../20..... tarihli toplantısında tescillenmiştir.

İmza

Prof. Dr. Kürşat Esat ALYAMAÇ
Enstitü Müdürü

BEYAN

Fırat Üniversitesi Fen Bilimleri Enstitüsü tez yazım kurallarına uygun olarak hazırladığım "Bulut Teknolojisi Kullanan Hasta Takip Hizmetlerinde Mikroservis Temelli Uç Sistem Tasarımı ve Geliştirilmesi " Başlıklı Yüksek Lisans Tezimin içindeki bütün bilgilerin doğru olduğunu, bilgilerin üretilmesi ve sunulmasında bilimsel etik kurallarına uygun davrandığımı, kullandığım bütün kaynakları atıf yaparak belirttiğimi, maddi ve manevi desteği olan tüm kurum/kuruluş ve kişileri belirttiğimi, burada sunduğum veri ve bilgileri unvan almak amacıyla daha önce hiçbir şekilde kullanmadığımı beyan ederim.

01.07.2022

Sinan TAŞLI

ÖNSÖZ

Uzaktan hasta takip hizmetleri sağlık alanında önemli avantajları da beraberinde getirmiştir. Özellikle Nesnelerin İnterneti (Nİ/IoT) teknolojilerinin sağlamış olduğu imkânlar, hem hastaların hem de sağlık personelinin faydasına yönelik hizmetleri çok daha kolaylaştırmıştır. Öte yandan, sağlık sektörünün devasa yapısı dikkate alındığında bu tip sistemlerin yönetilebilir ve sürdürülebilir olması zorunludur. Teknolojinin zaman içerisinde getirdiği yenilikleri bu tip karmaşık sistemlere uygulamak, bu zorlukların aşılmasında farklı faydalar sunabilmektedir. Son yılların önemli IoT teknolojilerinden olan bulut hesaplama her alanda olduğu gibi sağlık alanında da büyük yeniliklere ön ayak olmaktadır. Kaynakların verimli kullanılması veya büyük sistemlerin yönetim maliyetlerinin düşürülmesinde devrim niteliğinde yenilikler getiren bulut teknolojileri, sağlık sektöründe de artık temel alt yapı bileşenlerinden biri olmuştur. Öte yandan, bulut kaynak kullanım maliyetlerinin düşürülmesi ve gereksiz bilgi aktarımının engellenmesi maliyet ve performans açısından hayatidir. Bu sorun da yine son yılların önemli alt yapı teknolojilerinden olan uçta hesaplama ve mikroservis yazılım mimarilerinin sağlamış olduğu imkânlarla çözülebilir. Bu tez çalışması, bu yeni teknolojileri beraber ve uyumlu bir şekilde kullanan bir IoT hasta takip sisteminin tasarımına odaklanmaktadır.

Bu tez çalışmasını yaparken bilgi ve tecrübelerinden faydalandığım, benden yardımlarını esirgemeyen çok değerli danışman hocam Dr. Öğr. Üyesi Güngör YILDIRIM'a ve bugünlere gelmemde çok büyük emeği olan kıymetli aileme teşekkürlerimi sunarım.

Sinan TAŞLI
ELAZIĞ, 2022

İÇİNDEKİLER

Sayfa

ÖZET

Bulut Teknolojisi Kullanan Hasta Takip Hizmetlerinde Mikroservis Temelli Uç Sistem Tasarımı ve Geliştirilmesi

Sinan TAŞLI

Yüksek Lisans Tezi

FIRAT ÜNİVERSİTESİ
Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

Temmuz 2022, Sayfa: xii + 45

Sağlık sektörünün muazzam büyüklüğü ve kendine özgü spesifik zorlukları dikkate alındığında bu sektörde kullanılacak teknolojik sistemlerin yönetilebilirliklerinin ve sürdürülebilirliklerinin ne kadar önemli olduğu bir kez daha ortaya çıkacaktır. Bu sağlık sistemlerinden biri de hem hastalara hem de sağlık personeline önemli kolaylıklar sağlayan uzaktan hasta takip sistemleridir. Son yıllardaki teknolojik gelişmeler uzaktan hasta takip sistemlerinde iyileştirmelere olanak sağlamaktadır. Özellikle bulut hesaplamanın sağladığı avantajlar ve önemli bir yazılım mimarisi olan mikroservis teknolojileri kullanılarak daha esnek hasta takip uygulamaları geliştirmek mümkündür. Bu tez çalışması, günümüzün yeni teknolojilerinin getirmiş olduğu avantajlardan faydalanan bir uzaktan hasta takip sistemi tasarımına odaklanmaktadır. En önemli özgünlüğü ise bulut sistem yükünü ve maliyetini optimize edecek uçta/kenar hesaplama alt yapısını kullanmasıdır. Ayrıca önerilen sistem üzerinde yürütülecek yazılımların yönetilebilirlikleri konteyner teknolojileri ile sağlanmaktadır. Klasik monolotik yazılım mimarilerinin uçta hesaplama için çok verimli olmamalarından dolayı mikroservis temelli yazılım mimarileri tercih edilmiştir. Bununla birlikte hastaya ve diğer yardımcı servislere ait bilgilerin dağıtık mikroservisler üzerinde nasıl tutulacağı ve yönetileceği problemi de bu tez kapsamında ele alınmaktadır. Bu tez kapsamında ilk olarak mikroservisler için ne tür bir veri yönetim modelinin kullanılabileceği incelenmiş ve aday model tipleri için gerçek zamanlı testler gerçekleştirilmiştir. Bu testlerin sonucuna göre başarılı olan model, önerilen uzaktan hasta takip sisteminde kullanılmıştır. Önerilen sistemin hayati fonksiyonları yine bir mikroservis mekanizması olan devre kesici servisler ile kontrol edilmiştir. Hem hasta tarafı hem de sağlık kurumu taraflı servisler içeren sistem, aile hekimlikleri ve onların görev kapsamında bulunan hastalara yönelik tasarlanmıştır. Performans testleri gerçek uç cihazlar kullanılarak farklı senaryolar için gerçekleştirilmiş ve sistemin performansı ispatlanmıştır.

Anahtar Kelimeler: Bulut teknolojisi, Uç sistem, Mikroservis, Hasta takip hizmetleri, Sağlık

ABSTRACT

Design and Development of Microservice Based Edge System in Patient Monitoring Services Using Cloud Technology

Sinan TAŞLI

Master's Thesis

FIRAT UNIVERSITY
Graduate School of Natural and Applied Sciences

Department of Computer Engineering

July 2022, Pages: xii + 45

Considering the enormous size and specific challenges of the health sector, the importance of the manageability and sustainability of the technological systems to be used in this sector will be seen once again. One of these health systems is remote patient monitoring systems, which provide significant convenience to both patients and health personnel. Technological developments in recent years allow improvements in remote patient monitoring systems. It is possible to develop more flexible patient monitoring applications, especially by using the advantages of cloud computing and microservice technology that is an important software architecture. This thesis focuses on the design of a remote patient monitoring system that takes advantage of these new technologies. Its most important feature is that it uses edge computing infrastructure that will optimize cloud system load and cost. In addition, the manageability of the software to be executed on the proposed system is provided by container technologies. Microservice-based software architectures have been preferred because classical monolithic software architectures are not very efficient for edge computing. In addition, the problem of how to keep and manage the data of the patient and other auxiliry services on distributed microservices is also discussed within the scope of this thesis. For this, firstly, what kind of data management model can be used for microservices has been examined and real-time tests have been carried out for candidate model types. The model, which has been successful according to the results of these tests, has been used in the proposed remote patient monitoring system. The vital functions of the proposed system are controlled by circuit breaker services, which is also another microservice mechanism. The system, which includes both patient-side and healthcare institution-side services, is designed for family medicine and patients within their scope of duty. Performance tests have been carried out for different scenarios using end devices and the applicability of the system has been proven.

Keywords: Cloud technology, Edge system, Microservice, Patient monitoring services, Health

ŞEKİLLER LİSTESİ

Sayfa

TABLOLAR LİSTESİ

Sayfa

EKLER LİSTESİ

KISALTMALAR

API	: Uygulama Programlama Arayüzü
AWS	: Amazon Web Servis
BT	: Bilişim Teknolojileri
CNCF	: Bulut Yerel Bilgi İşlem Vakfı
CPU	: Merkezi İşlem Birimi
CRUD	: Oluştur-Oku-Güncelle-Sil
EKG	: Elektrokardiyografi
ESB	: Kurumsal Veri Yolu
GPIO	: Genel Amaçlı Giriş/Çıkış
HTTP	: Hiper Metin Transfer Protokolü
IBM	: Uluslararası İş Makineleri
IoHT	: Sağlık Hizmetleri Nesnelerin İnterneti
IoT	: Nesnelerin İnterneti
JSON	: JavaScript Nesnesi Gösterimi
KKY	: Konjestif Kalp Yetmezliği
REST	: Temsili Durum Transferi
RPM	: Uzaktan Hasta İzleme
SD	: Güvenli Sayısal Hafıza Kartı
SOA	: Hizmet Odaklı Mimari
TV	: Televizyon
UCLA	: Kaliforniya Üniversitesi, Los Angeles
Vb.	: Ve benzeri
VT	: Veritabanı
WANDA	: Kan Basıncı İzleme Sistemi ile Ağırlık ve Aktivite
WSDL	: Web Servisleri Tanımlama Dili

1. GİRİŞ

İnternetin gelişimi ve günlük hayattaki kullanımının artması beraberinde birçok teknolojinin de gelişmesini sağlamaktadır. Kullanıcıların ihtiyaçlarına ve taleplerine göre yeni teknolojiler ortaya çıkabilmektedir. Burada amaç, teknolojinin yardımıyla insanların günlük hayatta yaptıkları işlerin veya ihtiyaçların daha pratik ve kolay olarak yapılmasını sağlamaktır. Nesnelerin İnterneti(IoT) ve bulut bilişimin birçok uygulama alanlarındaki kullanımının genel amacı budur. Öte yandan klasik IoT uygulamalarında büyük veri ve uzun cevap süresi problemleri ortaya çıkabilmektedir. Araştırmacılar ve mühendisler bu problemin çözümü için uç/kenar (edge) sistemler önermektedirler. Uç sistemler IoT cihazları ve bulut arasında bulunan ara hesap cihazlarından oluşmaktadır [1].

Hem IoT cihazları hem de uç sistem cihazları genellikle tek kart bilgisayar olan Raspberry Pi gibi kaynak kısıtlı cihazlardan oluşmaktadır. Bu durum yapılacak uygulamaların kısıtlanmasına neden olabilmektedir. Bu sebepten dolayı uç sistemler oluşturulurken sistemin performansının ve verimliliğinin yüksek olması için kullanılan teknolojilere dikkat edilmelidir. Kaynak kısıtlı cihazlar üzerinde monolitik mimarinin [2] kullanımı çoğunlukla performans sorunlarına yol açabilmektedir. Monolitik mimarinin yerine kaynak kısıtlı cihazlarda mikrosevis mimarisinin [3] tercih edilmesi uç sistemin performansı ve kullanılabilirliği açısından daha esnek çözümler sunmaktadır. Uç sistemler üzerinde çalışan mikroservislerin dağıtımı ve kontrolü konteyner teknolojisiyle[4] gerçekleştirilebilir. Konteyner teknolojisi çalışmak için yüksek donanıma ihtiyaç duymaz. Bu durum konteyner teknolojisinin kaynak kısıtlı cihazlar üzerinde kullanımının performans yönünden daha elverişli olduğunu göstermektedir.

IoT ve bulut bilişim günümüzde akıllı şehir, akıllı ulaşım, endüstriyel ve sağlık uygulamalarında yaygın olarak kullanılmaktadır [5-7,47]. Son yapılan araştırmalar sağlık-IoT cihazlarının toplam IoT cihazlarının %40'ını oluşturduğunu göstermektedir [8,19]. Bu oranın gelecekte önemli oranda artacağı öngörülmektedir. Ülkelerde, kronik hastalıklar nedeniyle yüksek sağlık bakım giderleri ve toplumdaki bireylerin üretkenliğinin düşük olması gibi problemler yaşanabilmektedir. Diyabet hastalığı bakımı için ayrılan bütçe, ulusal sağlık bütçelerinin %15'i kadarını oluşturabilmektedir [9]. Sağlık alanındaki giderleri düşürmenin ve insanların üretkenliğini iyileştirmenin en iyi yollarından biri, kronik hastalıkların azaltılması ve bu tür hastaların izlenmesi vasıtasıyla uygun sağlık faaliyetlerinin uygulanmasıdır. E-sağlığın tüm avantajlarından yararlanmak için gelişmiş yöntemlerden ve modern teknolojilerden yararlanmak çok önemlidir [9]. Yeni teknolojik araçlar, sağlık alanının giderlerini azaltmak, hasta bireylerin güçlendirilmesini artırmak, semptom ve hastalık bulgularının devamlı olarak gözden geçirilmesi vasıtasıyla hastaların takibinin geliştirilmesi ve kronik hasta bireylerin azaltılması amacıyla en iyi şekilde kullanılmalıdır.

Teletıp, sağlık hizmetlerinin uzaktan gerçekleştirilmesi ve evde bakım hizmetleri için önemli bir araçtır. Sağlık bilgilerine hızlı ve zamanında erişim tıbbi hataların azaltılması ve sağlık çalışanları arasındaki koordinasyonun artırılması, uzak bölgelerdeki kentsel sağlık merkezlerinde hastaların seyahat ve fiziksel varlığının azaltılması bu teknolojinin sağlık alanındaki avantajlarından bazılarıdır [10-12]. Sağlık alanında IoT'nin kullanımı genellikle hasta takip hizmetleri üzerinedir. Özellikle kritik durumlu hastaların izlenmesi önem arz etmektedir. Bu tip hastaların izlenmesi için mobil tabanlı hasta takip sistemleri veya uç sistemlerden oluşan hasta takip sistemleri kullanmak mümkündür [13-15]. Hasta takip sistemleri, özellikle kritik durumlu hastaların izlenmesi ve olası riskli durumlarda erken müdahale ile hastanın hayatını olumsuz yönde etkileyecek durumların önlenmesi amacıyla kullanılmaktadır.

Sağlık, teknolojiyle birlikte hızla gelişen bir alandır. Sağlık sektöründeki yeni gelişmelerden biri, artan sağlık sorunları ile hızla yaşlanmakta olan dünya nüfusu problemlerine karşı birçok avantaja sahip olan hastaların uzaktan izlenmesine dayalı hasta izleme sistemleridir. Hastaların sağlık kuruluşlarında izlenmesi için daha basit uygulamaların tercih edilmesi yeterli olmaktadır. Teknoloji, modern iletişim ve sensör teknolojilerinin kullanımı ile hastanın evde günlük aktivitelerini gerçekleştirirken dahi izlenilmesine olanak tanıyacak kadar gelişmiştir. Elektrokardiyogram okuması, kalp atış hızı, solunum hızı, kan basıncı, sıcaklık, kan şekeri seviyeleri ve sinir sistemi aktivitesi gibi temel yaşamsal belirtileri izlemek için sensörler günümüzde mevcuttur. Uzaktan sağlık hizmeti yelpazesi, kronik hastaları, yaşlıları, prematüre çocukları ve kaza mağdurlarına kadar geniş bir kullanım alanına sahiptir. Yeni teknolojiler sayesinde, hastalar, hastalığa veya duruma göre izlenebilir. Teknoloji, vücuda takılı sensörlerden çevreye bağlı ortam sensörlerine kadar çeşitlilik gösterir ve yeni çalışmalar, yalnızca hastanın sensörden birkaç metre uzakta olmasını gerektiren temassız izlemeyi amaçlar. Düşme tespit sistemleri ve kronik hastaları izlemeye yönelik uygulamalar hâlihazırda kullanılmaktadır [16,17]. Sağlık alanında kullanılan uzaktan takip sistemleri ile ilgili yapılan literatür araştırmasında daha önce yapılan çalışmaların bazıları şu şekildedir. Lakmini ve arkadaşları [18] uygulamalarında, hem etkileşimli hem de etkileşimsiz uygulamalarla uzaktan sağlık hizmeti ve hasta bireylerin izlenmesi alanındaki son çalışmaların bir incelemesini vermektedir.

Dante ve arkadaşları [19], sağlık verilerini işleyebilen uç-sis hesaplama temelli mikroservis sistemlerinin uygulanabilirliğini ispatlamışlardır. EKG sinyallerinin izlenebildiği örnek uygulamaları ile büyük veri ön analizlerini uç sistemler üzerinde gerçekleştirmişlerdir. Çalışma uç ve sis sistemleri beraber kullanmaktadır. Bu da çok katmanlı bir analize olanak sağlamıştır. Katmanlar arası yürütüm ise mikroservisler aracılığı ile yapılmıştır. Sonuçta büyük veri analiz verimi %60'lara yakın oranda arttırılmıştır.

E-Sağlık uygulamalarının tasarlanması ve geliştirilmesine ilişkin mevcut ve ortaya çıkan yaklaşımlar, sağlık hizmeti sunumunun iyileştirilmesini sürdürme potansiyellerini giderek daha

fazla kanıtlamaktadır. IoT ve mikro hizmetler, çok çeşitli sağlık hizmeti uygulamalarının kalitesini artırmak için gerekli hale gelmektedir. Marilena ve arkadaşları [20] çalışmalarında, yaşlı insanların sağlıkla ilgili bilgilerinin uygun bir şekilde yönetimi için geliştirilen RO-SmartAgeing sistemini göstermektedir. Mimari, Sağlık Hizmetleri Nesnelerinin İnterneti (IoHT) ve bulut bilişim teknolojilerinin birlikte kullanımına dayanır ve çok katmanlı bir tıbbi karar desteği sağlar. Bu mimarinin, öngörülen kullanım durumlarında gösterildiği gibi, gelecekteki sisteme çeşitli uygulama gereksinimlerine yanıt verme yeteneği sağlaması beklenmektedir.

Uç sistem ve mikro hizmet içeren bir hücresel ağ, mobil sağlık uygulamalarını her zaman ve her yerde desteklemek için kullanılabilmektedir. Uç sistem, bir ağın hesaplama yeteneklerini potansiyel olarak artırabilir ve mikro hizmet, ağın çeşitli mobil sağlık uygulamalarına uyum sağlamasına yardımcı olabilir. Di ve arkadaşı [21] çalışmalarında böyle bir ağın, mimarisinin tasarımını, kaynaklarının tahsisini ve verilerinin yönetimini sunmaktadırlar. Spesifik olarak, mobil sağlık uygulamaları için iletişim, hesaplama ve hizmet açısından üç katmanlı bir ağ mimarisini ele almaktadırlar. Ayrıca, performansını optimize etmek için ağ içindeki kaynak tahsisi konularını tartışmaktadırlar. Son olarak, tıbbi verilerin yönetimini ele almaktadırlar.

Jorge ve arkadaşları [22] çalışmalarında, diyabet, kalp ve tansiyon gibi kronik hastalığı olan hastalara sağlık ve egzersiz rutini önerilerini izleyebilen bir ontolojiye dayalı mimari geliştirmişlerdir. Sahar ve arkadaşları [23], tip 2 diyabette glikosile edilmiş hemoglobini kontrol etmek için olağan bakıma kıyasla Uzaktan Hasta İzleme (RPM) sistemlerinin bir değerlendirmesini yapmaktadırlar. P. Varady ve arkadaşları [24] çalışmalarında, hasta izlemeye yönelik bir yaklaşım tanıtmaktadırlar. Bu yaklaşımla standart donanım araçları ve yazılım teknolojilerinin kullanımıyla oluşturulmuş mevcut bir endüstri standardı iletişim ağına dayalı olarak bir hasta takip uygulaması yapılmıştır. Bu hasta takip uygulaması sayesinde, açık mimari sistem modellemesi, ölçeklenebilirlik, standart arayüzler ve esnek sinyal yorumlama imkânları sunulmaktadır.

Mohammad Salah ve arkadaşları [25] çalışmalarında, sensörlere dayalı bağlı ağlar aracılığıyla hastaların sağlık durumunu otomatik olarak izlemek için akıllı bir hasta izleme sistemi önermişlerdir. Hastanın biyolojik davranışlarını toplamak için çeşitli sensörler kullanılmaktadır. Sensörlerden elde edilen biyolojik bilgilerden önemli olanları daha sonra IoT bulutuna gönderilmektedir. Sistem, sensör verilerini işleyerek bir hastanın kritik durumunu tespit edebilen ve doktorlara/hemşirelere ve hastane sorumlu personeline anında bildirim sağlayan bir hasta izleme sistemidir. Doktorlar ve hemşireler, ilgili hastalarını şahsen ziyaret etmeden uzaktan gözlemleyerek bu sistemden faydalanabilmektedir. Ayrıca hasta yakınları da bu sistemden sınırlı erişimle faydalanabilmektedir.

Kronik hastalıklar, insanlar için önde gelen ölüm nedenlerindendir. Konjestif Kalp Yetmezliği (KKY), bu tür hastalıklara örnek olarak verilebilir. Myunk-kyunk ve arkadaşları [26] çalışmalarında, KKY ile ilgili sorunların yaygınlığı nedeniyle, kalp hastalığının günlük olarak

önlenmesini, izlenmesini ve tedavisini kolaylaştıracak bir hasta izleme sistemi gerçekleştirmektedirler. Bu hasta izleme sistemi WANDA'yı (Kan Basıncı İzleme Sistemi ile Ağırlık ve Aktivite); KKY olan hastaların sağlıkla ilgili ölçümlerini izlemek için sensör teknolojilerinden ve kablosuz iletişimden yararlanan bir çalışmadır. WANDA uygulaması, sensörler, web hizmetleri ve arka planda çalışan veritabanlarından oluşan üç katmana sahip bir mimaridir. Sistem, UCLA (Kaliforniya Üniversitesi, Los Angeles) Hemşirelik Okulu ve UCLA Kablosuz Sağlık Enstitüsü ile birlikte, KKY ile ilişkili dekompansasyonu gösteren önemli klinik semptomların erken tespitini sağlamak için geliştirilmiştir. Başka bir çalışmada R. Kumar ve arkadaşları [14], hastanın vücut ısısı, solunum hızı, kalp atışı ve vücut hareketinin izlenmesine yönelik hasta takip sistemi yapmaktadırlar.

Teletıp sistemi, hasta ve doktor tarafında bulunan donanım ve yazılım bileşenlerinden oluşur. Teletıp uygulaması için önde gelen bir alan, EKG'nin teşhis için ana araç olduğu kardiyoloji alanıdır. Sherin ve arkadaşları [27] çalışmalarında, görüntü yakalama, bilgi çıkarma ve MATLAB araçları kullanılarak gerçekleştirilen analiz ve internet ağına dayalı veri gönderme sistemi için dijital kamera aracılığıyla sürekli bir EKG sinyali akışını elde etmek ve analiz etmek için görüntü tabanlı teknikler sağlamaktadırlar. Tercih edilen bu yöntemde, bir web kamerası kullanılarak yoğun bakım takip cihazından vital bulgu değerleri yakalanmakta ve bulut aracılığıyla gerekli yerlere iletilmektedir. Buluttan gelen görüntüler daha sonra bir cep telefonu vasıtasıyla hastanın doktoruna ulaşmaktadır. Hastada meydana gelebilecek riskli durumların oluşması halinde doktorun cep telefonuna uyarı mesajı iletilir. Bu çalışma sayesinde, hastanın kalp atış hızı takip edilerek hastanın durumunun izlenmesi, değerlendirilmesi ve gerek duyulması halinde bildirim oluşturmak için bir sistem önerilmekte ve elde edilen görüntünün doktora ulaşması amaçlanmaktadır.

Bu tez çalışmasında özellikle kritik durumlu hastalar için bulut teknolojisi kullanan hasta takip hizmetlerinde mikroservis temelli uç sistemlerin tasarımı ve geliştirilmesi üzerine çalışmalar yapılmıştır. Tez çalışmasının 2. bölümünde kullanılan teknolojiler ve tasarım mimarileri ile ilgili genel bilgiler verilmiştir. 3. bölüm yani Materyal ve Metot bölümünde yapılan çalışmalar gösterilmiştir. 4. bölümde çalışmalar sonunda elde edilen bulgular ve tartışmaya yer verilmiş, 5. bölümde de sonuçlar ve öneriler paylaşılmıştır.

2. KULLANILAN TEKNOLOJİLER VE TASARIM MİMARİLERİ

Bu tez çalışmasında bulut teknolojisi kullanan hasta takip hizmetlerinde mikroservis temelli uç sistemlerin tasarımı ve geliştirilmesi için çalışmalar yapılmıştır. Bu çalışmalar gerçekleştirilirken Nesnelerin İnterneti (IoT), uç sistem, bulut bilişim, docker teknolojileri ve mikroservis mimarisi kullanılmıştır.

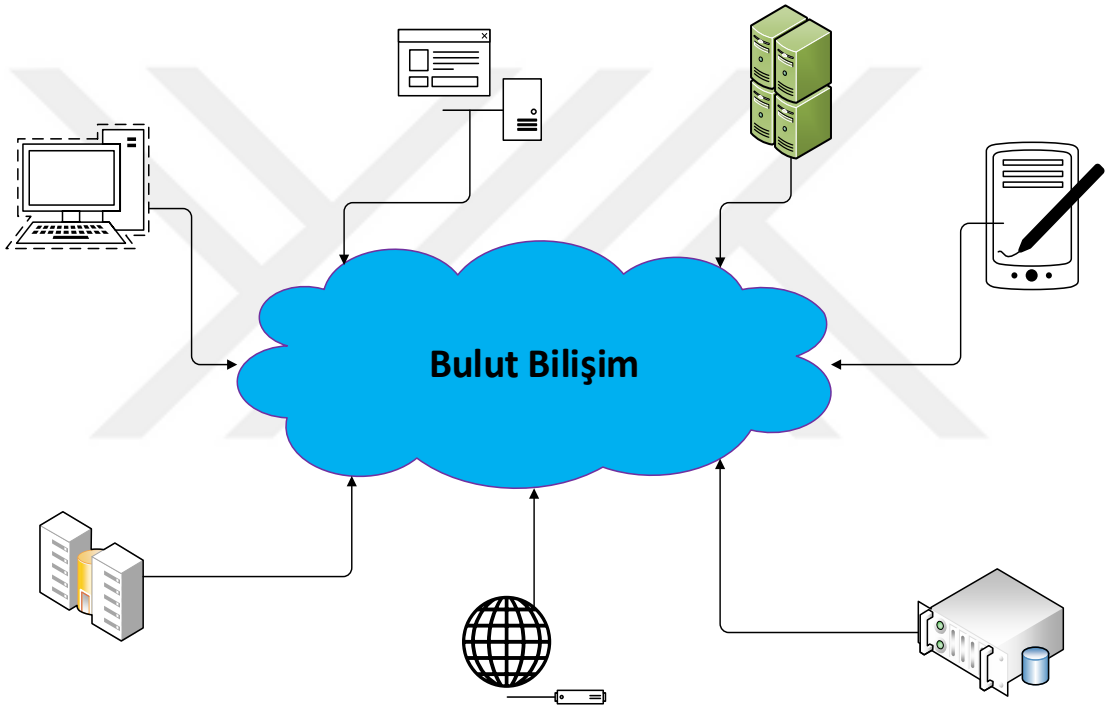
2.1. Nesnelerin İnterneti (IoT)

Nesnelerin İnterneti (IoT), dünya çapında çeşitli fiziksel cihazları internet aracılığıyla birbirine bağlamayı amaçlar. IoT ifadesi ilk olarak 1999 yılında Kevin Ashton'ın bir şirket için yaptığı sunumda kullanmasıyla ortaya çıkmıştır [28]. IoT, internet vasıtasıyla diğer nesneler ve sistemlerle iletişim kurmak ve bu tür cihazlarla veri alışverişi gerçekleştirmek için sensörler, yazılımlar ve kullanılan diğer teknolojilerle fiziksel nesnelerin ağını tanımlar. IoT cihazları, günlük hayatta kullanılan ev eşyaları, endüstriyel uygulamalarda kullanılan araçlar ve bu tür amaçlarda kullanılan tüm nesneler olabilir [29]. IoT günümüzde, akıllı şehir, akıllı ulaşım, endüstriyel ve sağlık uygulamaları başta olmak üzere birçok alanda kullanılmaktadır. IoT cihazları, kendi aralarında insan müdahalesi olmaksızın iletişim sağlayabilir. IoT'nin geliştirilmesi, altyapı, iletişim, arayüzler, protokoller ve standartlar gibi birçok konuyu içerir [28].

2.2. Bulut Bilişim

Bulut teknolojisi, veri depolama, sunucular, veri tabanları, ağ oluşturma ve yazılım gibi araçları ve uygulamaları internet yardımıyla kullanıcılara sunmaktadır. Bulut teknolojisi denilince akla ilk olarak veri depolama gelmektedir. Fakat bulut teknolojisi yalnızca veri depolama hizmeti sunmamaktadır. Veri depolama sayısız hizmetlerden sadece biridir. Bulut teknolojisi, bilgisayarlar ve diğer elektronik cihazlar için, gerek duyulduğunda kullanılabilen ve kullanıcılar arasında paylaşılan bilgisayar kaynakları sağlayan, internet tabanlı bilişim hizmetlerinin tümünü kapsamaktadır. Bilgisayarların veya IoT cihazların internete ulaşımı sağlandığı sürece, buluttaki verilere ve gerekli yazılım programlarına erişimi vardır. Bulut bilişim, mevcut sunuculardan doğan maliyet artışını azaltmak, kolay erişim, kullanım kolaylığı, hız ve verimlilik, performans ve güvenlik gibi çeşitli sebeplerden dolayı kullanıcılar tarafından çokça tercih edilen popüler bir araçtır. Bulut hizmeti sağlayan birçok şirket bulunmaktadır. Bunlardan bazıları Amazon, IBM, Microsoft ve Google'dır. Bulut bilişim sağlayıcıları, kullanıcılara üç temel modelde hizmet sağlamaktadırlar. Bunlar, altyapı hizmeti (IaaS), yazılım hizmeti (SaaS) ve platform hizmeti (PaaS)'dir. IaaS'de sunucular fiziksel veya sanal makineler olarak sunulur. En temel bulut hizmeti modelidir. PaaS'de kullanıcılara hizmet olarak platform sunulur. Kullanıcılar bu platformlar

üzerinde test ve uygulama geliştirme gibi işlemlerini gerçekleştirebilmektedirler. SaaS'de kullanıcılara yazılım hizmeti sunulur. Kullanıcılar bu hizmetlerin arayüzüne erişerek gerekli işlemlerini yapabilmektedir. Bulut hizmeti sayesinde kullanıcıların dosyalarını ve uygulamalarını yerelde depolamak yerine daha güvenli olan uzak sunucularda saklanması ve bu sayede verilere internet yardımıyla ulaşmak oldukça kolaydır. Bu durum, insanların erişim sağlamak için belirli bir lokasyonda olmasının gerek duyulmadığı ve kullanıcının uzaktan çalışmasına fırsat tanındığı anlamına gelmektedir. Şekil 2.1'de görüldüğü gibi bulut bilişim kullanıcılara birçok farklı alanda hizmet sağlayabilmektedir.



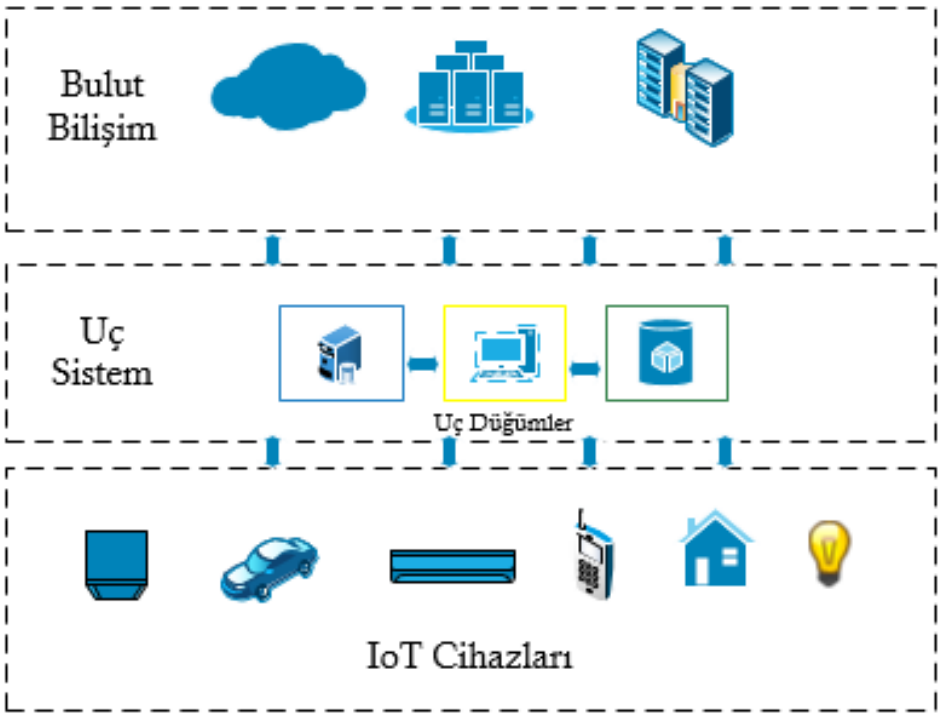
Şekil 2. 1. Bulut bilişim

Bulut bilişim sayesinde, verilerin saklanması ve analiz edilmesi gibi tüm ağır işleri, insanların yanında taşıdıkları veya üzerinde işlem yaptıkları cihazlardan uzaklaştırmak mümkündür[30]. Bu sayede kullanıcıların işlerini ve uygulamalarını dünyanın herhangi bir yerinden ve internete bağlanabilen herhangi bir cihaz tarafından yürütmeleri ve kullanmaları mümkündür. Bulut hizmetlerinin farklı kullanım türleri bulunmaktadır. Genel kullanıma ait bulut hizmetleri, gerçekleştirilen hizmetleri kullanıcılardan aldığı ücret karşılığında internet tabanlı gerçekleştirir. Özel kullanılan bulut hizmetleri ise sadece kullanıcıların belirlediği sınırlı sayıdaki kişilere hizmet sunmaktadır. Hem genel kullanıma hem de özel kullanıma hitap eden hibrit sistem

de bulunmaktadır. Bulut bilişim verileri yerel ortamdan bağımsız olarak sakladığı ve dünya üzerinde her hangi bir ortamdan internet üzerinden verilere çok kolay bir şekilde ulaşıldığı için endüstriyel alanlardaki kullanımı oldukça yaygındır. Özellikle uç sistemin endüstriyel alandaki öneminin artmasından dolayı bulut bilişim de son yıllarda önemini giderek artırmaktadır. Uç sistemlerde genelde verilerin yerelden bağımsız bir ortama aktarılması istendiğinden uç sistem ve bulut bilişim endüstriyel uygulamalarda beraber kullanılmaktadır.

2.3. Uç (Kenar) Sistem

Günümüz internet teknolojilerinin bir diğer önemli konusu her geçen gün daha da artan bulut bilişim yüküdür [31]. İnternet kullanıcılarının ve IoT aygıt sayılarının hızla artması doğal olarak bulut servislerde de ciddi bir baskının oluşmasına yol açmaktadır. Bu baskının kırılmasında son yılların önemli çözümlerinden olan uç/kenar (edge) sistemler [32] büyük faydalar sağlamıştır. Uç sistem, son kullanıcı veya otonom sisteme yakın bir konumda yer alır ve hangi hizmetlerin lokal (edge) servisler hangilerinin bulut servislerden alınacağına karar verir. Bu özelliği ile bulut sağlayıcılar için verimli bir alan tahsisi, son kullanıcılar açısından ise zaman ve maliyet kazancı sağlar. Uç sistemlerin hesaplama kaynakları bulut sistemlere göre kısıtlıdır. Bu nedenle bulut veya diğer uç sistemlerle olan iletişim ve servis alışverişleri için mikroservisleri kullanmaları daha uygundur. Özellikle son yıllarda konteyner teknolojisinin gelişmesi ile uç sistemlerde mikroservis temelli uygulama kullanımı daha da artmıştır. Son kullanıcı sistemde toplanan verilerin bir kısmı uç sistem kaynaklar üzerinde tutulurken bir kısmı bulut kaynaklar üzerinde tutulur veya işlenir. Her ne kadar mikroservislerin ayrı veri tabanı kullanmaları çok istenmese de doğası gereği uç sistemlerde çalışan mikroservisler kendilerine ait veri tabanları bulundurmak zorunda kalabilirler. Örnek olarak bir fabrikanın farklı bölümlerinde sensör verilerini takip eden uç bilgisayarlarda çalışan mikroservisler veya bir hasta takibi yapan gömülü uç aygıtlar üzerinde çalışan mikroservisler kendilerine ait veri tabanları barındırabilirler. Bu durumda hem uç sistem içinde hem de bulut servisler ile veri tutarlılığının sağlanması zorunludur. Veri tutarlılığı ile birlikte verimli kaynak kullanımı ve hızlı cevap süresi dikkatli bir veri paylaşımı mekanizması gerektirmektedir. Uç sistem yapısı Şekil 2.2'den daha kolay anlaşılabilir.



Şekil 2. 2. Uç sistem

Uç sistemler, son kullanıcıya yakın bir konumdadır ve bulut servisleri ile lokal sistem arasında filtreleme, veri toplama, güvenlik, hesaplama ve depolama gibi görevleri icra ederler. Bu özellikleri ile son kullanıcıya, özellikle hız ve bulut servis maliyetlerini düşürme konusunda önemli faydalar sağlarlar. Uç cihazlar nispeten kısıtlı donanımsal kaynaklara sahiptir. Bu cihazlar, fiziksel sunucular olabileceği gibi tek kart bilgisayarlardan oluşan gömülü sistemler de olabilir. Son kullanıcı sistemin fiziksel şartlarına bağlı olsa da genellikle lokal sistemde birden fazla uç düğüm bulunabilmektedir (Örneğin akıllı şehirler, sağlık uygulamaları vb.). Her bir uç cihaz kendi alt bölgesinden veri toplayabilir, işleyebilir ve depolayabilir. Sürekli artan son kullanıcı IoT cihaz sayısı da dikkate alındığında birden fazla uç düğüm bulunduran lokal sistemlerde veri tutarlılığının ve güvenilirliğinin hayati öneme sahip olacağı açıktır.

2.4. Monolitik ve Mikroservis Mimarisi

Yazılım tasarım mimarisi denilince hiç kuşkusuz monolitik mimari [33] ve SOA akla ilk gelenler arasındadır. Monolit uygulamalar tek bir proses içerisinde farklı yazılım komponentleri içerirler. Tek bir platformda geliştirilirler ve bu nedenle platform bağımlı kütüphaneler kullanırlar. Genel anlamda tek bir yapıya sahip oldukları için iç komponentleri arasında bir ağ iletişimine veya Uygulama Programlama Arayüzü (API) kullanımına ihtiyaçları yoktur. Aynı şekilde mesaj aktarımına ihtiyaç duymazlar ve proses içi bellek içi veritabanı kullanarak yüksek performans sağlayabilirler. Ancak öte yandan monolit uygulamaların doğası gereği önemli bazı dezavantajları

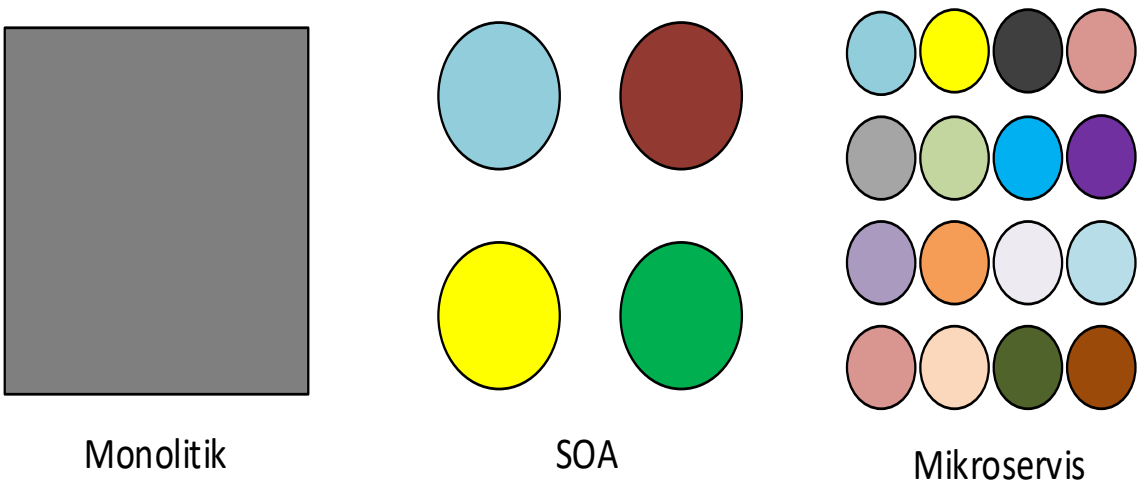
da bulunmaktadır. Monolit bir uygulamadaki bir komponentin çökmesi tüm uygulamayı olumsuz yönde etkileyebilir. Platform bağımlı olması ve yeniden kullanılabilir komponent geliştirmenin limitli olması geliştirme ekibi koordinasyonu ve sonraki uygulama güncellemeleri için manevra yeteneğini kısıtlar. Karmaşık uygulamalarda, geliştirme ekibini zorlayabilir. Proses temelli çalışmalarından dolayı ihtiyaç fazlası sistem kaynakları tüketebilirler. Ayrıca farklı monolit uygulamalar arasında proses-proses iletişimi karmaşık ve maliyetlidir. Avantaj ve dezavantajları dikkate alındığında, monolit uygulamaları pek çok yerde görebiliriz (Örneğin, SaaS uygulamalar, .NET uygulamalar, single java WAR file gibi).

IoT ve bulut teknolojilerinin getirmiş olduğu imkânlar, dağıtık uygulamaların hızlı bir şekilde yayılmasına yardımcı olmuştur. Özellikle son yıllarda akıllı şehir, akıllı enerji, akıllı ulaşım ve sağlık sistemler bünyelerinde birçok dağıtık uygulamayı barındırmaktadır [34]. Bu uygulamalar tek bir projenin bileşenleri olabileceği gibi bir birlerinin sağladıkları servislere ihtiyaç duyan otonom varlıklarda olabilirler. Her iki durumda da ortaya çıkacak iletişim, sürdürülebilirlik ve beraber çalışabilirlik problemleri Servis Odaklı Mimari (SOA) temelli çözümlere ihtiyaç duymaktadır. Yaklaşık yirmi yıl önce ortaya çıkan SOA [35] yaklaşımı bu alanda önemli bir köşe taşı olmayı başarmıştır. Klasik SOA çözümler, uzun yıllar büyük ölçekli projelerde başarı ile uygulanmıştır. Öte yandan, klasik SOA çözümler genellikle maliyetlidir ve nispeten yavaştır. Bu nedenle araştırmacılar ve mühendisler, eski bir Unix görüşü olan Mikroservis [36] temelli çözümleri önermektedirler. Bir yazılım geliştirme teknolojisi olan mikroservisler aslında SOA'nın bir alt tipidir. Temel prensibi böl ve yönet olan bu teknolojide karmaşık büyük sistemler gevşek bağlı küçük hizmetlere ayrılır. Bu servisler, HTTP, JSON, REST gibi yaygın internet teknolojilerini kullanarak modüler dağıtık sistemler tasarlanmasına olanak sağlarlar.

İnternet ve ağ teknolojilerinin gelişmesiyle uygulamalar birbirleri ile daha hızlı ve verimli bir şekilde etkileşime geçebilmiştir. Bu da karmaşık bir uygulamanın bir birleri ile iletişime geçebilen dağıtık küçük modüllere ayrılabilmesine imkân sağlamıştır. Bu SOA mimarinin başlıca özelliğidir. SOA mimaride, platform bağımsız (polygot) yeniden kullanılabilir komponent geliştirmek daha kolaydır. Bu da beraberinde yatay ölçekleme ve servis paylaşımı yapabilen heterojen uygulamalar geliştirmeye yardımcı olmuştur. SOA servisleri uç noktalara sahiptir ve servisler internet ortamında deklere edilebilir. Klasik SOA çözümlerde servislere Enterprise Service Bus (ESB) aracılığı ile erişilir [37]. ESB'ler, gerekli yönlendirme, kimlik doğrulama ve protokol dönüşümlerini gerçekleştirerek ana iletişim omurgasını oluştururlar. SOAP protokolü ve WSDL gibi önemli teknolojiler sayesinde uzun yıllar SOA uygulamaları başarıyla kullanılmıştır ve kullanılmaya da devam edilmektedir. Ancak klasik SOA sistemler de kendine özgü dezavantajlara sahiptir. Örneğin, klasik SOA çözümler ESB'den dolayı daha yavaştır ve kurulum maliyeti de yüksektir. Buna ek olarak ESB'lerin geliştirilmesi pahalıdır ve genellikle uzman bir ekip gerektirmektedir. Bunu da genellikle büyük teknoloji şirketleri (IBM, Amazon vb.)

sağlayabilmektedir. Ayrıca klasik SOA uygulamalarda her servis için her zaman yazılım araçları bulunmayabilir.

Mikroservisler, SOA'nın bir alt tipidir ve monolitik mimari ile klasik SOA'nın eksiklerini tamamlamaya yönelik yaklaşım sunar. Klasik SOA'nın tüm avantajlarına sahip olan mikroservis mimaride ESB'ye ihtiyaç olmaması kurulum ve sürdürüm maliyetini oldukça düşürür. Ayrıca platformdan bağımsız bir teknoloji olmasından dolayı esnek ve geniş bir geliştirme ortamı seçeneğine sahiptir. Mikroservisler farklı lokasyon ve aygıtlarda farklı proseslerde çalışabilir. Mikroservisler, REST API [38], GraphQL [39] ve gRPC [40] gibi farklı API teknolojileri kullanarak iletişime geçebilir ve tek bir sistem gibi davranabilir. Mikroservis temelli uygulamalar, servis keşfine ve otomatik dağıtıma izin verir. Mikroservisler, bire bir senkron, bire bir asenkron ve olay odaklı (event-driven) teknikleri ile birbirleri ile iletişim kurabilirler. Bunla birlikte mikroservisler arası iletişimde ağ iletişim maliyeti ve hata olasılığı daha yüksektir. Ayrıca her bir mikroservis ayrı bir proses olarak çalıştığı için bağlam maliyetleri yüksek olabilmektedir. Mikroservisler, küçük ve modüler yapılarından dolayı uç sistemlerde rahatlıkla kullanılabilir. Hafif konteyner [41] içinde çalışabilen mikroservisler uç cihazlar üzerinde hızlı kurulum ve kolay yönetim imkânına sahip olabilirler. Uç sistemler üzerinde çalışan mikroservisler, son kullanıcı için bulut maliyetini, gereksiz veri transferini ve gecikme süresini ciddi oranda düşürebilir. Bulut hizmeti şirketler tarafından sağlanmaktadır. Bu hizmetin karşılığında kullanıcılardan ücret alınır. IoT'lerde sensörlerden alınan tüm verilerin direkt olarak buluta aktarılması zamanla bulutta ayrılan yerin tükenmesine yol açar. Bunun sonucunda ek alana ihtiyaç duyulur. Bu da kullanıcılar için ek maliyet demektir. Uç sistemlerin mikroservislerle beraber kullanımı bu maliyeti azaltabilmektedir. Tüm bu mimarilere ait genel mekanizma yapısı Şekil 2.3'deki gibidir.



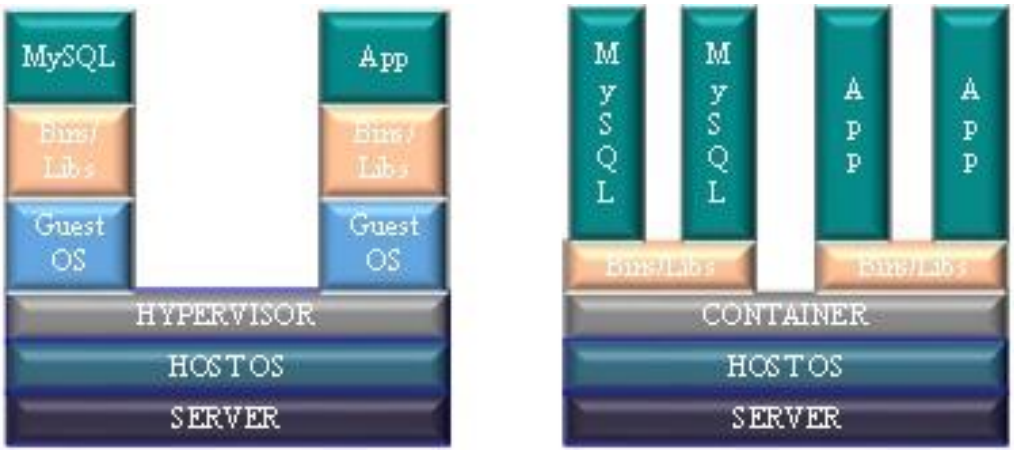
Şekil 2. 3. Monolitik, SOA ve mikroservis mimarisi

2.5. Konteyner Teknolojileri

Konteyner teknolojisi, uygulamaları çalıştırmak için kullanılan bir paketleme yöntemidir. Bu teknoloji sayesinde herhangi bir işletim sisteminde çalışan bir uygulamanın başka bir cihazda ve işletim sistemlerinde çalışması mümkündür. Konteyner teknolojisi 1979 yılında ortaya çıkmıştır. Fakat 2000'li yılların başına kadar istenen ilgiyi görmemiştir. Docker'in piyasaya sürülmesiyle birlikte konteyner teknolojisi de hızlıca gelişmeye başlamıştır. Sanallaştırma teknolojisinin bilişim dünyası üzerinde önemli bir etkisi bulunmaktadır. Teknolojinin gelişimi ve bilişim dünyasının sanallaştırma teknolojisi üzerindeki beklentisi nedeniyle yeni sanallaştırma teknolojileri ortaya çıkmaktadır. Bu sanallaştırma teknolojilerinden biri olan docker konteyner teknolojisi 2010'lı yıllarda yayımlandı. Yayımlandığı ilk yıllarda pek ilgi görmemesine rağmen 2013 yılından günümüze kadar ki süreçte en çok tercih edilen sanallaştırma teknolojilerinden birisi olmayı başarmıştır. Bunda docker konteyner teknolojisinin bilişim dünyasının isteklerine ve gelişen teknolojiye ayak uydurmasının payı oldukça büyüktür. Docker konteyner açık kaynak kodludur ve günümüzde de halen gelişimini sürdürmektedir.

Docker konteyner diğer sanallaştırma teknolojilerine göre oldukça hızlıdır. Çalışmak için çok fazla kaynak gereksinimine ihtiyaç duymaz. Bu özelliği sayesinde uç hesaplama (edge computing) ve sis hesaplama (fog computing) gibi yerlerde tercih edilmektedir. Docker konteyner sanallaştırma teknolojisi donanımı en optimum şekilde kullanmayı sağlar. Günümüzün sık kullanılan sanallaştırma teknolojilerinden hipervizör sanallaştırma teknolojisinde istenmeyen bir atıl kapasite oluşabilmektedir. Bu durum, kaynak israfını ve maliyetini artırmaktadır. Docker konteyner sanallaştırma teknolojisi, sanallaştırma işlemi yaparken donanımı en verimli şekilde kullanarak oluşacak atıl kapasiteyi büyük ölçüde azaltmaktadır.

Docker teknolojisinin avantajı, sanal makine kaynak kullanım senaryolarında daha net bir şekilde görülebilir. Hipervizör sanallaştırma teknolojisi kullanılarak bir bilgisayarda birden fazla sanal makine kurulmak istenirse bu durumda oluşturulacak her sanal makine için ayrı bir işletim sistemi kurmak gerekir. Oluşturulacak her sanal makine için ayrı bir işletim sistemi kurulması sanal makinelerin kullanılmasını ve bakımını güçleştirir. Docker konteyner sanallaştırma teknolojisi, oluşturulan sanal makinelerin tek bir işletim sistemi ile kontrol edilmesine olanak sağlamaktadır. Bu sayede oluşturulan her sanal makine için ayrı ayrı işletim sistemi kurmaya gerek kalmaz. Kullanılan bu yaklaşımlar Şekil 2.4 ile daha iyi anlaşılabilir.



Şekil 2. 4 Hipervizör ve konteyner yapısı

Hipervizör sanallaştırma teknolojisinde bir uygulama için bile işletim sistemi kurulur ve sanal makine kurulurken donanımın belirli bir kısmı oluşturulan sanal makine için ayrılır. Kurulan sanal makine kendisine ayrılan donanımın hepsini kullanmasa dahi boşta kalan donanım revize olduğu için diğer sanal makineler tarafından kullanılmaz. Bu durum kullanıcılar tarafından istenmeyen bir atıl kapasiteye neden olur. Ayrıca Şekil 2.4'ten de görüleceği üzere hipervizör sanallaştırma teknolojisi ile sanal makine kurulurken her sanal makine için ayrı ayrı kütüphane dosyaları yüklenmelidir. Bunun asıl nedeni her sanal makinenin farklı işletim sistemi kullanmasıdır. Bu durumda her işletim sisteminin kendine ait kütüphane dosyaları yüklenmesi gerekir. Bu işlem hem docker konteynere göre daha uzun sürer hem de kullanımı ve bakımı docker konteynere göre daha zordur.

Docker konteyner sanallaştırma teknolojisi, hipervizör sanallaştırma teknolojisinde karşılaşılan ve istenmeyen durumları önemli ölçüde ortadan kaldırmaktadır. Docker konteyner sanallaştırma teknolojisiyle oluşturulan sanal makineleri ortak bir işletim sistemi üzerinden kullanmak mümkündür. Bu durum kurulan her sanal makine için ayrı bir işletim sistemi kurulmasının önüne geçmektedir. Kullanıcılar bu şekilde sanal makinelerinin kullanımını ve bakımını daha rahat bir şekilde yapabilmektedirler. Docker konteyner sanallaştırma teknolojisiyle kurulan sanal makineler ortak bir işletim sistemi üzerinden çalışabildiği için kütüphane dosyalarını da ortak kullanabilmektedir. Docker konteyner sanallaştırma teknolojisi kullanmak için öncelikle bir docker imaj dosyasına ihtiyaç vardır. Bu imaj dosyası internet üzerinden paylaşılabilir ve özelleştirilebilir. Uygulama geliştirme giderleri ve süresi, geleneksel sanal makine yerine docker konteynerin kullanımıyla büyük ölçüde azaltılabilir. Ayrıca Docker kullanımı ile bulut platformunu yeniden geliştirme maliyeti büyük ölçüde azaltılabilir [42]. Docker kullanarak konteyner oluşturma için imaj dosyası kullanılır. İmaj dosyalarını oluşturmak için bazı komutlar kullanılmalıdır. Bu komutlardan en çok kullanılanları Tablo 2.1'de gösterilmektedir.

Tablo 2. 1. Docker imaj için kullanılan komutlar ve açıklamaları

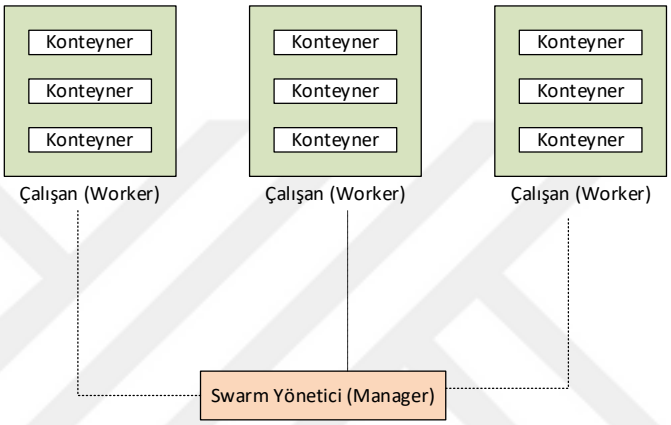
Komut	Açıklama
FROM	Yeni bir imaj oluşturma aşamasını başlatır ve sonraki talimatlar için temel imajı ayarlar.
WORKDIR	Konteyner için çalışma dizinidir.
COPY	Dosyaları ana bilgisayardaki konumdan imaja kopyalar.
ADD	Dosyaları ana bilgisayardaki bir konumdan imaja kopyalar. ADD komutu ayrıca bir URL'den kopyalamayı ve bir tar dosyasının içeriğini imaja çıkarmayı da sağlar.
RUN	Bir komutu yürütür ve sonuçları imaja yeni bir katman olarak kaydeder.
LABEL	Konteyner hakkındaki meta verileri depolamak için bir anahtar / değer çiftidir.
BUILD	Build komutuna iletilecek bir değişken tanımlar.
CMD	Konteyner içerisinde belirli bir komutu yürütmek için kullanılır.
ENV	Ortam değişkenlerini ayarlar.
EXPOSE	Konteyner ve dış dünya arasında ağ oluşturmaya olanak sağlamak için belirli bir bağlantı noktasını ilişkilendirir.
VOLUME	Konteynerden ana makinedeki bir dizine erişimi etkinleştirmek için kullanılır.
USERS	Konteyneri çalıştıracak kullanıcı adını ayarlar.

2.5.1. Konteyner Yönetim Sistemleri

Docker konteyner sanallaştırma teknolojisi kullanarak yapılan uygulamaları geniş ölçekte dağıtırken, tüm mimari bileşenleri mevcut ve gelecekteki stratejileri göz önünde bulundurarak planlamak ve koordine etmek gerekir. Konteyner yönetim araçları, birden çok kümedeki konteynerlerin yönetimini kolay bir şekilde gerçekleştirmeye yardımcı olur. Konteyner yönetim araçları sayesinde BT geliştiricileri ve yöneticileri, sadece tek bir sanal sistem ile konteyner kümesini kolayca oluşturabilir ve yönetebilir. Kümeleme, docker konteyner için önemli bir etkendir ve yöneticilerin bir sistem grubu oluşturmasına imkân sağlar [43]. En çok tercih edilen konteyner yönetim araçlarından ikisi Kubernetes ve Docker Swarm'dır.

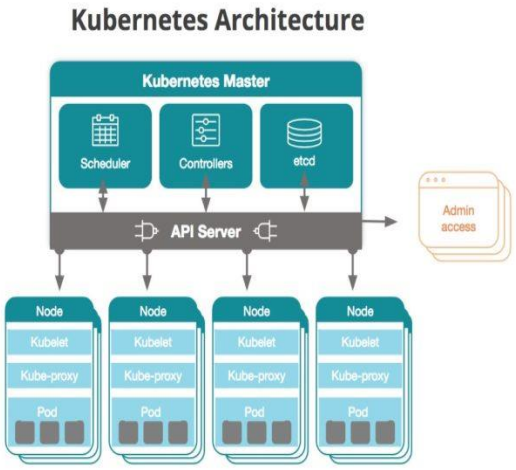
Docker Swarm, konteynerleri dağıtmak ve kontrolünü sağlamak için kullanılan bir konteyner yönetim aracıdır. Docker Swarm konteynerleri sürekli olarak izler ve herhangi bir konteynerin zarar görmesi halinde çalışan konteynerler arasında yük devretme yapabilir. Docker Swarm, yöneticilere ve yazılım geliştiricilere, uygulamada değişen şartlar doğrultusunda yeni bir konteyner ekleme veya çıkarma işlemlerini yapmalarına olanak tanır. Docker Swarm, manager (yönetici) ve worker (çalışan) düğümlerinden oluşur. Manager düğümü, konteynerlerin yürütülmesi ve kontrol edilmesi işlemlerini yapar. Manager düğüm sayısı bir veya birden fazla olabilmektedir. Tek manager düğümlü sistemler test işlemleri için tercih edilebilir. Fakat gerçek uygulamalar üzerinde en az üç manager düğümünün olması önerilir. Manager düğüm sayısının birden fazla olması halinde düğümler arasında bir lider düğüm seçilir. Lider düğümün zarar görmesi durumunda diğer manager

düğümlerden birisi lider düğüm olur ve bu sayede sistemin çalışması devam eder. Sistemdeki manager düğüm sayısının (n-1)/2 kadarı zarar görse dahi sistem çalışmasını sürdürebilir. Örnek olarak üç manager düğüme sahip bir sistemde bir manager düğüm zarar görse bile sistem çalışmasını sürdürebilir. Manager düğüm olmayan diğer tüm düğümler worker düğümlerdir. Bu düğümlerde uygulama konteynerleri çalışır. Manager düğümler ayrıca worker düğüm olarak ta çalıştırılabilir. Fakat bu durum sistemin güvenliği için önerilmez. Şekil 2.5'te Docker Swarm yapısı gösterilmektedir.



Şekil 2. 5 Docker swarm

Google bünyesinde 2014 yılında ortaya çıkan Kubernetes, konteyner dağıtımını ve kontrolünü etkili bir şekilde yönetmeye yarayan açık kaynaklı bir diğer konteyner yönetim aracıdır. 2015 yılında Google, Bulut Yerel Bilgi İşlem Vakfı'nı (CNCF) kurmak için Linux Vakfı ile de ortaklık kurdu. Kubernetes projesinin yönetimi CNCF tarafından gerçekleştirilmektedir. Kubernetes mimarisi, baştan sona yönetim göz önünde bulundurularak tasarlanmıştır. Gelen talimatları çalışan konteynerlere dağıtan bir ana düğümün bulunduğu birincil çoğaltma modeline sahiptir. Çalışan konteynerler, mikro hizmetlerin üzerinde çalıştırıldığı makinelerdir. Bu tasarım, her bir düğümün bir konteyner çalışma süresi içinde çalışan tüm konteynerleri barındırmasına imkan tanır [43]. Şekil 2.6'da Kubernetes mimarisi gösterilmektedir.



Şekil 2. 6. Kubernetes[44]

Düğümler ayrıca kubelet içermektedir. Kubelet, her düğümün kendine ait karar verme ve yönetim mekanizması olarak düşünülebilir. Kubeletler, ana düğümdeki API'den aldıkları talimatların işlenmesini sağlar. Buna ek olarak, bir konteyner arızalanırsa arızalanan konteyner yerine yenisini oluşturmak da dahil olmak üzere bölmeleri yönetir. Hem Kubernetes hem de Docker Swarm konteynerli uygulamaları akıllıca yönetmek için kullanılan kapsamlı fiili yönetim çözümlerdir. Benzer yetenekler sunsalar bile, farklı temellere sahip oldukları ve farklı problemleri giderdikleri için ikisi doğrudan karşılaştırılabilir değildir. Docker Swarm, kullanıcıların hızlı bir kuruluma ihtiyaç duydukları ve çok geniş yapılandırma isteklerine sahip olmadıkları durumlar için iyi bir tercihtir. Mikro hizmet temelli mimariye sahip yazılım ve uygulamaları verimli bir şekilde sunar. Kubernnetes'e göre avantajı, uygulama gerçekleştirilirken kurulumun kolaylığı ve daha kolay bir öğrenme eğrisine sahip olmasıdır. Docker Swarm, uygulamalarda yazılım geliştirme ve test işlemleri için oldukça kullanışlıdır. Bu sebepten dolayı, konteynerlerin kolay dağıtımı ve otomatik yapılandırması aşamalarında, Kubernetes'e göre daha az donanım ihtiyacı gerektirir ve uygulama geliştirirken dikkate alınması gereken ilk konteyner yönetim aracı olabilir. Fakat Docker Swarm'ın dezavantajları da vardır. Bunlar, yerel izleme olanaklarının eksik olması ve Docker API'sının yetkisini sınırlamasıdır. Ancak buna rağmen yine de yer paylaşımlı ağ iletişimi, yük dengeleme, yüksek kullanılabilirlik ve çeşitli ölçeklenebilirlik imkanları sağlar. Docker Swarm, konteynere alınmış bir uygulama oluşturmak ve onu gecikmeden çalıştırmak isteyen kullanıcılar için oldukça ideal bir çözümdür. Kubernetes, geliştirilmekte olan uygulama karmaşıksa ve bu süreçte yüz binlerce konteyner kullanılıyorsa kullanılacak en iyi konteynerleştirme platformudur. Yüksek kullanılabilirlik özelliklerine ve kolay ölçeklendirme imkânlarına sahiptir. Fakat Kubernetes, Docker Swarm'a göre öğrenilmesi ve uygulanması daha zordur. Uygulamayı oluşturmak ve ayağa kaldırmak uzun sürmektedir. Kubernetes, kullanıcıların uygulamalarını özelleştirmek istemeleri ve çok geniş işlemlere ihtiyaç duydukları durumlar için ideal bir çözümdür.

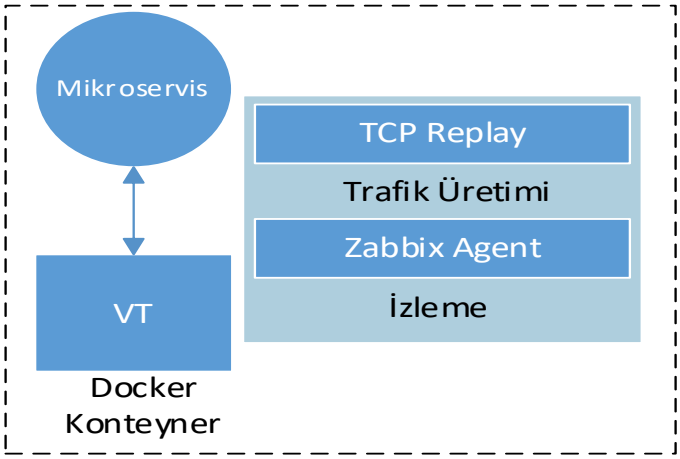
3. MATERYAL VE METOT

Bu tez çalışmasında önerilen sistem iki farklı safhada gerçekleştirilmiştir. Önerilen sistemde bazı mikroservisler kendilerine ait veri tabanlarına sahiptir. Dağıtık ve bağımsız veri tabanlarının olduğu sistemlerde veri yönetimi ve sürdürülebilirliği oldukça önemlidir. Bu sebeple birinci safhada farklı veri yönetimi mekanizmaları test edilerek bu tip uç sistemler için uygun olanın belirlenmesi amaçlanmıştır. Bu safhada farklı senaryolar için yapılan test sonuçlarına göre uygun görülen mekanizma sistem yapısı içinde kullanılmıştır. İkinci safhada ise aile hekimlikleri ve sorumluluk alanlarına yönelik üç alt üniteden oluşan sistem yapısı oluşturulmuştur. Bu alt üniteler; hasta uç ünitesi, aile hekimliği uç ünitesi ve bulut ünitesidir. Son olarak önerilen sistem farklı senaryolar için performans testi sürecinden geçirilmiştir. Yapılan test sonuçları önerilen sistemin başarısını ispatlamıştır.

3.1. Mikroservis Tabanlı Uç Sistemler İçin Veri Depolama ve Yönetim Modellerinin Performans Analizi

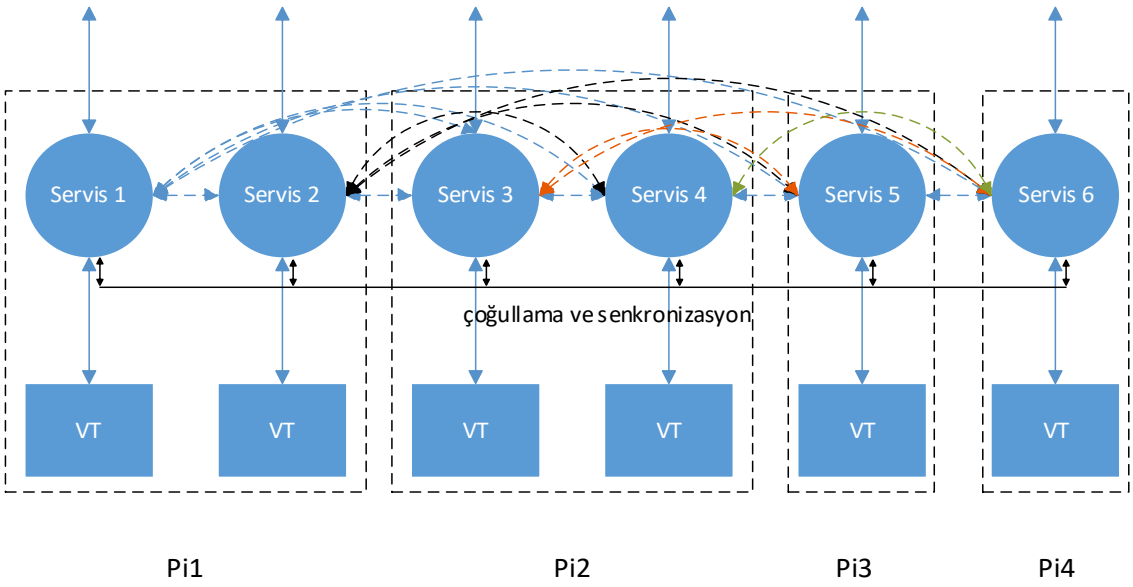
Bulut servislere gereksiz veri akışını azaltmak, kullanıcılar için maliyet ve sistem gecikmesini de azaltır. Veri trafiğinin optimize edilmesinde uç sistem önemli faydalar sağlar. Öte yandan uç sistemlerde çok büyük (heavyweight) uygulamalar fazla tercih edilmez. Sağladığı avantajlarla mikroservisler özellikle kısıtlı kaynaklı uç sistemler için uygun (lightweight) çözümler sunmaktadır. Bulut ile son kullanıcı sistem arasındaki veri trafiğinin kontrolünde uç sistemin kullandığı veri depolama ve yönetim mekanizması doğru seçilmelidir. Veri güvenilirliğinin sağlanması ve uç sistem performansı kullanılacak olan modele bağlıdır. Bu çalışmanın ilk safhasında, mikroservis tabanlı uç sistemler için üç farklı veri depolama ve yönetim modeli incelenmiştir. Bunlar, mikroservislerin kendilerine ait veritabanlarının olduğu veya veri tabanının dağıtık olarak yönetildiği modellerdir. Tüm modeller, dağıtık uç cihazlarda bulunan mikroservislerin senkron ve asenkron çalışma yöntemleri için test edilmiştir. Çalışma sonunda özellikle kısıtlı kaynaklı uç sistemler için uygun modelin bulunması hedeflenmiştir.

Genel olarak mikroservislerin kendilerine ait veritabanlarına sahip olmaları pek tercih edilmez ancak mikroservis kullanan uç sistemlerde bu kaçınılmaz bir hale gelebilir. Bu sebeple uç sistem özelliklerine uygun, performans ve kaynak maliyetini gözeten bir veri depolama ve yönetim sistemi gereklidir. Çalışmada üç farklı veri yönetim modeli, altı mikroservis çalıştıran dört farklı uç düğüm üzerinde denenmiştir. Kullanılan modeller hem senkron hem de asenkron çalışma modları için ayrı ayrı denenmiştir. Uç düğüm olarak Raspberry Pi 4 kullanılmıştır. İletişim HTTP ve Restful API'ler aracılığı ile gerçekleştirilmiştir. Uygulama örneği olarak hasta bilgilerinin işlendiği bir sağlık bakımı (healthcare) uç sistem tercih edilmiştir. Uç cihaz yapısı Şekil 3.1'de gösterilmiştir.



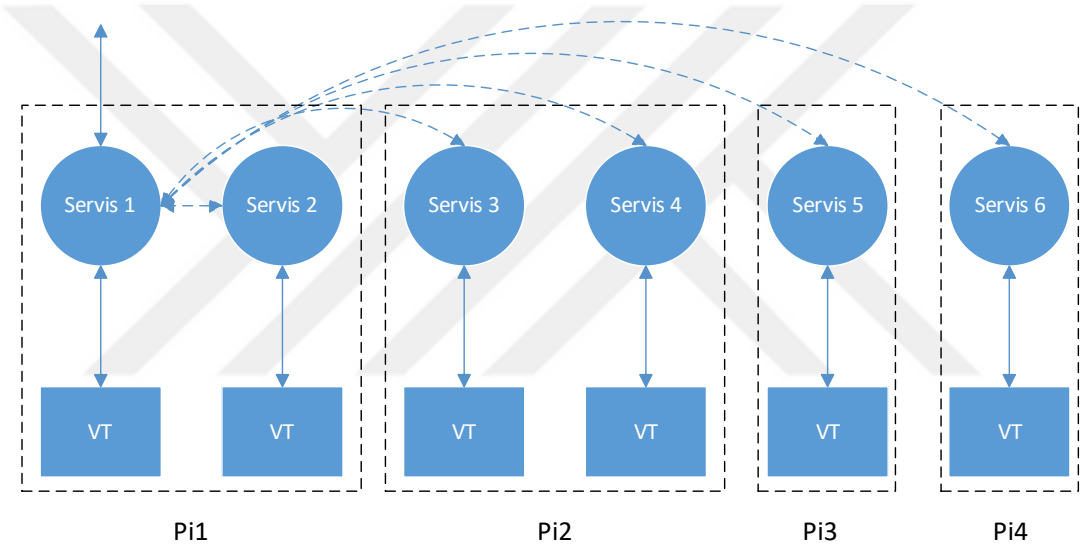
Şekil 3. 1. Uç cihaz yapısı

Şekil 3.2'de Model 1'in yapısı gösterilmektedir. Model 1 her bir mikroservisin kendi veri tabanına sahip ve her birinin ayrı istek (request) alabildiği senkron modeldir. Bu model genellikle donanımsal olarak riskli çalışma durumunun olduğu senaryolarda kullanılmaktadır. Bu modelde veri bütünlüğü kopya veri bulundurma (dublication) ile sağlanmaktadır. Bir mikroservise ait veri tabanında meydana gelen değişiklik diğer tüm mikroservislere bildirilir. Böylece tüm mikroservisler aynı bilgilere sahip olurlar. Deneylerde veritabanı işlemleri için farklı senaryolar oluşturulmuştur. Gelen select işlemleri veritabanlarında bir değişikliğe neden olmaz bu nedenle bu işlem temel performans ölçümü için analiz edilmiş, insert/delete/update işlemleri ise veri tutarlılığı kontrolü için gerçekleştirilmiştir



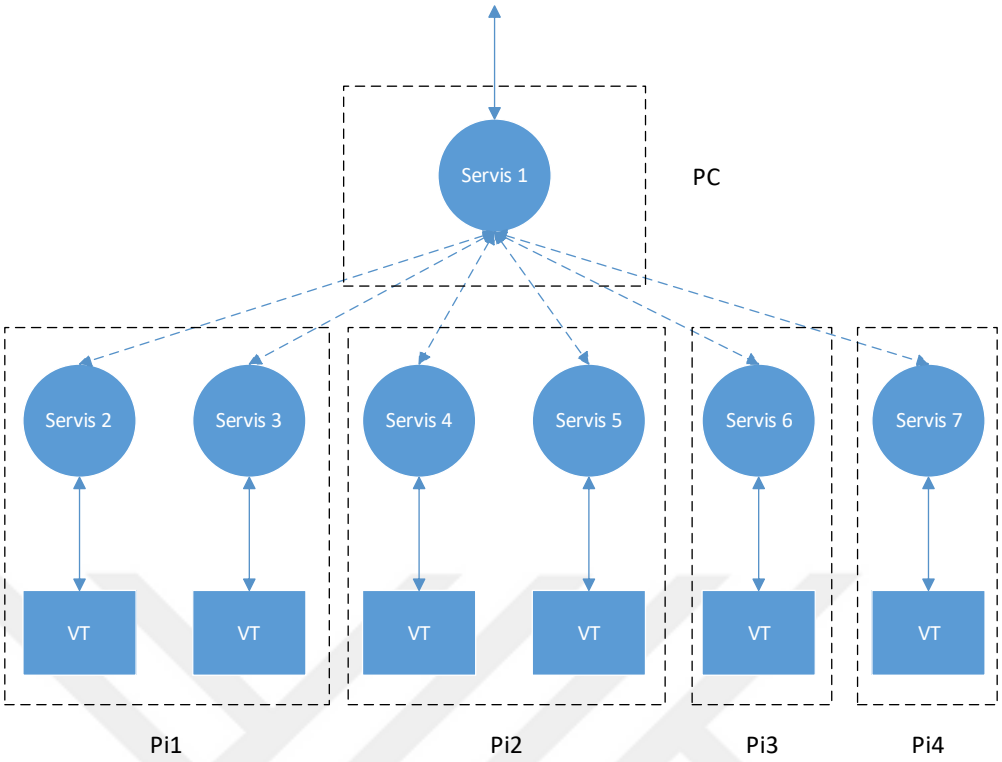
Şekil 3. 2. Model 1 (senkron mod)

İkinci model Şekil 3.3'te gösterilmektedir. Bu modelde hastaya ait farklı bilgiler farklı mikroservislerdeki ayrı veritabanlarında tutulmaktadır. Örneğin bir mikroservis hasta özlük bilgilerini tutarken, bir diğer mikroservis randevu bilgilerini tutmaktadır. Bu modelde son kullanıcı erişimi bir mikroservis üzerinden gerçekleştirilmektedir. Bu mikroservis gelen sorguları uygun şekilde dağıtır ve gelen cevapları birleştirir. Genellikle karmaşık verilerin işlendiği modüler senaryolarda kullanılan bu modelde güvenilirlik kırılgan olabilmektedir. Tüm modellerde uç cihaz, mikroservisleri ve veritabanlarını konteynerler içerisinde çalıştırmaktadır. Ayrıca uç cihaz, iletişim trafiği üretimi ve performans ölçümleri için Tcpreplay [45] ve Zabbix Agent [46] uygulamalarını içermektedir.



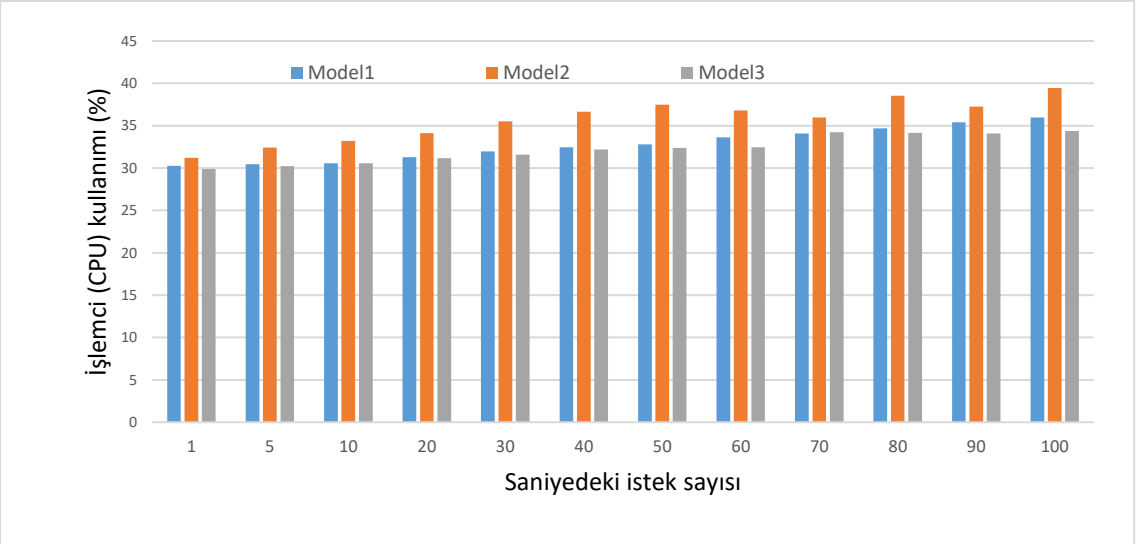
Şekil 3. 3. Model 2 (senkron mod)

Son model Şekil 3.4'te gösterilmektedir ve Model 2'ye benzer olmakla beraber fazladan bir mikroservise sahiptir. Bu mikroservis bir veritabanı hizmeti sunmaz. Görevi son kullanıcı isteklerine göre sorgulama yapma ve dönen cevapları birleştirmedir.

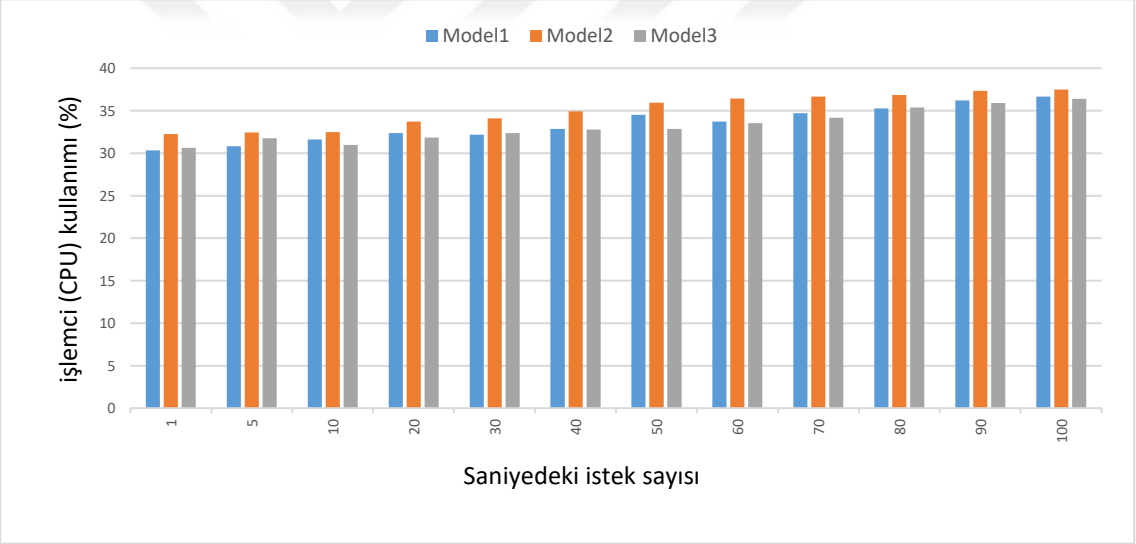


Şekil 3. 4. Model 3 (senkron mod)

Bu üç model ayrı ayrı çalıştırılarak uç düğüm performansları izlenmiştir. Farklı sayıda CRUD (Create-Read-Update-Delete) istekleri için performansı izlenen metrikler, uç cihaz üzerindeki ortalama işlemci (CPU) kullanımı, hafıza kullanımı ve ortalama cevap süresidir. Her bir model için select, insert, delete ve update işlemlerine ait uç düğüm işlemci (CPU) kulanım oranları Şekil 3.5 ve Şekil 3.6'da verilmiştir.

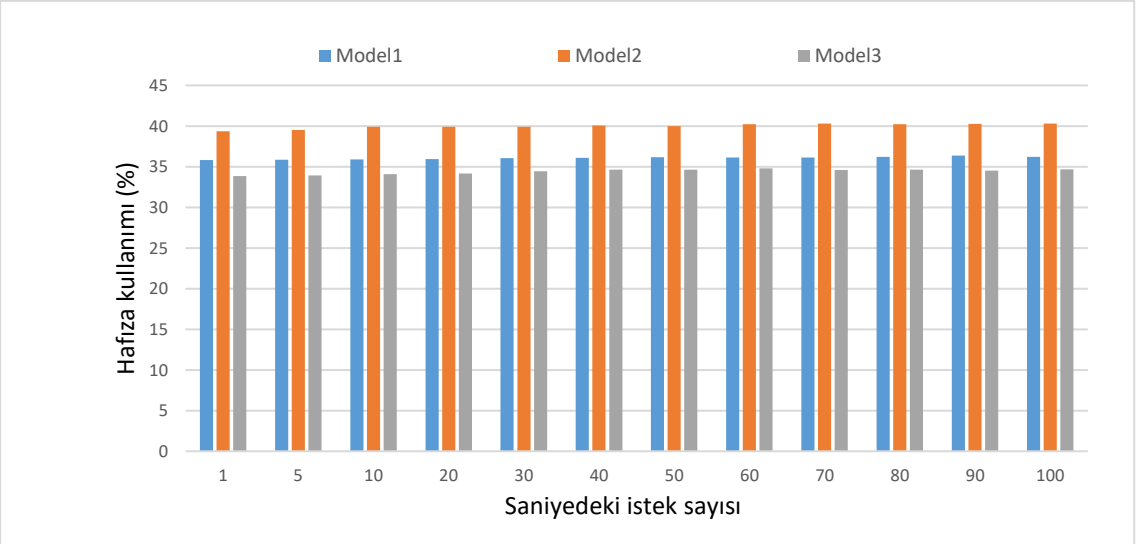


Şekil 3. 5. Select işlemi için işlemci (CPU) kullanımı (senkron mod)

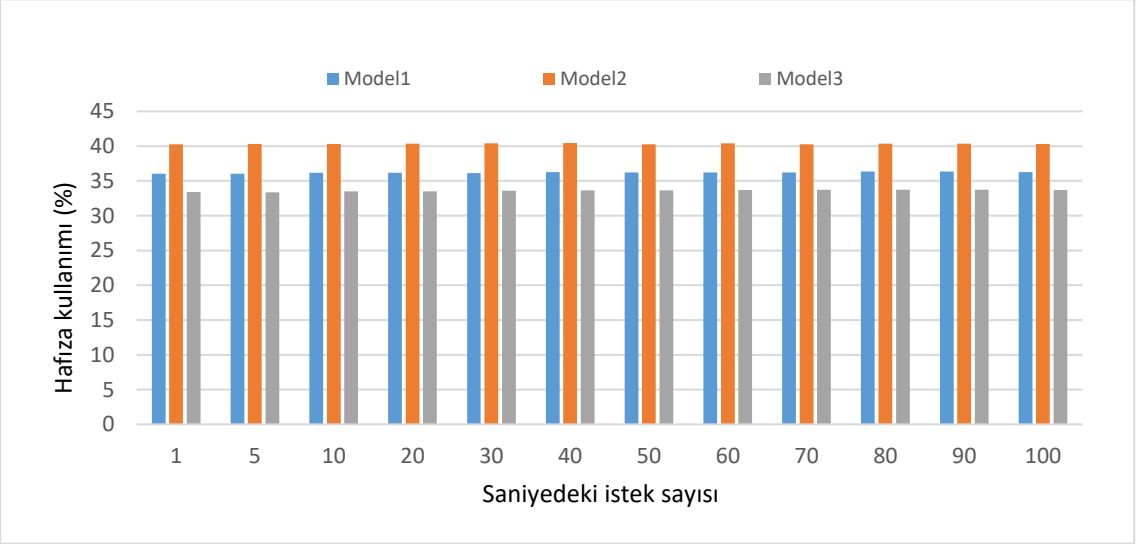


Şekil 3. 6. Insert, delete ve update işlemleri için işlemci (CPU) kullanımı (senkron mod)

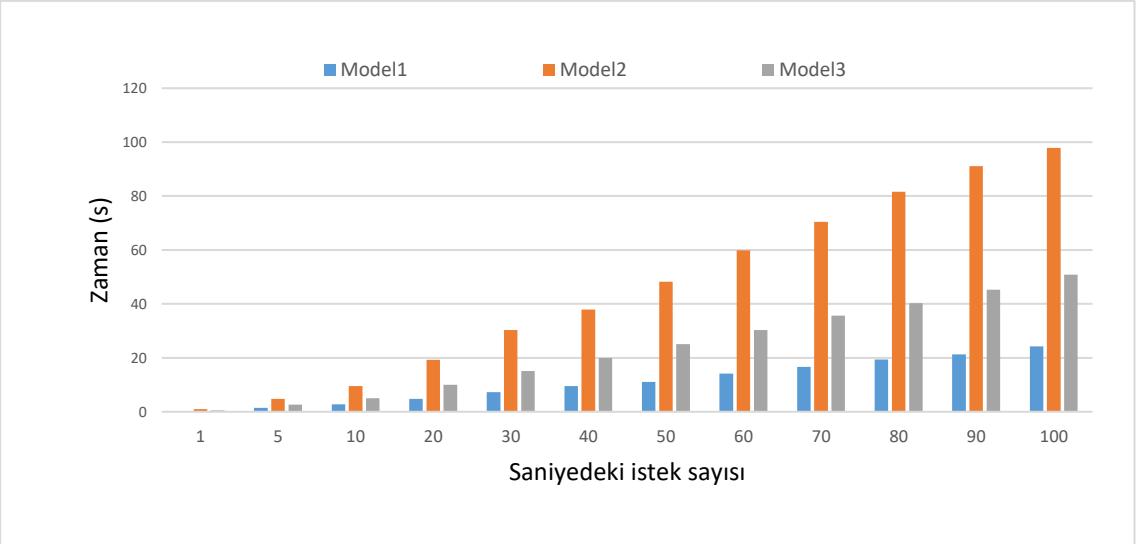
Select işleminde model 1'in işlemci kullanım oranı model 2'den daha iyidir. Insert, delete ve update işlemlerinde durum farklıdır. Bunun sebebi model 1'de sadece select isteğini alan mikroservisin işlem yapmasıdır. Bu durum hafıza kullanımı ve cevap süreleri değerlerinde de görülmektedir. Ölçümler yapılırken tüm uç cihazlardan alınan değerlerin aritmetik ortalaması alınarak hesaplanmıştır. Şekil 3.7 ve Şekil 3.8'de select, insert, delete ve update işlemlerine ait uç cihazların hafıza kullanım oranları gösterilirken Şekil 3.9 ve Şekil 3.10'da da isteklerin cevaplanma süreleri verilmiştir.



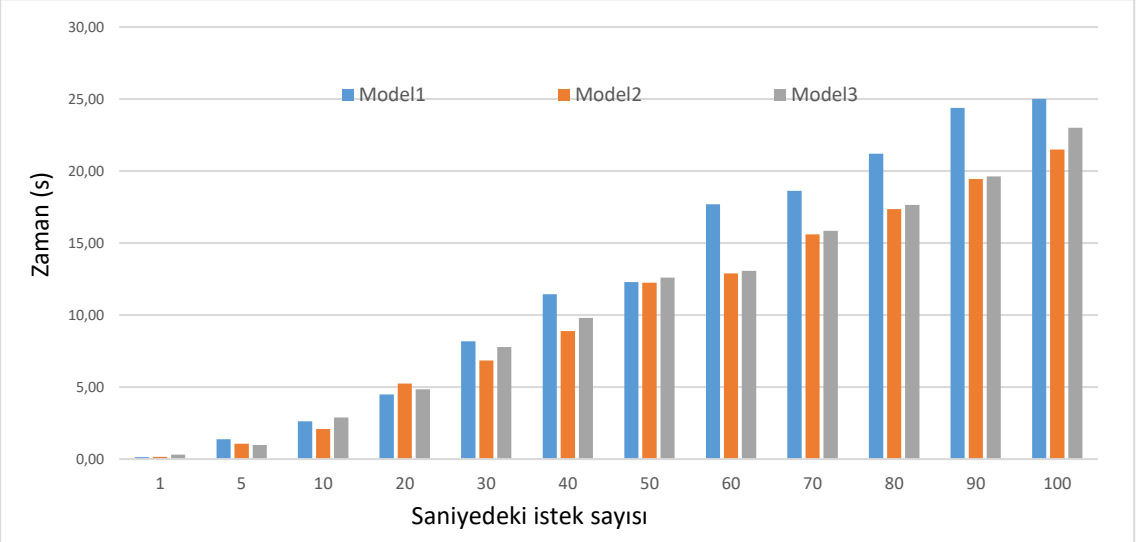
Şekil 3. 7. Select işlemi için hafıza kullanımı (senkron mod)



Şekil 3. 8. Insert, delete ve update işlemleri için hafıza kullanımı (senkron mod)



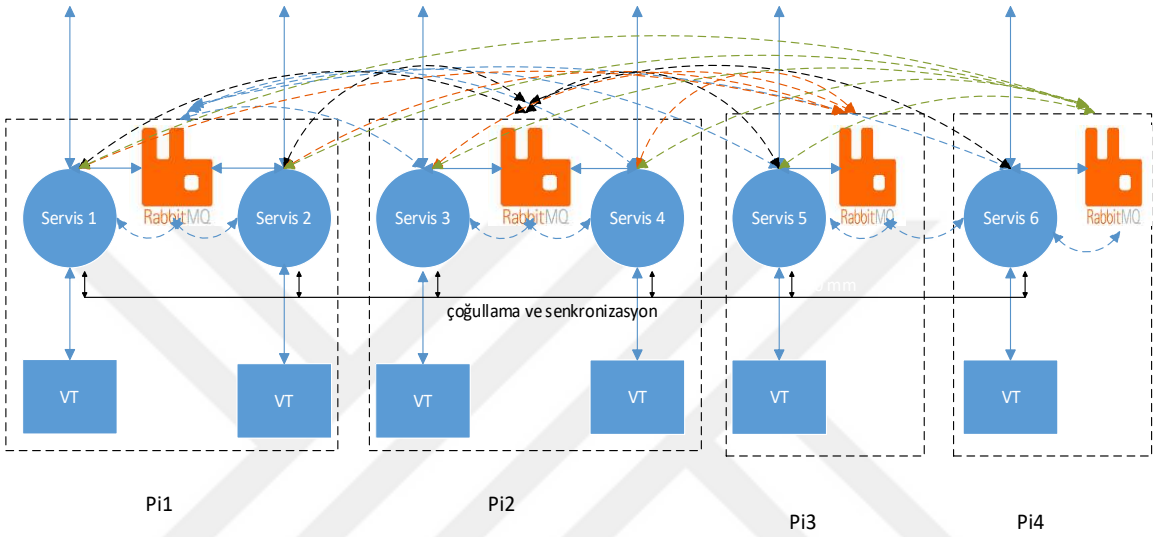
Şekil 3. 9. Select işlemi için cevap süreleri (senkron mod)



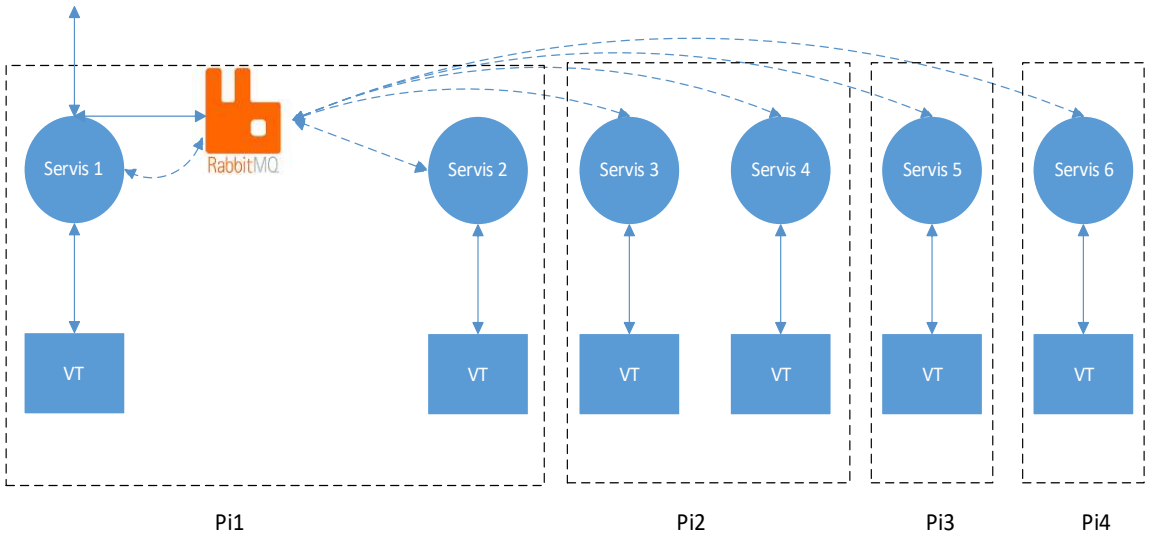
Şekil 3. 10. Insert, delete ve update işlemleri için cevap süreleri (senkron mod)

Modellere yapılan istekler sonucunda işlemci ve hafıza kullanım oranlarında Model 3'ün en iyi sonucu verdiği gözlenmiştir. Select işleminin cevaplanma süresi değerlerinde en iyi sonucu Model 1 vermektedir. Fakat insert, delete ve update işlemlerindeki cevaplanma süresi değerleri dikkate alınırsa Model 2 ve Model 3'ün Model 1'e göre daha iyi olduğu görülmektedir. Ölçümlerdeki istekler tek kullanıcı tarafından gönderilmiştir. Senkron olarak çalışan bu modellerde kullanıcı sayısının birden fazla olduğu durumlarda mikroservislerin bazen isteklere cevap veremediği görülmüştür. Senkron çalışan modeller üzerinde yapılan ölçümler sonucunda, genel anlamda kaynak kullanımı ve performans açısından Model 3'ün diğer modellere göre daha iyi olduğu anlaşılmaktadır.

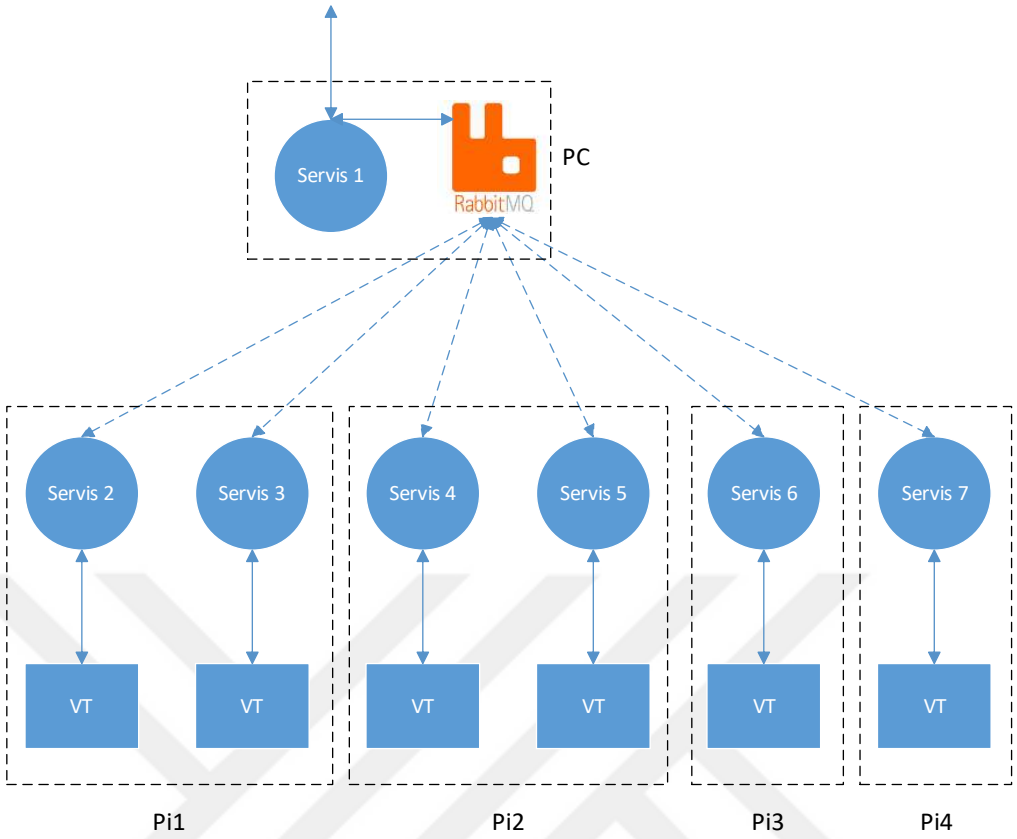
İkinci deneyler modellerin asenkron çalışmaları için gerçekleştirilmiştir. Kullanılan asenkron modeller Şekil 3.11, Şekil 3.12 ve Şekil 3.13'te verilmiştir. Asenkron iletişim RabitMQ tarafından oluşturulan kuyruk yapısı ile gerçekleştirilmiştir. İlk modelde tüm uç cihazlar kendi kuyruk yapısına sahipken, diğer modellerde kullanıcı etkileşimi olan uç cihazda bulunmaktadır. Temel deney parametreleri senkron çalışma ile aynıdır.



Şekil 3. 11. Model 1 (asenkron mod)

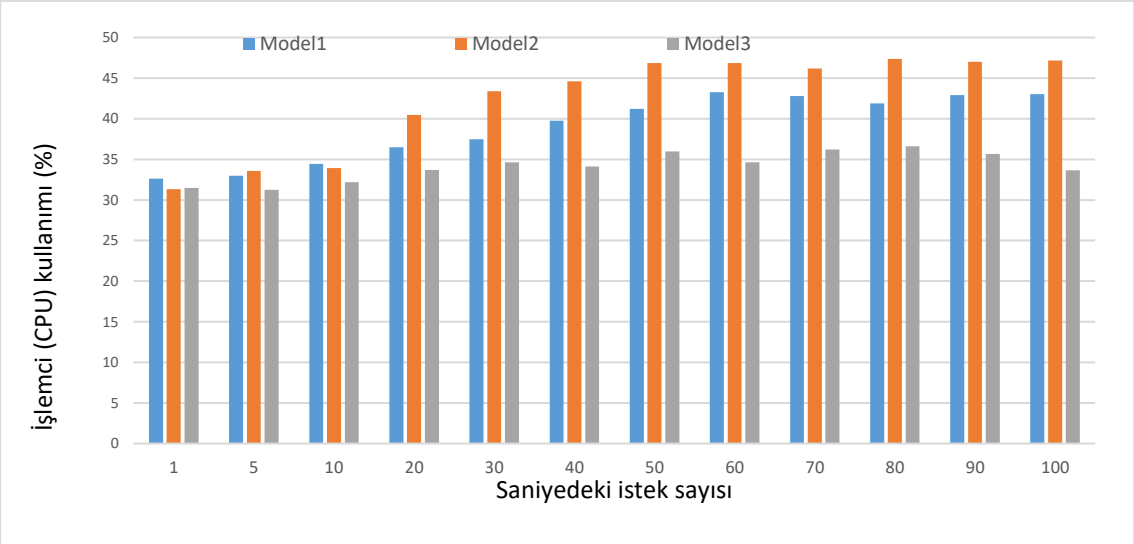


Şekil 3. 12. Model 2 (asenkron mod)

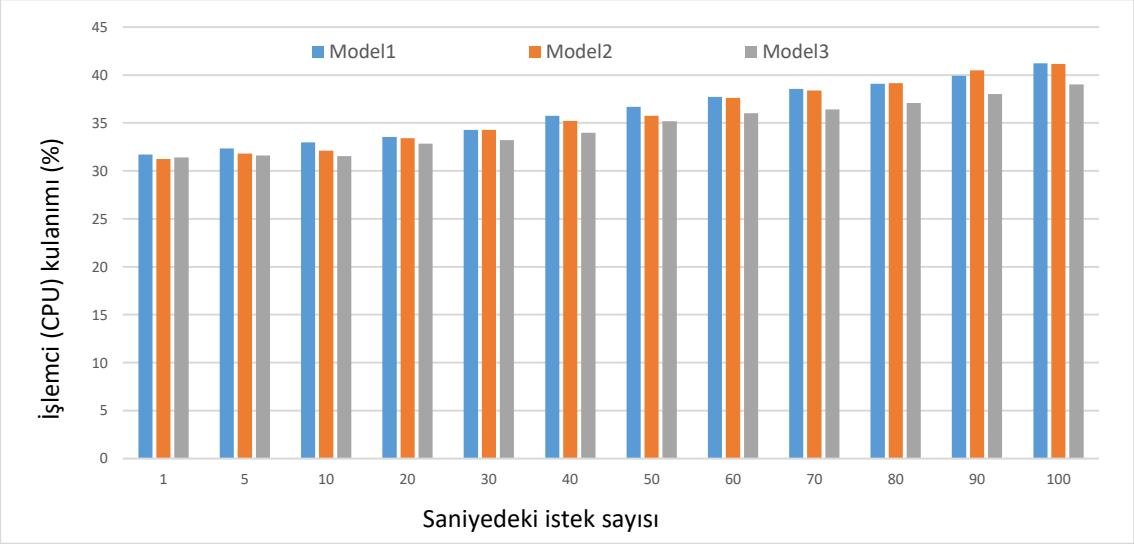


Şekil 3. 13. Model 3 (asenkron mod)

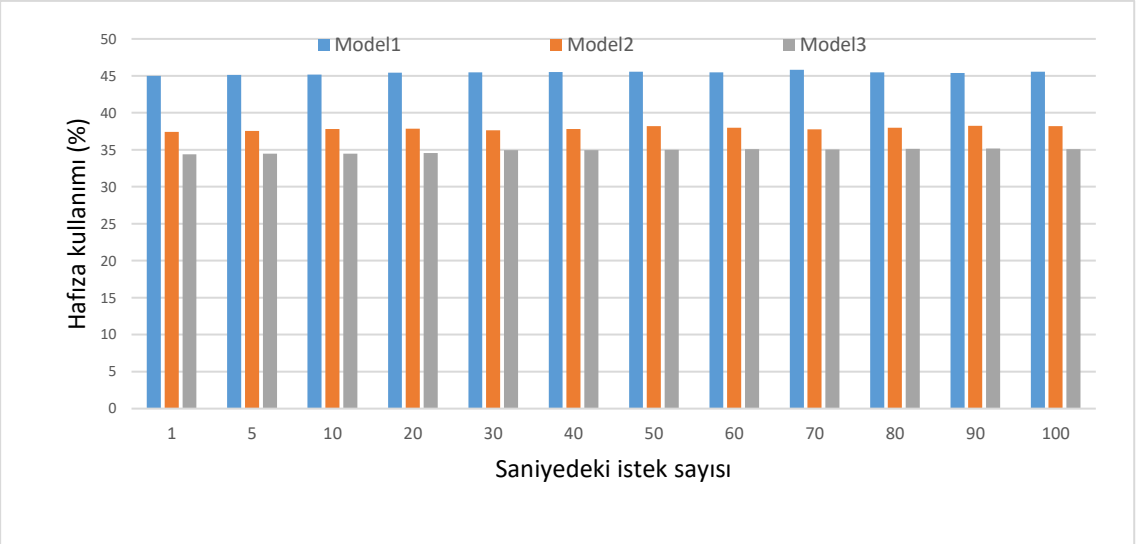
Senkron çalışan modeller üzerinde yapılan tüm deneyler asenkron çalışan modeller içinde yapılmıştır. İşlemci kullanım oranları Şekil 3.14 ve Şekil 3.15'te gösterildiği gibidir. Senkron ve asenkron çalışma arasındaki işlemci kullanım oranları karşılaştırıldığında, asenkron modellerde genel anlamda yaklaşık %10'luk bir artış meydana gelmiştir. Bu artışın sebebi asenkron çalışmanın gerçekleştirilmesi için oluşturulan kuyruk yapısından kaynaklanmaktadır. Bir diğer ölçüm metriği olan hafıza kullanım oranları Şekil 3.16 ve Şekil 3.17'de sunulmuştur. Hafıza kullanım oranlarında senkron ve asenkron modeller arasında Model 1 hariç önemli bir değişim olmamıştır. Model 1'in asenkron çalışmasında hafıza kullanım oranında senkron çalışmasına göre %25'lik bir artış yaşanmıştır. Bu artışın sebebi Model 1'in yapısı gereği her uç düğümde asenkron çalışma için oluşturulan kuyruk yapısının bulunmasıdır. Asenkron modellere yapılan isteklerin cevaplanma sürelerini ise Şekil 3.18 ve Şekil 3.19'da görmek mümkündür. Senkron ve asenkron çalışan modeller arasındaki en büyük değişim isteklerin cevaplanma süreleri değerlerinde olmuştur. Modeller asenkron olarak çalıştırıldığında, senkron çalıştırılmalarına göre cevap süresi değerlerinde en az %50'lik bir azalış olduğu görülmüştür. Bu sayede modellerin asenkron çalışmalarıyla birlikte gecikme süreleri büyük oranda düşürülmüştür.



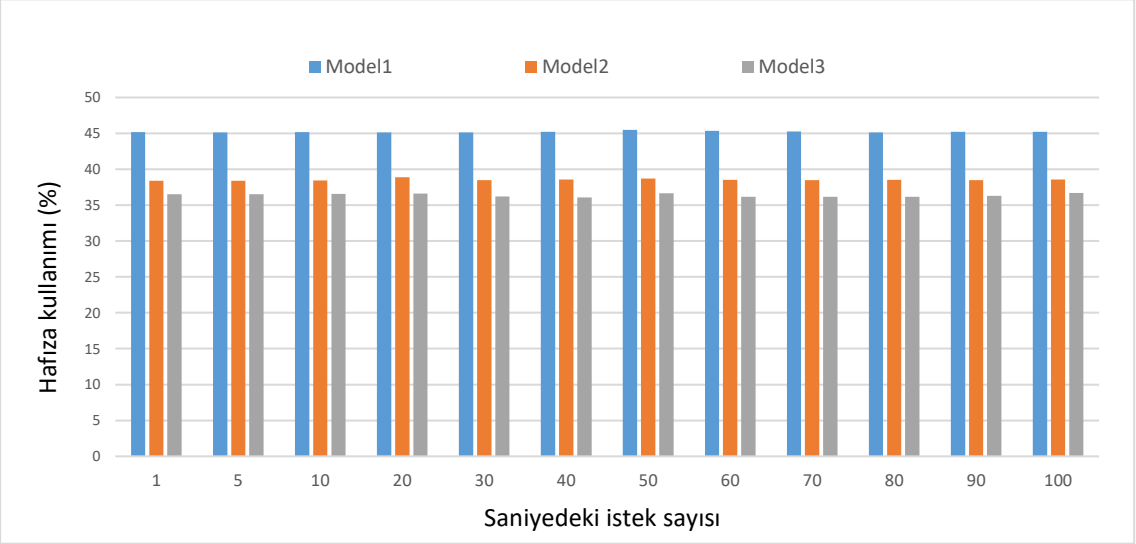
Şekil 3. 14. Select işlemi için işlemci (CPU) kullanım oranı (asenkron mod)



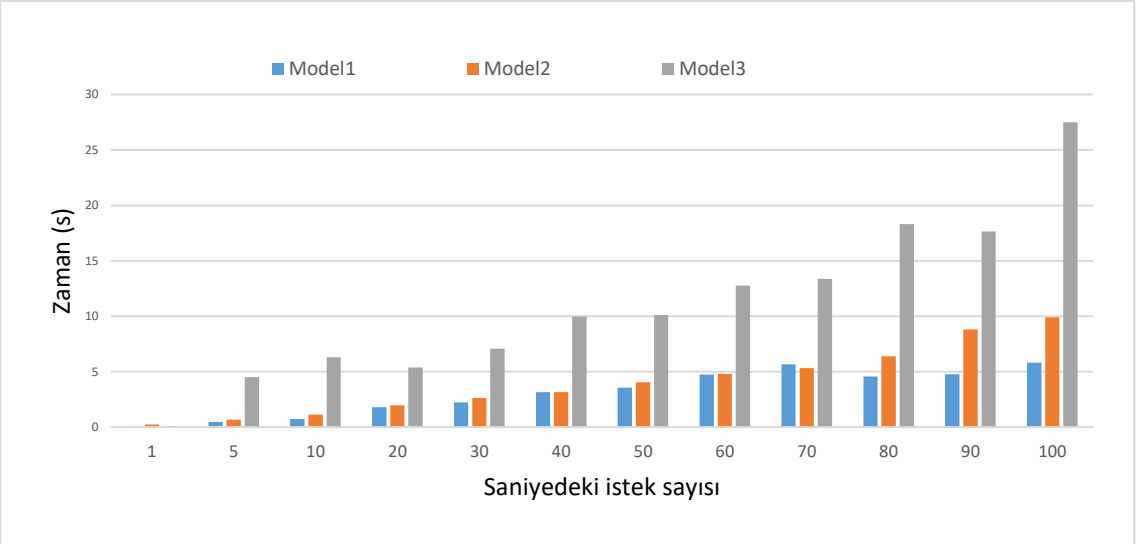
Şekil 3. 15. Insert, delete ve update işlemlerine ait işlemci (CPU) kullanımı (asenkron mod)



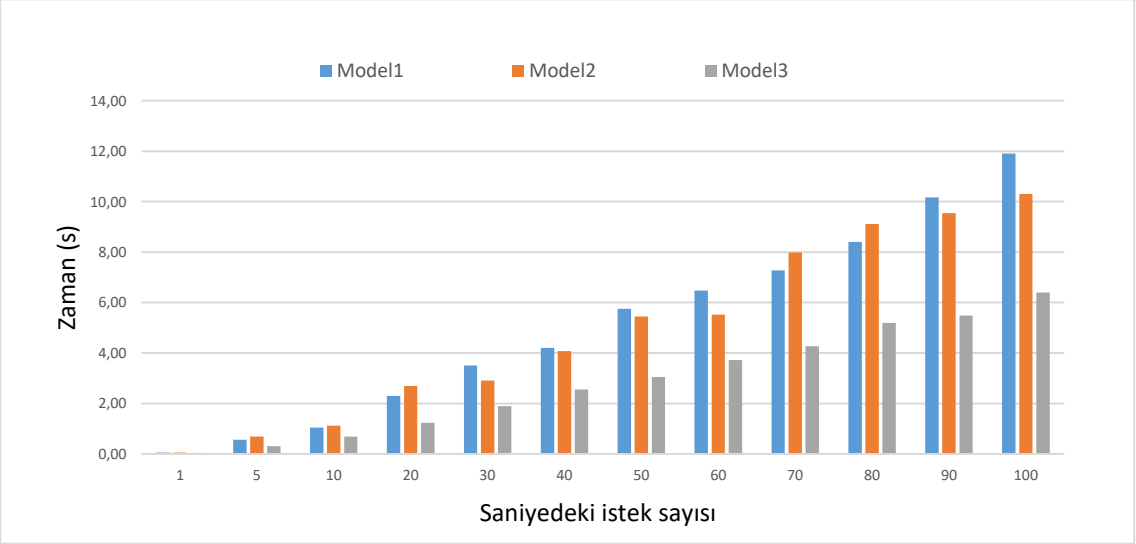
Şekil 3. 16. Select işlemine ait hafıza kullanımı (asenkron mod)



Şekil 3. 17. Insert, delete ve update işlemlerine ait hafıza kullanımı (asenkron mod)



Şekil 3. 18. Select işlemine ait cevap süreleri (asenkron mod)



Şekil 3. 19. Insert, delete ve update işlemlerine ait cevap süreleri (asenkron mod)

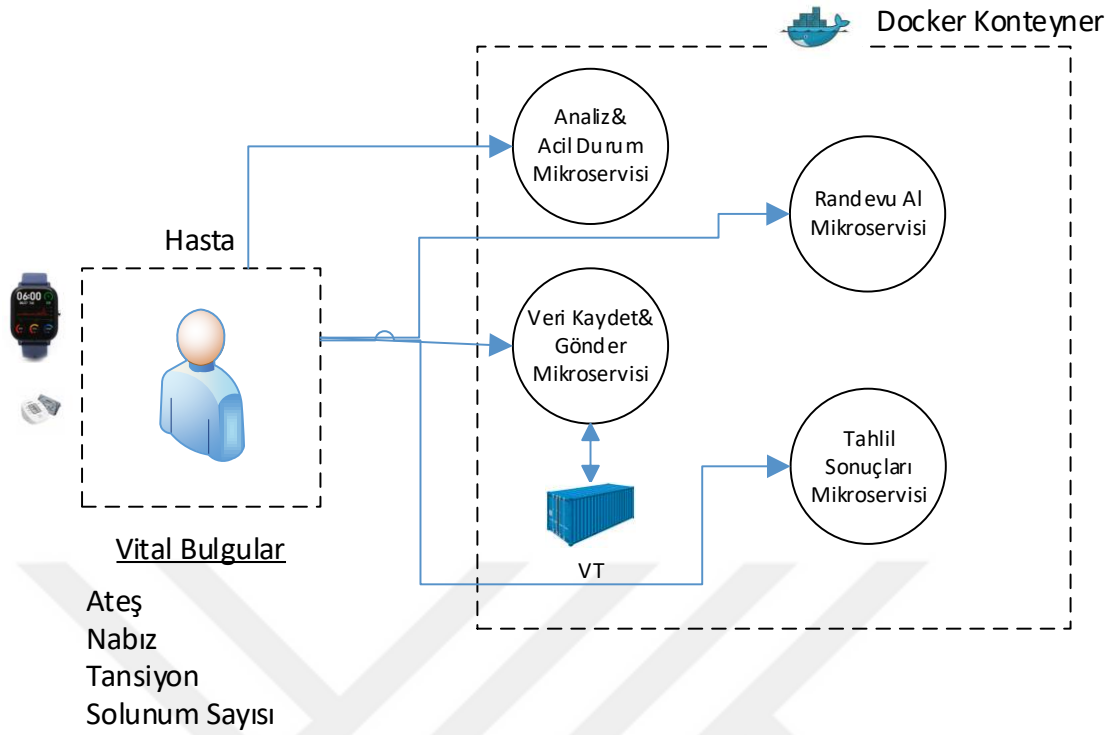
İşlemci ve hafıza kullanım oranları dikkate alınarak en iyi modelin Model 3 olduğu görülmüştür. Select işlemi için Model 1'in cevap süresi diğer modellere göre daha iyidir. Insert, delete ve update işlemlerine bakılacak olunursa en iyi cevap süresi değerinin Model 3'e ait olduğu görülmektedir. Asenkron çalışan modellerde kullanıcı sayısı arttırıldığında da mikroservislerin düzgün çalıştığı görülmüştür. Asenkron çalışan modellerin ölçüm değerlerine bakılırsa genel olarak en iyi modelin Model 3 olduğu anlaşılmaktadır.

Önerilen sistemin ilk safhasında, mikroservis temelli uç sistemler için üç farklı veri depolama ve yönetim modeli incelenmiştir. Deneyler, mikroservislerin hem senkron hem de asenkron çalışma durumları için yapılmıştır. Depolama ve sorgulama için ilişkisel veri tabanının kullanıldığı

deneylerde, mikroservisler dağıtık uç cihazlar üzerinde çalıştırılmıştır. Yapılan deneyler sonucunda, uç sistemlerde mikroservis veri paylaşım yönetimi için en iyi modelin Model 3 olduğu anlaşılmıştır.

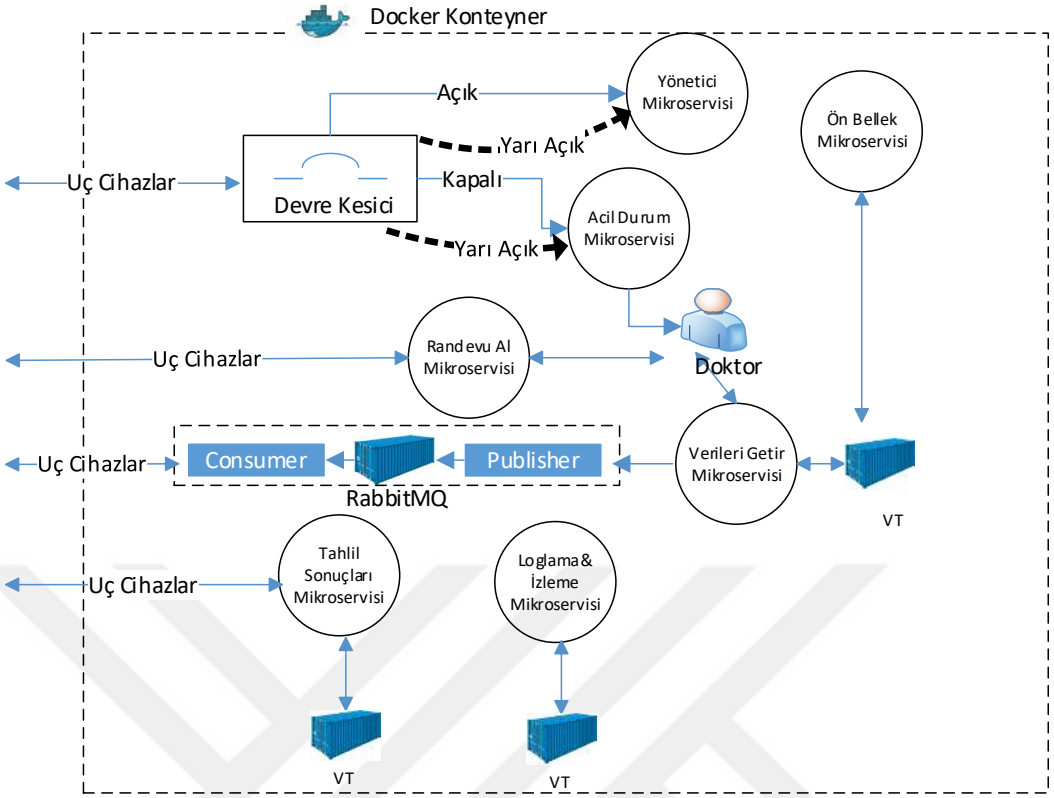
3.2. Bulut Teknolojisi Kullanan Hasta Takip Hizmetleri İçin Önerilen Mikroservis Temelli Uç Sistem Tasarımı

İkinci safhada önerilen sistemin bileşen yapısı ve kullanılacak olan mikroservis mimarisi tasarlanmıştır. Birinci safhada belirlenen veri yönetim mekanizması bu tasarımın geliştirilmesinde belirleyici faktörlerden birisi olmuştur. Önerilen sistem, aile hekimlikleri kapsamında bulunan acil durumlu hastaların uzaktan takip edilebilmesine yönelik tasarlanmıştır. Sistem üç ana üniteden meydana gelmektedir. Bunlar Hasta Uç Ünitesi (HUÜ), Aile Hekimliği Uç Ünitesi (AHUÜ) ve Bulut Ünitesidir (BÜ). HUÜ gözlemlenecek olan hasta lokasyonunda bulunur ve kısıtlı kaynaklı bir donanım üzerinde yürütülür. Şekil 3.20'den görüleceği üzere HUÜ'de toplamda 4 adet mikroservis çalışmaktadır. Bu mikroservisler, Analiz&Acil Durum, Veri Kaydet&Gönder, Tahlil Sonuçları ve Randevu Al mikroservisleridir. Hastadan alınan veriler Analiz&Acil Durum ve Veri Kaydet&Gönder mikroservislerine gönderilmektedir. Veri Kaydet&Gönder mikroservisinin görevi hastadan alınan vital bulgu (Ateş, Nabız, Tansiyon ve Solunum Sayısı) verilerini veri tabanına kaydetmek ve aile hekiminin gerek duyduğu zamanlarda verileri paylaşmaktır. Analiz&Acil Durum mikroservisine gelen veriler hastanın yaşı ve durumu itibariyle olması gereken normal değerler ile karşılaştırılmaktadır. Hastanın vital bulgu değerlerinden herhangi birinin hasta üzerinde risk oluşturabilecek duruma ulaşması halinde, hastanın aile hekimine ve gerekli sağlık kuruluşlarına bildirim gönderilmektedir. Tasarlanan bu hasta takip sisteminde sadece hasta izleme odaklı düşünülmemiş, ayrıca hastanın tahlil sonuçları ve randevu alma istekleri de dikkate alınmıştır. Hastanın bu isteklerini karşılamak için Randevu Al adında bir randevu alma mikroservisi ve Tahlil Sonuçları adında da tahlil sonuçlarını görebileceği bir mikroservis oluşturulmuştur.



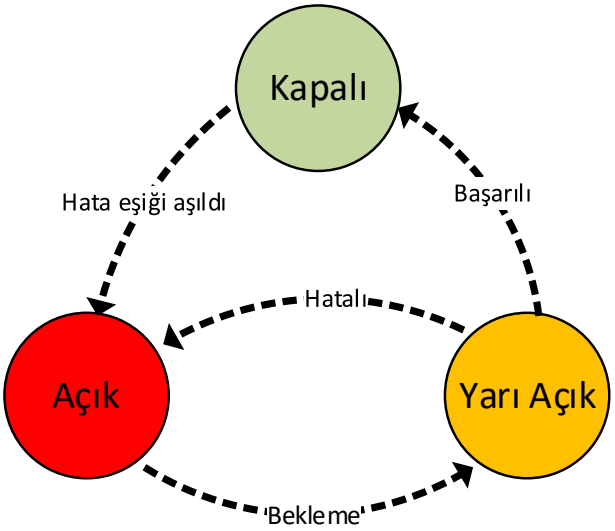
Şekil 3. 20. Hasta tarafındaki uç cihaz yapısı(HUÜ)

Hasta randevu almak istediği tarihi ve saati Randevu Al mikroservisi aracılığıyla aile hekimliğine bildirmektedir. Aile hekimliği tarafındaki aynı isimdeki mikroservis hastanın randevu için istediği tarih ve saati kontrol ederek randevunun uygun olup olmadığı bilgisini hastaya dönmektedir. Hasta bu sistem üzerinden tahlil sonuçlarını da görebilmektedir. Tek yapması gereken Tahlil Sonuçları mikroservisine tahlil sonuçlarını görmek istediğini bildirmektir. Tahlil Sonuçları mikroservisi aile hekimliği tarafındaki aynı isimdeki mikroservis ile haberleşerek hastaya ait tahlil sonuçlarını istemektedir. Hastalara ait tahlil sonuçları aile hekimliğinde ilişkisel veri tabanında tutulmaktadır. Aile hekimliğindeki Tahlil Sonuçları mikroservisi hasta bilgileri ile veritabanına sorgulama yapmaktadır. Tahlil sonuçları bulunan hastalara sonuçlar gönderilmektedir. Tahlil sonucu bulunmayan hastalara sonuçlarının henüz hazır olmadığı bilgisi geri dönülmektedir. Bu hasta izleme sisteminde aile hekimliği tarafında da uç cihaz kullanılmaktadır. Şekil 3.21'de aile hekimliği tarafındaki uç cihazın yapısı gösterilmektedir.



Şekil 3. 21. Aile hekimliği tarafındaki uç cihaz yapısı(AHUÜ)

Bu cihaz üzerinde toplam 7 adet mikroservis çalışmaktadır. Bu mikroservislerden Acil Durum mikroservisi hastalardan gelen acil durum bilgilerini almaktadır. Bu mikroservis, sistemin düzgün çalışabilmesi için önem arz etmektedir. Hastalardan gelen acil durum bilgilerinin alınamaması sistemin amacına uygun çalışamadığını gösterir. Bu sebepten dolayı bu mikroservisin bir devre kesici yardımıyla takip edilmesi ve güvenliğinin sağlanması gereklidir. Burada kullanılan devre kesici, Acil Durum mikroservisinin cevap süresinin belirlenen eşik değerini aşması durumunda devreye girmektedir. Bu çalışmada eşik değeri 80 milisaniye olarak belirlenmiştir. Bu değer belirlenirken Acil Durum mikroservisine istekler gönderilmiş ve en yüksek cevaplanma sürelerinin ortalaması alınarak bulunmuştur. Eşik değerinin aşıldığı durumlar Acil Durum mikroservisinin yoğun olarak çalıştığı durumlardır. Bu zamanlarda Acil Durum mikroservisi üzerindeki yoğunluğu azaltmak mikroservisin düzgün çalışabilmesi için önemlidir. Şekil 3.22'de devre kesicinin durum diyagramı gösterilmektedir.

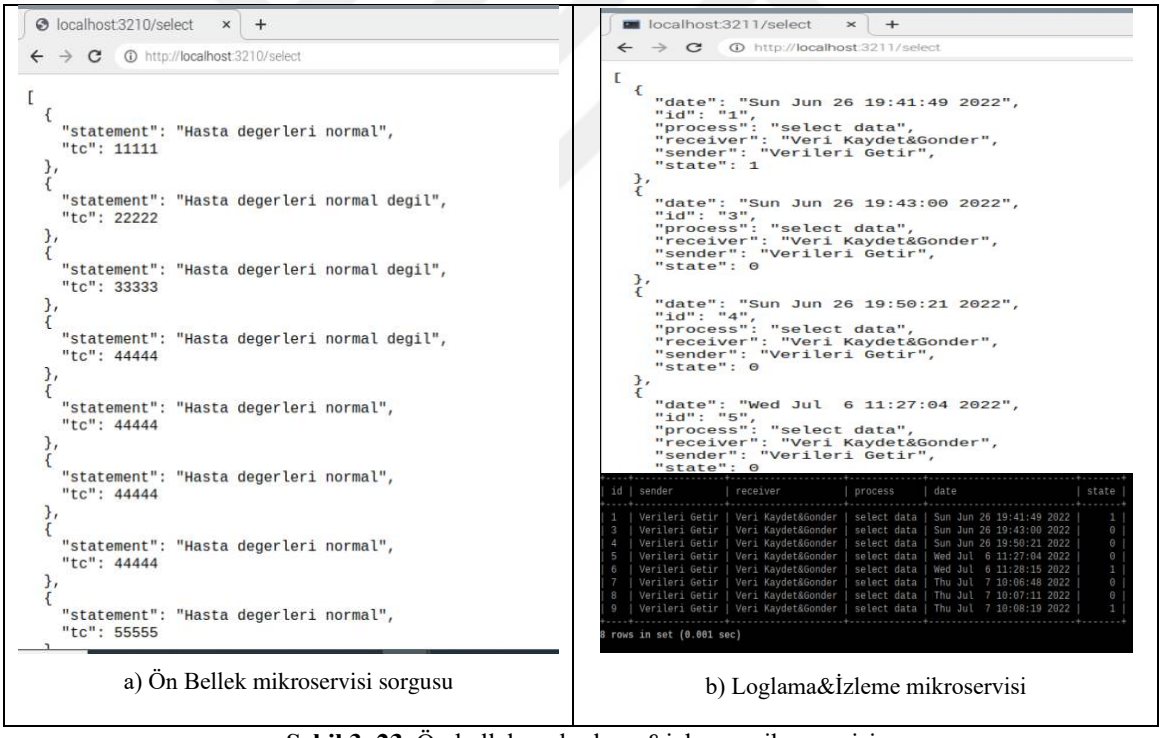


Şekil 3. 22. Devre kesici durum diyagramı

Devre kesici, Acil Durum mikroservisinin cevap süresini izler. Cevap süresinin belirlenen eşik değerini aşması durumunda Acil Durum mikroservisini olası kötü senaryolardan korumak için anahtarı açık konumuna getirerek devre kesiciyi açar ve gelen istekleri Yönetici mikroservisine yönlendirir. Bu sayede Acil Durum mikroservisinin yoğunluğu azaltılmış olunur. Bundan sonra gelecek belirli sayıdaki isteklerin tümünü direkt olarak Yönetici mikroservisi alır. Bu çalışmada 5 istek sayısı kadar beklenmektedir. Bu şart sağlandıktan sonra anahtar yarı açık konumuna geçer ve gelen ilk 5 istek Yönetici ile birlikte Acil Durum mikroservisine de gönderilir. Burada Acil Durum mikroservisinin düzgün çalışıp çalışmadığı kontrol edilir. Acil Durum mikroservisi gelen ilk 5 isteğe de istenen düzeyde cevap verir ise anahtar kapalı konumuna geçer ve yeni gelecek olan isteklerin tümü sadece Acil Durum mikroservisine iletilir. Eğer Acil Durum mikroservisi gelen ilk 5 isteğe de istenen düzeyde cevap veremez ise anahtar tekrar açık konumuna geçerek yeni gelecek isteklerin Yönetici mikroservisine gönderilmesi sağlanır.

Aile hekimi takip ettiği hastaların verilerini belirli aralıklar ile görmek isteyebilir. Bu durumu gerçekleştirmek için aile hekimliği tarafında Verileri Getir mikroservisi bulunmaktadır. Aile hekiminin bu mikroservise sadece verilerini görmek istediği hasta bilgilerini söylemesi yeterlidir. Asenkron olarak iletişim kuran bu mikroservis hasta verilerini getirip kendine ait veritabanında saklar. Doktor tarafından yapılan teşhis bilgileri de bu veritabanında saklanmaktadır. Tasarlanan bu uç sistemde, bakanlığın veya hastanenin aile hekiminin hastalar için yaptığı teşhis bilgilerine de ulaşmasına imkan tanınmaktadır. Bu durumda bakanlığa veya hastaneye ait buluttan aile hekimliğine belirli aralıklarla aynı istekte bulunulacağı için araya bir Ön Bellek mikroservisi kullanılması cevap süresi için avantaj sağlayacaktır. Ön Bellek mikroservisi buluttan gelen isteğe ait cevap bilgisini kendinde tutmaktadır. Aile hekiminin herhangi bir hasta için yaptığı teşhisten Ön Bellek mikroservisinin haberi olmaktadır. Bu sayede Ön Bellek mikroservisi her zaman için en

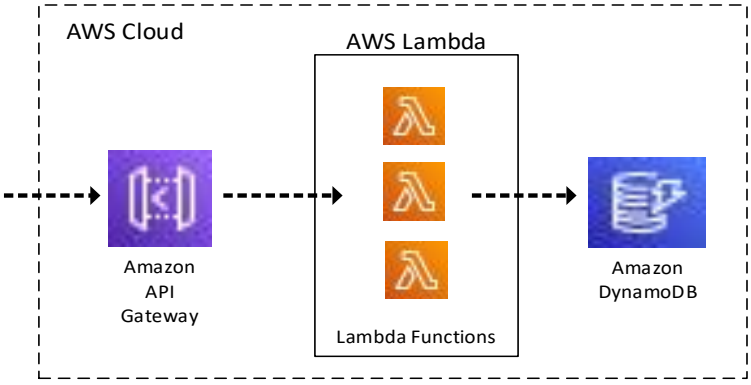
güncel teşhis bilgilerine sahip olmaktadır. Şekil 3.23a'da Ön Bellek mikroservisine ait örnek sorguyu görmek mümkündür. Buluttan gelen her istekte hastalara ait teşhis bilgileri için veri tabanına başvurulmamaktadır. Bu durum buluttan gelen isteğin cevap süresini azaltmaktadır. Uç sistemin izlenmesi ve olası hataların kayıt altına alınması için Loglama&İzleme mikroservisi bulunmaktadır. Bu mikroservis hasta takip sistemi üzerindeki tüm mikroservisleri izlemekte ve servisler arasındaki iletişim bilgilerini kendine ait veritabanında tutmaktadır. Bu veritabanında isteği yapan mikroservis, isteği alan mikroservis, isteğe ait bilgi, isteğin yapıldığı tarih ve isteğin başarılı olup olmadığı bilgileri tutulmaktadır. Bu sayede sistem üzerinde yaşanacak problemlerin tespitinin daha kolay bir şekilde yapılması sağlanmaktadır. Şekil 3.23b'de Loglama&İzleme mikroservisinin yapısı gösterilmiştir. Bu çalışmada Amazon Web Servisi(AWS)'ne ait bulut hizmeti kullanılmıştır. Şekil 3.24'te kullanılan AWS bulut yapısı gösterilmektedir. Buluta gelen istekler API Gateway üzerinden gerekli Lambda fonksiyonlarına iletilmektedir. Bulut üzerinde gerek duyulan verilerin kaydedilmesi için Amazon DynamoDB veritabanı kullanılmaktadır.



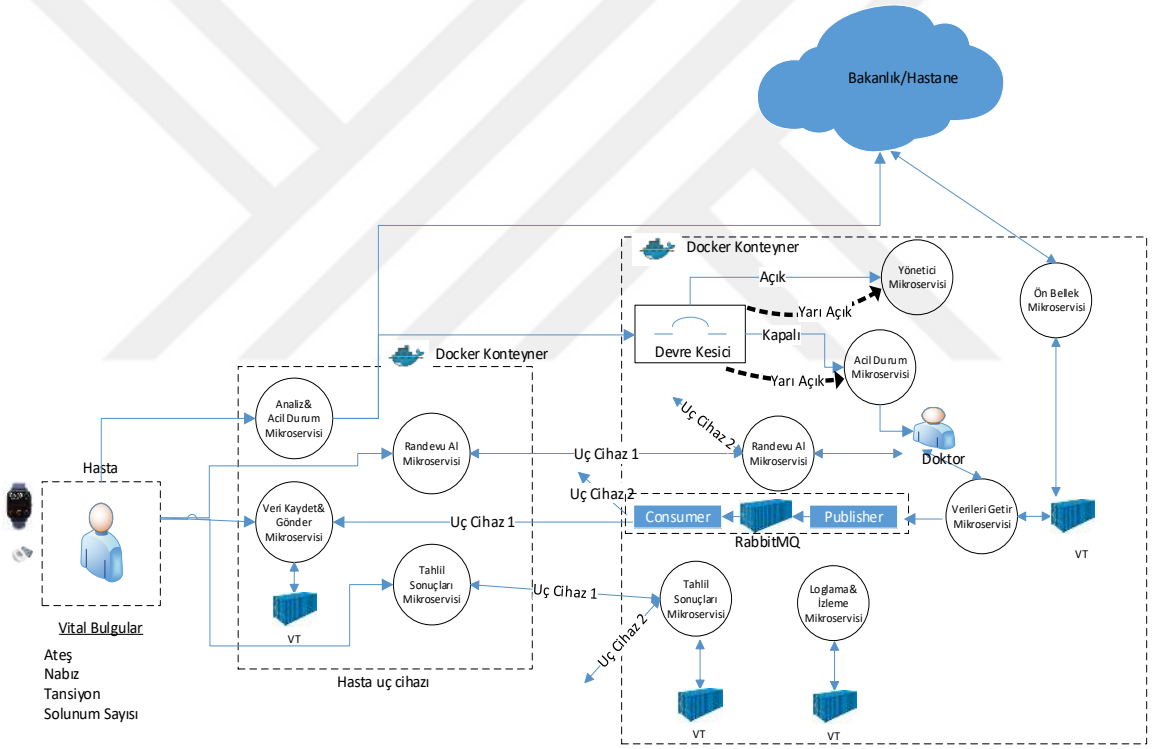
a) Ön Bellek mikroservisi sorgusu

b) Loglama&İzleme mikroservisi

Şekil 3. 23. Ön bellek ve loglama&izleme mikroservisi



Şekil 3. 24. Bulut yapısı (BU)

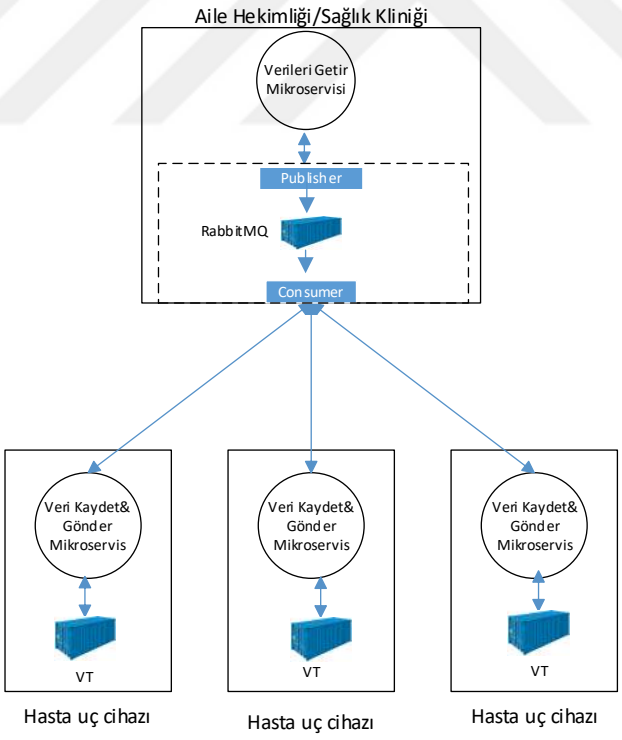


Şekil 3. 25. Hasta takip sistemi genel yapısı

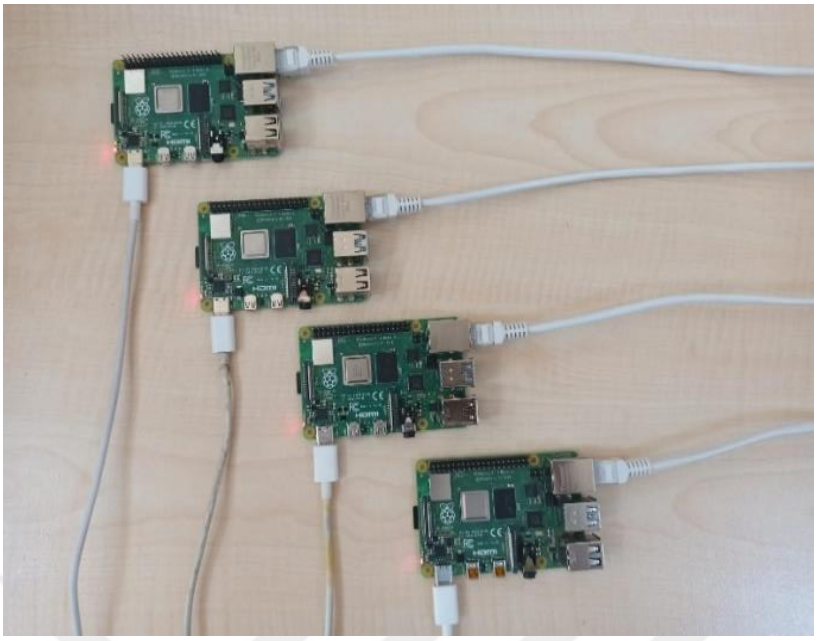
Şekil 3.25'te bulut teknolojisi kullanan hasta takip hizmetleri için önerilen mikroservis temelli uç sistemin genel yapısı gösterilmektedir. Bu hasta takip sistemde ilişkisel veri tabanı kullanılmaktadır. Mikroservisler kendi aralarında ve bulut ile senkron ve asenkron olarak haberleşmektedirler. Asenkron haberleşme için kuyruk yapısı bulunmaktadır. İlişkisel veri tabanı olarak MariaDB ve kuyruk yapısı için RabbitMQ kullanılmaktadır. Veri tabanları, kuyruk yapıları ve mikroservisler konteyner olarak çalıştırılmaktadır. Hasta takip sistemindeki tüm konteynerlerin kontrolü ve yönetimi Docker Swarm tarafından yapılmaktadır. Docker Swarm sayesinde,

konteynerler üzerinde yük dengeleme, izleme ve konteynerlerden herhangi birinin çalışamaz duruma gelmesi halinde aynı konteynerden yenisinin oluşturulması işlemleri yapılabilmektedir.

Şekil 3.25'te görüldüğü üzere hasta takip sistemi üzerindeki tüm işlemler mikroservisler aracılığıyla yapılmaktadır. Hastadan alınan vital bulgulardan (ateş, nabız, tansiyon ve solunum sayısı) oluşan veriler hasta tarafındaki uç cihaza iletilmektedir. Bu sistemde hem hasta tarafında hem de aile hekimliği tarafında uç cihaz bulunmaktadır. Şekil 3.26'da uç cihazlar arasındaki ilişkiyi görmek mümkündür. Bu kısımda çalışmanın ilk safhasında yapılan deneyler sonucu belirlenen en performanslı veri yönetim modeli olan Model 3 yapısı kullanılmıştır. Bu çalışma gerçekleştirilirken uç cihaz olarak Şekil 3.27'de gösterilen Raspberry Pi 4 Model B kullanılmıştır. Uç cihazlar üzerine mikroservisler Python dilinde kodlanmıştır. Tasarlanan sisteme Apache JMeter yardımıyla istekler gönderilmiş ve sistemin bu istekleri gerçekleştirirken uç cihazların işlemci kullanımı, hafıza kullanımı ve mikroservislerin isteklere verdikleri cevap süreleri izlenmiştir. Bu ölçümlerin yapılabilmesi için uç cihazlarda Tcpreplay kullanılarak ağ üzerinde trafik oluşturulmuş ve Zabbix Agent üzerinden de uç cihazların performans değerleri ölçülmüştür[27].



Şekil 3. 26. Uç cihazlar arası ilişki

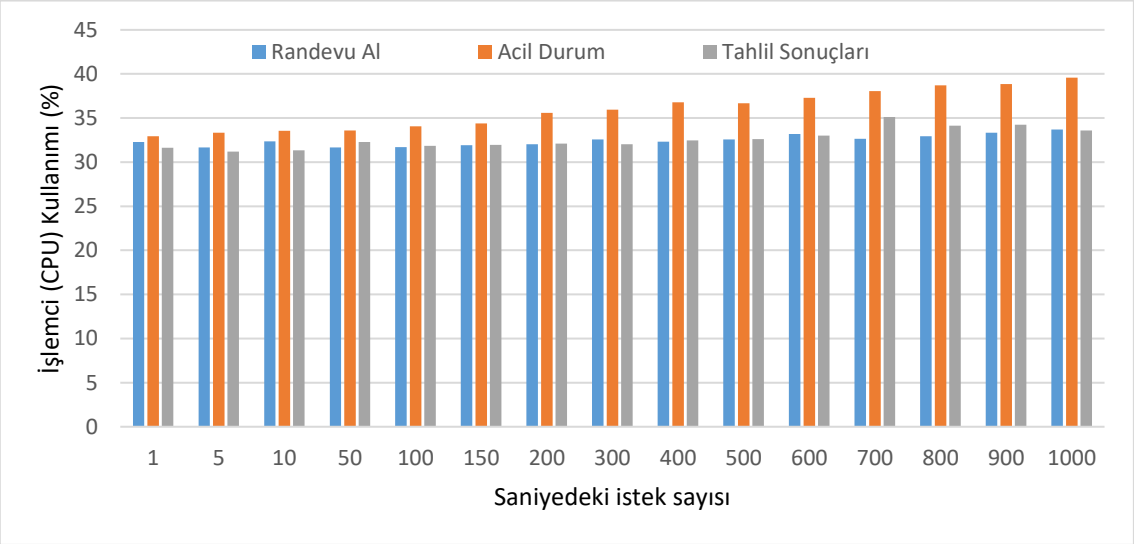


Şekil 3. 27. Kullanılan raspberry pi'ler

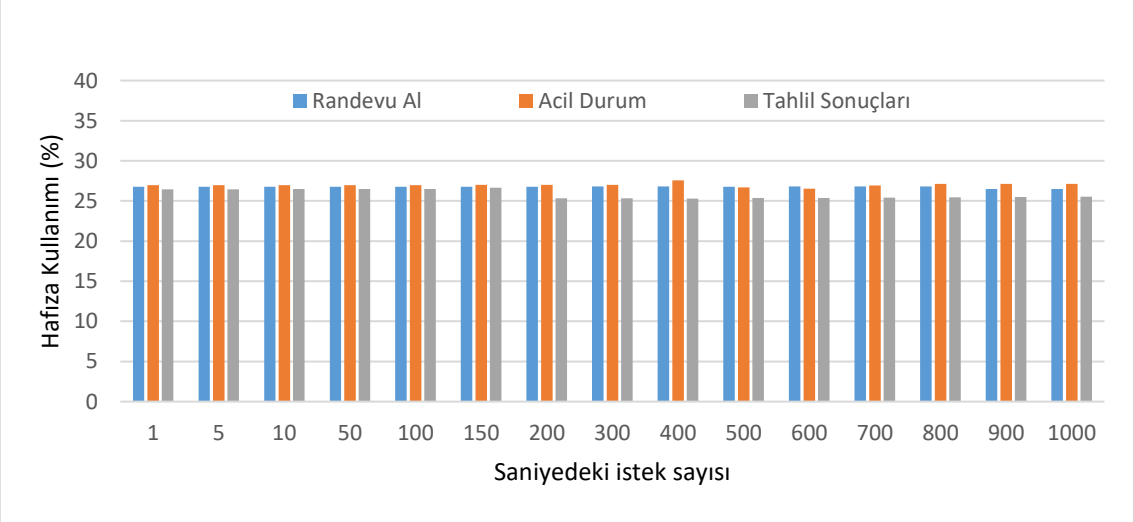
Acil Durum mikroservisine yapılan isteklerde özellikle saniyede 100 istek ve üzerindeki durumlarda devre kesicinin açıldığı görülmüştür. Ölçümler esnasında devre kesici sayesinde Acil Durum mikroservisinin hiçbir zaman için çalışamaz duruma gelmediği gözlenmiştir. Bu durum tasarlanan bu uç sistemin her zaman için kararlı olarak çalıştığını göstermektedir. Ayrıca Ön Bellek mikroservisinin kullanımı sayesinde buluttan gelen isteklerin cevap sürelerinin %40 oranında azaldığı gözlenmiştir.

4. BULGULAR VE TARTIŞMA

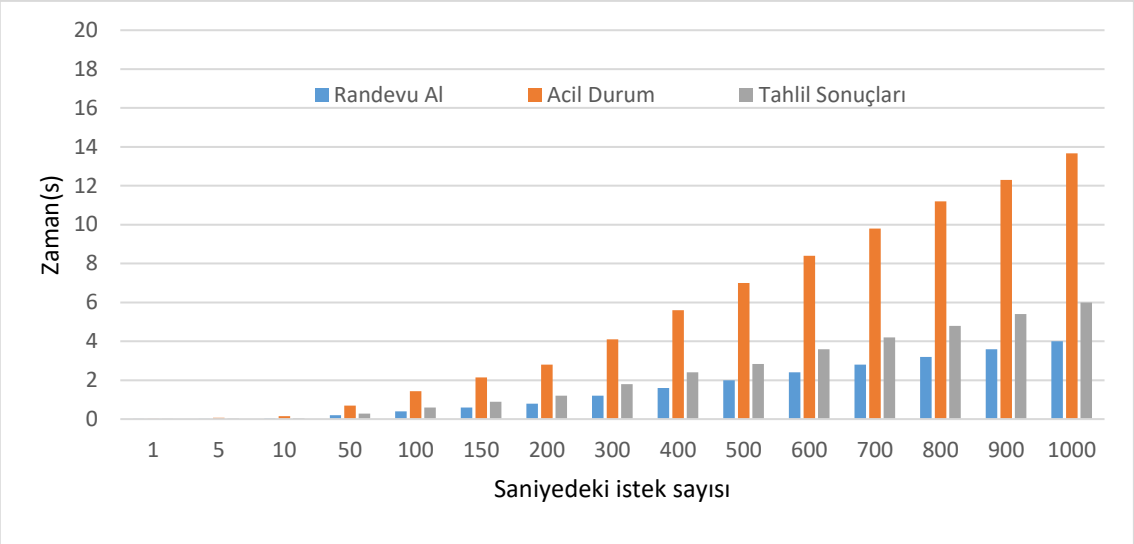
Aile hekimliği tarafındaki uç cihazda çalışan Tahlil Sonuçları, Acil Durum ve Randevu Al mikroservisleri hastalardan gelecek olan istekleri karşıladıkları için ölçümler bu mikroservisler üzerinde yapılmıştır. Ölçümler alınırken, ölçüm değerleri saniyede 1000 isteğe kadar test edilmiş ve tasarlanan sistemin problemsiz çalıştığı gözlenmiştir. Gerçek çalışma ortamında isteklerin sayısının bu kadar yüksek olması beklenmemektedir. Fakat deneyler yapılırken gerçeğin daha üstü değerler ile test edilmiştir. Şekil 4.1'de aile hekimliği tarafındaki uç cihazın işlemci kullanımı, Şekil 4.2'de hafıza kullanımı ve Şekil 4.3'te de isteklerin cevap süreleri gösterilmektedir.



Şekil 4. 1. İşlemci (CPU) kullanımı

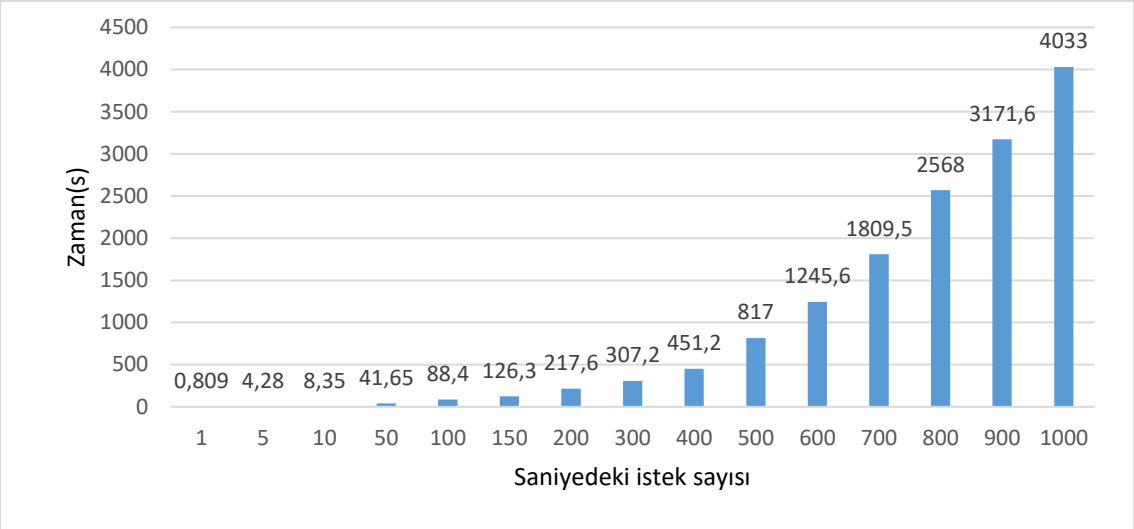


Şekil 4. 2. Hafıza kullanımı



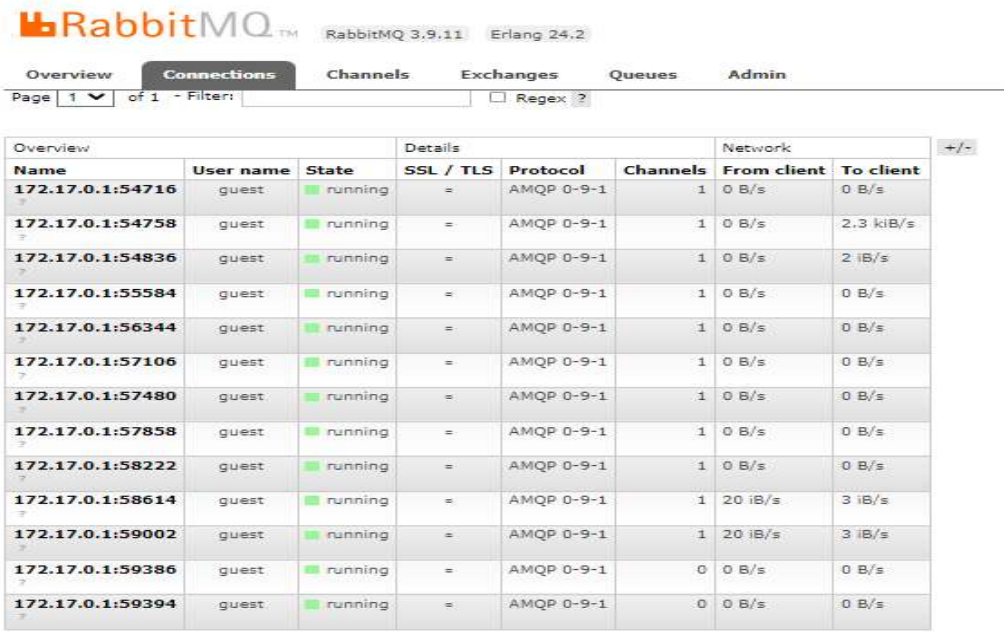
Şekil 4. 3. İsteklerin cevaplanma süreleri

Grafikler incelendiğinde aile hekimliğine yapılan istekler sonucunda en çok işlemci kullanımının Acil Durum mikroservisi üzerine gelen isteklerin gerçekleştirilmesi sırasında harcandığı görülmektedir. Bunun sebebi, Acil Durum mikroservisinin bir devre kesiciye sahip olmasıdır. Acil Durum mikroservisine gelen isteklerde diğer isteklerin aksine devre kesici, Acil Durum mikroservisini takip etmek için fazladan işlem yapmaktadır. Bu durum işlemci kullanımını ve isteğin cevaplanma süresini artırmaktadır. Yapılan deneylerde uç cihazın (AHUÜ) her durumda işlemci kullanımının %40'ın altında olduğu görülmüştür. Kaynak kısıtlı cihazlar üzerinde geliştirilen uygulamaların performans kriterleri oldukça önemlidir. İsteklerin cevaplanma süreleri göz önüne alındığında ikinci en yüksek değerin Tahlil Sonuçları mikroservisine ait olduğu görülmektedir. Tahlil Sonuçları mikroservisinin cevap süresinin Randevu Al mikrosevisinin cevap süresinden daha yüksek olmasının sebebi hastalara ait tahlil sonuçlarının ilişkisel veritabanında tutulması ve Tahlil Sonuçları mikroservisine gelen isteklerde veritabanı üzerinde sorgulama ve veri çekme işlemlerinin gerçekleşmesidir. Tüm bu işlemler gerçekleştirilirken bir başka önemli kriter olan hafıza kullanımı da ölçülmüştür. Uç cihaz (AHUÜ) üzerinde yapılan isteklerde hafıza kullanım oranının her durum için yaklaşık %26 olduğu görülmektedir. Uç cihaz (AHUÜ) üzerindeki tüm mikroservisler konteyner olarak çalıştırılmaktadır. Fakat bu durum hafıza kullanımı üzerinde olumsuz bir etkiye neden olmamaktadır. Bulut üzerinde de gerekli deneyler yapılmış ve olası acil durumlarda istekleri cevaplama süreleri ölçülmüştür. Şekil 4.4'te buluta yapılan isteklerin cevap süreleri gösterilmektedir. Değerler dikkate alındığında uç cihaz (AHUÜ) ile bulutun (BU) istekleri cevaplama süreleri arasındaki büyük farkı görmek mümkündür.

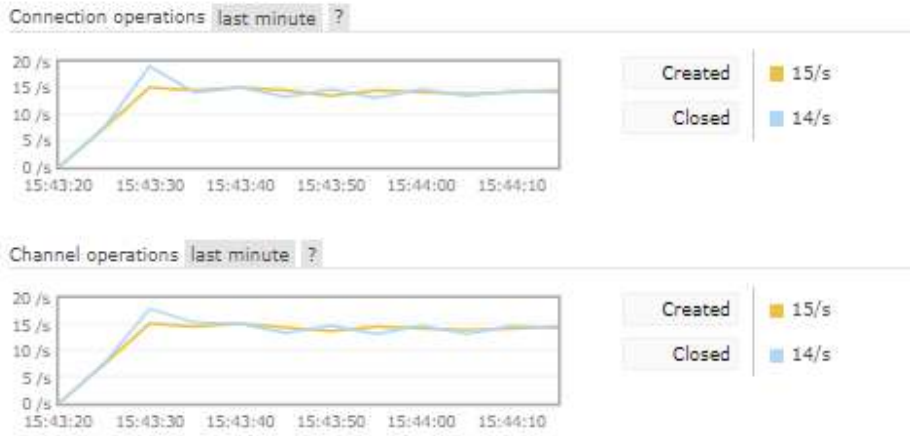


Şekil 4. 4. Buluta yapılan isteklerin cevap süreleri

Hasta takip sisteminde bazı mikroservisler kendi aralarında asenkron olarak haberleşmektedirler. Asenkron haberleşmede isteklerin diğer mikroservise iletiminde kuyruk yapısı kullanılmıştır. Bu tez çalışmasında kuyruk yapısı için RabbitMQ tercih edilmiştir. RabbitMQ AMQP protokolünü kullanır ve çalışırken istekleri almak ve gerekli mikrosevise iletmek için kanallar oluşturur. Arayüzü sayesinde kurulduğu cihaz üzerindeki oluşturulan tüm kanalları ve bu kanallardan alınan ve gönderilen mesaj bilgilerini görmek mümkündür. Şekil 4.5'te RabbitMQ tarafından oluşturulan kanallara ait görsel verilmişken, Şekil 4.6' da kanallar üzerinde oluşturulan ve gönderilen mesajlara ait örnek istatistik görselleri sunulmuştur.

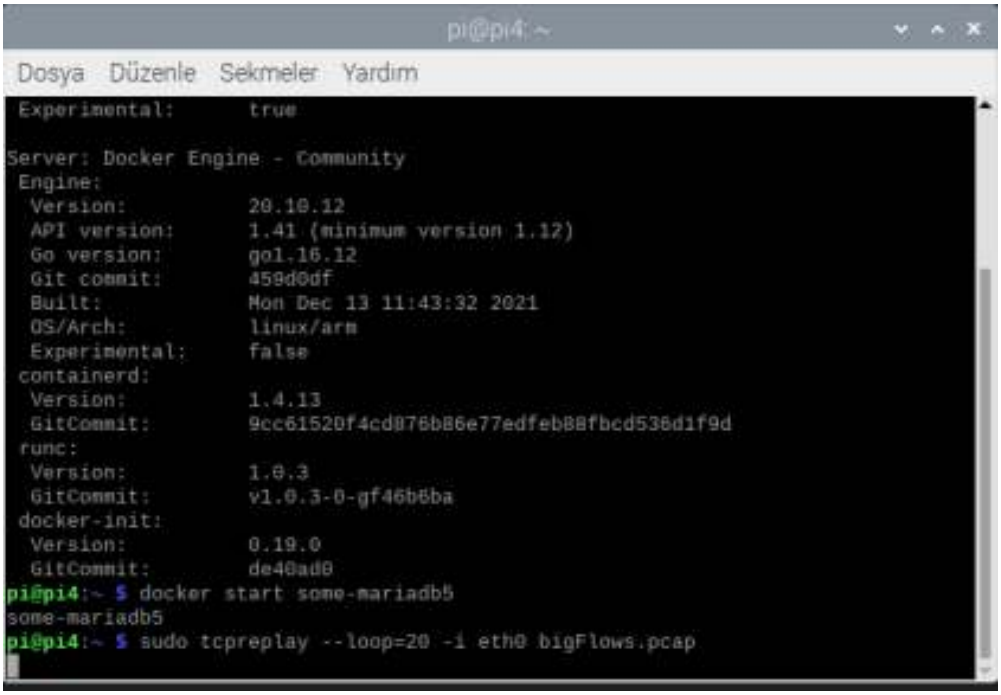


Şekil 4. 5. RabbitMQ tarafından oluşturulan kanallar

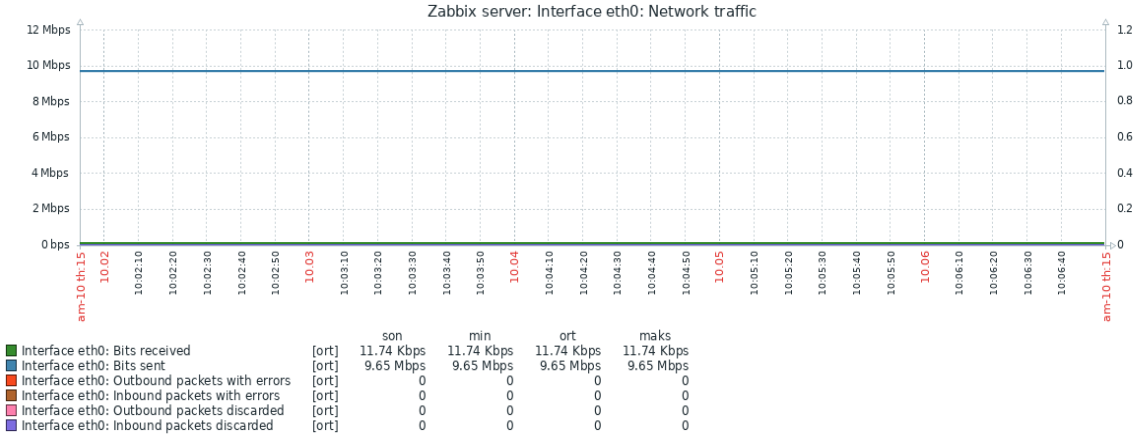


Şekil 4. 6. RabbitMQ kanal istatistikleri

Tcpreplay ile önceden ağ üzerinde yakalanan paketleri yeniden ve farklı hızlarda oynatmak mümkündür. Hâlihazırda Tcpreplay'in kendi internet sayfasında kullanılabilecek .pcap dosyalar bulunmaktadır. Bu tez çalışması gerçekleştirilirken Tcpreplay'in hazır dosyalarından olan bigFlows.pcap dosyası kullanılmıştır. Şekil 4.7 Tcpreplay ile eth0 üzerinden bigFlows.pcap dosyasını kullanarak trafik oluşturmayı, Şekil 4.8 Zabbix üzerinden alınan ağ trafiğine ait değerleri göstermektedir. BigFlows.pcap dosyası uç cihaz (AHUÜ) üzerinde farklı hızlarda çalıştırılmış ve Acil Durum mikroservisinin istekleri cevaplama süreleri ile devre kesicinin kaç defa açıldığı izlenmiştir. Elde edilen sonuçlar Tablo 4.1'de verilmiştir.



Şekil 4. 7. Tcpreplay ile eth0 üzerinde trafik oluşturma

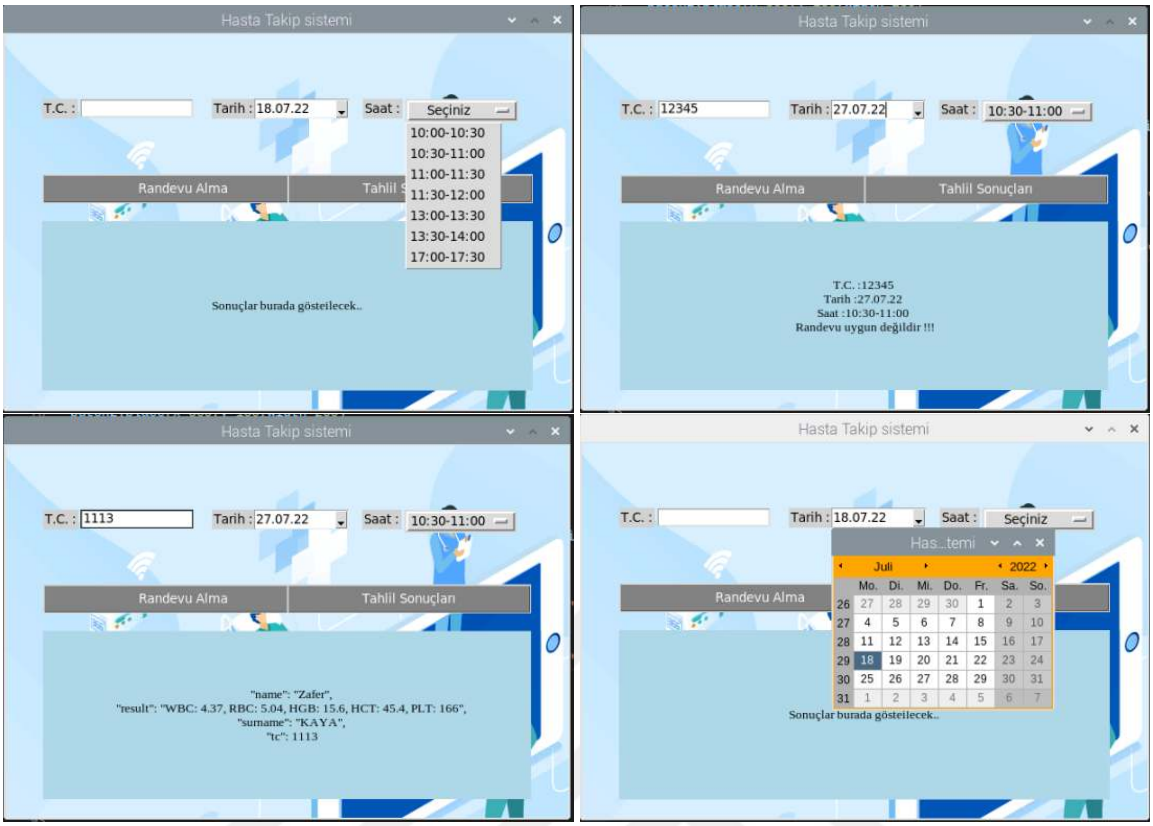


Şekil 4. 8. Tcpreplay ile eth0 üzerinde oluşturulan trafik

Tablo 4. 1. Devre kesici sonuçları

	5 Mbps		10 Mbps		15 Mbps		20 Mbps		25 Mbps		30 Mbps		35 Mbps		40 Mbps	
İstek Say.	Dev. Kes.	Zaman (ms)	Dev. Kes.	Zaman (ms)	Dev. Kes.	Zaman (ms)	Dev. Kes.	Zaman (ms)	Dev. Kes.	Zaman (ms)	Dev. Kes.	Zaman (ms)	Dev. Kes.	Zaman (ms)	Dev. Kes.	Zaman (ms)
1	0	21	0	22	0	22	0	24	0	19	0	26	0	22	0	26
5	0	33	0	34	0	23	0	27	0	34	0	28	0	27	0	30
10	0	39	0	33	0	42	0	32	0	35	0	40	0	39	0	36
50	0	41	0	46	0	39	0	34	0	39	0	46	0	47	1	44
100	0	45	1	42	1	44	1	42	0	46	1	42	1	47	1	44
150	1	46	0	40	3	47	2	42	1	44	0	46	1	44	2	44
200	1	49	0	48	3	47	2	41	4	46	1	46	0	41	1	44
300	4	50	2	43	1	43	1	42	2	42	2	43	3	44	2	46
400	4	55	3	45	3	44	2	44	2	46	3	46	2	44	1	43
500	2	46	6	45	5	43	0	43	3	39	5	45	1	44	1	45
600	4	49	4	44	4	43	4	43	4	45	4	45	3	46	2	44
700	5	43	3	43	7	46	2	43	5	44	2	38	1	36	2	45
800	1	36	9	45	4	42	3	43	4	44	2	43	6	45	6	45
900	2	38	3	42	7	44	7	43	6	51	7	47	5	45	4	44
1000	1	37	8	43	4	43	6	43	9	52	9	45	3	43	6	46

Tasarlanan bu hasta takip sisteminde hastaların randevu alma ve tahlil sonuçlarını görme işlemlerini kolay bir şekilde yapabilmeleri için hasta tarafındaki uç cihaz üzerinde uygulama arayüzü oluşturulmuştur. Oluşturulan arayüz Şekil 4.9'da gösterildiği gibidir. Hasta takip sistemindeki tüm mikroservisler konteyner olarak çalıştırılmaktadır. Docker Swarm ile konteynerlerin kontrolü sağlanmaktadır. Şekil 4.10'da Docker Swarm tarafından konteynerleri kontrol etmek için oluşturulan servisler gösterilmektedir.



Şekil 4. 9. Hasta uç cihazı (HUÜ) arayüzü

```
pi@pi4:~/Desktop/getpatient_container $ docker service ls
ID             NAME         MODE         REPLICAS   IMAGE                          PORTS
lii007j9m9r2   analysis     replicated   1/1        stasli/s_analysis:latest       *:3206->3206/tcp
hn1v32ht72x0   appointment  replicated   1/1        stasli/s_appointment:latest    *:3207->3207/tcp
q3llpeif5p3h   caching      replicated   1/1        stasli/s_caching:latest        *:3210->3210/tcp
ysmueahbggcv   circuit      replicated   1/1        stasli/s_circuit:latest        *:3212->3212/tcp
k13tonmftouw   emrgency     replicated   1/1        stasli/s_emergency:latest      *:3208->3208/tcp
zjd2mnwil42b   getpatient   replicated   1/1        stasli/s_getpatient:latest     *:3205->3205/tcp
jctpx1j3hg4t   logg         replicated   1/1        stasli/s_logg:latest           *:3211->3211/tcp
0z3ipa9f7fc5   management   replicated   1/1        stasli/s_management:latest     *:3209->3209/tcp
pi@pi4:~/Desktop/getpatient_container $
```

Şekil 4. 10. Docker swarm tarafından çalıştırılan servisler

5. SONUÇLAR

Bu çalışmada, bulut sistem temelli bir IoT hasta takip sistemi tasarımı önerilmiştir. Bu sistemin özgün yanı, bulut sistem yükünü düşürebilen ve sistem cevap süresini iyileştirebilen mikroservis yazılım mimarisi kullanan bir uç-bulut alt yapı mekanizmasını kullanmasıdır. Klasik IoT'lerde karşılaşılan büyük veri ve uzun cevap süresi problemleri dikkate alınarak bu sistemde veriler direkt olarak buluta aktarılmamış, ilk önce uç sistem üzerinde değerlendirmeler yapılmıştır. Bu sayede buluta gereksiz veri gönderiminin ve buluttan dönecek uzun cevap sürelerinin önüne geçilmiştir. Yapılan ölçümlerde, verilerin bulut üzerinde işlenmesinin yerine uç sistem üzerinde işlenmesiyle cevap süresinin ve bulutta oluşacak kaynak maliyetin önemli ölçüde azaldığı görülmüştür.

Bu çalışma hasta takip hizmetlerinde özellikle Aile Hekimlikleri ve onların sorumluluk alanlarında kullanılmasına yönelik yeni bir model önermektedir. Önerilen bu modelde mikroservis temelli uç/kenar sistem kullanılmış ve yönetilebilirlik konteyner teknolojileri ile sağlanmıştır. Bu teknik ile özellikle bulut tarafı kaynak kullanım maliyeti düşürülmüş ve sistemin daha performanslı çalışması sağlanmıştır. Sistemin geliştirilmesi iki safhada gerçekleştirilmiştir. Birinci safha, mikroservisler için hangi veri yönetim modelinin kullanılacağının test edilmesidir. Bu test sonucunda karar kılınan model ile ikinci safhada sistem yapısı oluşturulmuştur. Üç alt üniteden meydana gelen ana sistem, senkron ve asenkron çalışma modlarını yapısında bulundurmaktadır. Sistem, gerçek uç aygıtlar ve bulut teknolojileri üzerinde test edilmiştir. Farklı senaryolar için yapılan test sonuçları sistemin başarımını ispatlamış ve uygulanabilirliklerini göstermiştir.

Ülkelerin en çok harcama yaptıkları alanlardan biri de sağlık harcamalarıdır. Hasta takip sistemlerinin bir avantajı da bu alandaki maliyeti düşürmektir. Bu tez çalışması ile günümüzün önemli çalışma alanlarından olan IoT temelli sağlık sistemleri üzerine ekonomik ve sürdürülebilir bir sistem yaklaşımı sunulmuştur. Son yılların önemli yazılım mimarilerinden olan mikroservis temelli bir altyapıyı kullanan bu sistem, özellikle bulut entegrasyonuna ihtiyaç duyan alt sağlık birimleri için esnek bir çözüm sağlayabilir. Özellikle güvenlik, yetkilendirme mekanizmalarının katılması ile birlikte ticari potansiyele de sahip olacaktır. Yazarın bundan sonraki amacı, önerilen bu sistem üzerinden yapay zekâ temelli teşhis servislerinin sunulabilmesini sağlamak olacaktır.

KAYNAKLAR

[1] Ray, P. P., Dash, D., & De, D. (2021). Intelligent internet of things enabled edge system for smart healthcare. National Academy Science Letters, 44(4), 325-330.

[2] Gos, K., & Zabierowski, W. (2020, April). The comparison of microservice and monolithic architecture. In 2020 IEEE XVIth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH) (pp. 150-153). IEEE.

[3] Nadareishvili, I., Mitra, R., McLarty, M., & Amundsen, M. (2016). Microservice architecture: aligning principles, practices, and culture. " O'Reilly Media, Inc.".

[4] Von Leon, D., Miori, L., Sanin, J., El Ioini, N., Helmer, S., & Pahl, C. (2019). A lightweight container middleware for edge cloud architectures. Fog and edge computing: principles and paradigms, 145-170.

[5] Darshan K R and Anandakumar K R, "A comprehensive review on usage of internet of things (IOT) in healthcare system," 2015 International Conference on Emerging Research in Electronics, Computer Science and Technology (ICERECT), 2015, doi: 10.1109/erect.2015.7499001.

[6] S. Shah, R. Rutherford, and S. Menon, "Emerging technologies of IOT usage in global logistics," 2020 International Conference on Computation, Automation and Knowledge Management (ICCAKM), 2020, doi: 10.1109/iccakm46823.2020.9051530.

[7] M. Miller, in The internet of things: How smart tvs, Smart Cars, smart homes, and Smart Cities are changing the world, 2015.

[8] Routray, S. K., & Anand, S. (2017, February). Narrowband IoT for healthcare. In 2017 International Conference on Information Communication and Embedded Systems (ICICES) (pp. 1-4). IEEE.

[9] Mohammadzadeh, N., & Safdari, R. (2014). Patient monitoring in mobile health: opportunities and challenges. Medical Archives, 68(1), 57.

[10] Tan, J. (Ed.). (2008). Medical Informatics: Concepts, Methodologies, Tools, and Applications: Concepts, Methodologies, Tools, and Applications. IGI Global.

[11] Barjis, J., Kolfschoten, G., & Maritz, J. (2013). A sustainable and affordable support system for rural healthcare delivery. Decision Support Systems, 56, 223-233.

[12] Toledo, F. G., Triola, A., Ruppert, K., & Siminerio, L. M. (2012). Telemedicine consultations: an alternative model to increase access to diabetes specialist care in underserved rural communities. JMIR research protocols, 1(2), e2235.

[13] H. T. Yew, M. F. Ng, S. Z. Ping, S. K. Chung, A. Chekima, and J. A. Dargham, "IOT based real-time Remote Patient Monitoring System," 2020 16th IEEE International Colloquium on Signal Processing & Its Applications (CSPA), 2020, doi: 10.1109/cspa48992.2020.9068699.

[14] R. Kumar and M. P. Rajasekaran, "An IOT based patient monitoring system using Raspberry Pi," 2016 International Conference on Computing Technologies and Intelligent Data Engineering (ICCTIDE'16), 2016, doi: 10.1109/icctide.2016.7725378.

[15] T. Santra, "Mobile Health Care System for patient monitoring," Information and Communication Technologies, pp. 695–700, 2010, doi: 10.1007/978-3-642-15766-0_123.

[16] Greene, S., Thapliyal, H., & Carpenter, D. (2016, December). IoT-based fall detection for smart home environments. In 2016 IEEE international symposium on nanoelectronic and information systems (iNIS) (pp. 23-28). IEEE.

[17] Luo, X., Liu, T., Liu, J., Guo, X., & Wang, G. (2012). Design and implementation of a distributed fall detection system based on wireless sensor networks. EURASIP Journal on Wireless Communications and Networking, 2012(1), 1-13.

[18] Malasinghe, L. P., Ramzan, N., & Dahal, K. (2019). Remote patient monitoring: a comprehensive study. Journal of Ambient Intelligence and Humanized Computing, 10(1), 57-76.

[19] Sánchez-Gallegos, D. D., Galaviz-Mosqueda, A., Gonzalez-Compean, J. L., Villarreal-Reyes, S., Perez-Ramos, A. E., Carrizales-Espinoza, D., & Carretero, J. (2020). On the continuous processing of health data in edge-fog-cloud computing by using micro/nanoservice composition. IEEE Access, 8, 120255-120281.

[20] Ianculescu, M., Alexandru, A., Neagu, G., & Pop, F. (2019, November). Microservice-based approach to enforce an IoHT oriented architecture. In 2019 E-Health and Bioengineering Conference (EHB) (pp. 1-4). IEEE.

[21] Lin, D., & Tang, Y. (2019). Edge Computing-Based Mobile Health System: Network Architecture and Resource Allocation. IEEE Systems Journal, 14(2), 1716-1727.

[22] Gómez, J., Oviedo, B., & Zhuma, E. (2016). Patient monitoring system based on internet of things. Procedia Computer Science, 83, 90-97.

[23] Salehi, S., Olyaeemanesh, A., Mobinizadeh, M., Nasli-Esfahani, E., & Riazi, H. (2020). Assessment of remote patient monitoring (RPM) systems for patients with type 2 diabetes: a systematic review and meta-analysis. Journal of Diabetes & Metabolic Disorders, 19(1), 115-127.

[24] Várady, P., Benyó, Z., & Benyó, B. (2002). An open architecture patient monitoring system using standard technologies. IEEE transactions on information technology in biomedicine, 6(1), 95-98.

[25] Uddin, M. S., Alam, J. B., & Banu, S. (2017, September). Real time patient monitoring system based on Internet of Things. In 2017 4th International conference on Advances in Electrical Engineering (ICAEE) (pp. 516-521). IEEE.

[26] Suh, M. K., Chen, C. A., Woodbridge, J., Tu, M. K., Kim, J. I., Nahapetian, A., ... & Sarrafzadeh, M. (2011). A remote patient monitoring system for congestive heart failure. Journal of medical systems, 35(5), 1165-1179.

[27] Sebastian, S., Jacob, N. R., Manmadhan, Y., Anand, V. R., & Jayashree, M. J. (2012). Remote patient monitoring system. International Journal of Distributed and Parallel Systems, 3(5), 99.

[28] Gokhale, P., Bhat, O., & Bhat, S. (2018). Introduction to IOT. International Advanced Research Journal in Science, Engineering and Technology, 5(1), 41-44.

[29] https://www.oracle.com/tr/internet-of-things/what-is-iot/ , Erişim:25 Mart 2022

[30] https://www.investopedia.com/articles/personal-finance/090715/8-best-cloud-storage-solutions-small-business.asp, Erişim:29 Mart 2022

[31] S. K. Mishra, B. Sahoo, and P. P. Parida, “Load balancing in cloud computing: A big picture,” Journal of King Saud University - Computer and Information Sciences, vol. 32, no. 2, pp. 149–158, 2020.

[32] B. Chen, J. Wan, A. Celesti, D. Li, H. Abbas and Q. Zhang, " Edge Computing in IoT-Based Manufacturing," in IEEE Communications Magazine, vol. 56, no. 9, pp. 103-109, Sept. 2018.

[33] F. Ponce, G. Marquez, and H. Astudillo, “Migrating from monolithic architecture to microservices: A rapid review,” 2019 38th International Conference of the Chilean Computer Science Society (SCCC), 2019.

[34] K. Shafique, B. A. Khawaja, F. Sabir, S. Qazi, and M. Mustaqim, “Internet of things (IOT) for next-generation smart systems: A review of current challenges, future trends and prospects for emerging 5G-IOT scenarios,” IEEE Access, vol. 8, pp. 23022–23040, 2020.

[35] N. Niknejad, W. Ismail, I. Ghani, B. Nazari, M. Bahari, and A. R. Hussin, “Understanding service-oriented architecture (SOA): A systematic literature review and directions for further investigation,” Information Systems, vol. 91, p. 101491, 2020.

[36] X. Larrucea, I. Santamaria, R. Colomo-Palacios and C. Ebert, " Microservices," in IEEE Software, vol. 35, no. 3, pp. 96-100, May/June 2018.

[37] M. Keen, Patterns implementing an SOA using an enterprise service bus. Research Triangle Park, NC: IBM, International Technical Support Organization, 2004.

[38] M. Massé, REST API design rulebook designing consistent restful web service interfaces. Beijing u.a.: O'Reilly, 2012.

[39] G. Brito, T. Mombach, and M. T. Valente, “Migrating to graphql: A practical assessment,” 2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER), 2019.

[40] “Performance analysis of computational offloading on ...” [Online]. Available: https://hal.archives-ouvertes.fr/hal-02495252/document. [Accessed: 01-Feb-2022].

[41] D. von Leon, L. Miori, J. Sanin, N. El Ioini, S. Helmer, and C. Pahl, “A lightweight container middleware for Edge Cloud Architectures,” Fog and Edge Computing, pp. 145–170, 2019.

[42] Bhatia, G., Choudhary, A., & Gupta, V. (2017). THE ROAD TO DOCKER: A SURVEY. International Journal of Advanced Research in Computer Science, 8(8).

[43] https://phoenixnap.com/blog/kubernetes-vs-docker-swarm, Erişim:5 Mart 2022.

[44] https://blog.newrelic.com/wp-content/uploads/kubernetes_architecture.jpg, Erişim: 15 Mart 2022.

[45] Tcpreplay. [Online]. Available: https://tcpreplay.appneta.com/. [Accessed: 05-Feb-2022].

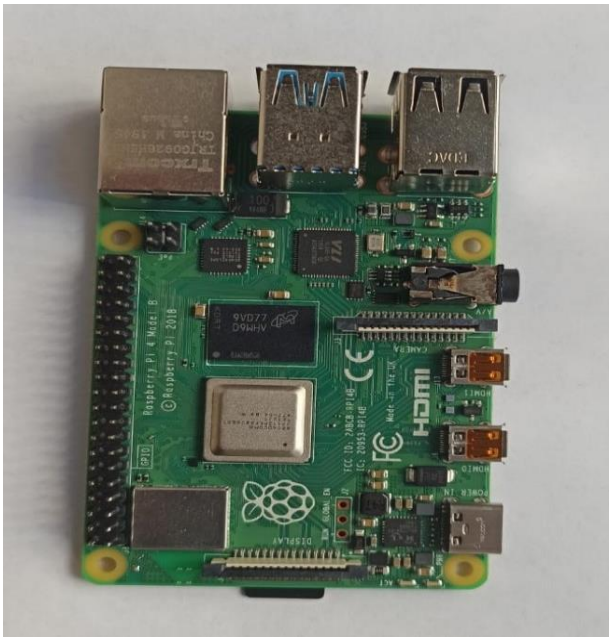
[46] “Zabbix features overview,” Zabbix. [Online]. Available: https://www.zabbix.com/features. [Accessed: 07-Feb-2022].

[47] Taşlı, S, Yıldırım, G. Sağlık Hizmetlerinde Kullanılan Mikroservis Mimarisi Temelli Kenar Sistem Çözümleri, Tashkent 1st-International Congress on Modern Sciences,2022.

EKLER

EK- 1: RASPBERRY Pİ

Raspberry Pi, kullanıcıları bilişim konusunda geliştirmeyi ve bilgisayar eğitimine daha kolay ulaşım sağlamayı hedefleyen tek karta sahip bir bilgisayardır. Raspberry Pi, kablo vasıtasıyla bir bilgisayar ekranına veya TV'ye takılabilen ve normal bir klavye ve fare kullanabilen bir bilgisayara göre oldukça düşük maliyetli, ebat olarak bir kredi kartı boyutunda olan bilgisayardır. Her yaştan kullanıcının bilgisayara ulaşmasını ve Scratch ve Python gibi yazılım dillerinde kendilerini geliştirmelerine olanak tanıyan bir cihazdır. Normal bir bilgisayarla yapılabilen internette gezinmek, video izlemek, elektronik tablolar oluşturmak, kelime işleme ve oyun oynama gibi çoğu işlemleri yapabilir. Raspberry Pi, bir bilgisayarın ihtiyaç duyduğu donanımların hemen hemen hepsine sahiptir. Raspberry Pi her ne kadar çocukların programlama yetkinliklerini artırmaya yönelik olarak çıkarılmış olsa da günümüzde birçok farklı alanlarda kullanılmaktadır. Dünya genelinde kullanıcılar, Raspberry Pi'yi programlama yeteneklerini geliştirmek, donanım uygulamaları oluşturmak, akıllı ev sistemleri yapmak ve bunların dışında endüstriyel ve akademik çalışmaları gerçekleştirmek için kullanmaktadırlar. Raspberry Pi, linux tabanlı ve maliyet olarak oldukça ucuz bir bilgisayardır. Buna rağmen fiziksel projeler için elektronik araçların kontrol edilmesini ve IoT alanında oldukça yaygın kullanılan bir dizi GPIO (genel amaçlı giriş / çıkış) pinine sahiptir. Şekil E1.1'de Raspberry Pi 4 Model B serisi gösterilmektedir.



Şekil E1. 1. Raspberry pi 4 model b

Şekil E1.1'de görüldüğü üzere Raspberry Pi'nin bir bilgisayar olarak kullanılabilmesi için üzerinde gerekli donanımların bağlanabileceği girişler bulunmaktadır. Raspberry Pi'yi kullanabilmek için klavye, fare, SD kart ve Raspberry Pi'nin arayüzünü görebilmek için hdmı kablosuyla bir monitöre ihtiyaç vardır. Raspberry Pi'nin yerel depolama alanı yoktur. Raspberry Pi'de SD kart yongası vardır. Depolama işlemleri bu SD kart üzerinden yapılmaktadır. Tablo E1.1'de Raspberry Pi 4 Model B'ye ait teknik özellikler sunulmaktadır.

Tablo E1. 1. Raspberry pi 4 model b teknik özellikleri

İşlemci	Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
Hafıza	4GB LPDDR4
Bağlantı	2.4 GHz ve 5.0 GHz IEEE 802.11b/g/n/ac wireless LAN, Bluetooth 5.0 BLE Gigabit Ethernet 2 × USB 3.0 port 2 × USB 2.0 port
Pinler	Standart 40-pin GPIO başlığı
Video & Ses	2 × micro HDMI port (4Kp60'a kadar desteklenir) 2-lane MIPI DSI ekran portu 2-lane MIPI CSI kamera portu 4 kutuplu stereo ses ve bileşik video bağlantı noktası
Multimedya	H.265 H.264 OpenGL ES, 3.0 grafik
SD Kart Desteği	İşletim sistemini ve veri depolamayı yapmak için Micro SD kart yuvası vardır
Giriş Gücü	USB-C konnektör üzerinden 5V DC (minimum 3A) GPIO başlığı üzerinden 5V DC (minimum 3A) İnternet üzerindeki güç (PoE)
Çalışma Sıcaklığı Aralığı	0–50°C arasında çalışabilir

ÖZGEÇMİŞ

Sinan TAŞLI

[redacted]