

T.C.
EGE ÜNİVERSİTESİ
Fen Bilimleri Enstitüsü

YALIN BİR NESNELERİN İNTERNETİ PLATFORMU GERÇEKLEŞTİRİMİ

Gizem ÇELEBİ ŞAHİN

Danışman : Doç. Rıza Cenk ERDUR

Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Mühendisliği Yüksek Lisans Programı

İzmir
2019

Gizem Çelebi Şahin tarafından yüksek lisans tezi olarak sunulan “Yalın Bir Nesnelerin İnterneti Platformu Gerçekleştirimi” başlıklı bu çalışma EÜ Lisansüstü Eğitim ve Öğretim Yönetmeliği ile EÜ Fen Bilimleri Enstitüsü Eğitim ve Öğretim Yönergesi’nin ilgili hükümleri uyarınca tarafımızdan değerlendirilerek savunmaya değer bulunmuş ve 3.09.2019 tarihinde yapılan tez savunma sınavında aday oybirliği/oyçokluğu ile başarılı bulunmuştur.

Jüri Üyeleri:

Jüri Başkanı : Doç. Dr. Rıza Cenk ERDUR
Raportör Üye : Dr. Öğr. Üyesi Emine SEZER
Üye : Dr. Öğr. Üyesi Övünç ÖZTÜRK

İmza

.....
.....
.....



EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ**ETİK KURALLARA UYGUNLUK BEYANI**

EÜ Lisansüstü Eğitim ve Öğretim Yönetmeliğinin ilgili hükümleri uyarınca Yüksek Lisans Tezi olarak sunduğum “Yalın Bir Nesnelerin İnterneti Platformu Gerçekleştirimi” başlıklı bu tezin kendi çalışmam olduğunu, sunduğum tüm sonuç, doküman, bilgi ve belgeleri bizzat ve bu tez çalışması kapsamında elde ettiğimi, bu tez çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara atıf yaptığımı ve bunları kaynaklar listesinde usulüne uygun olarak verdiğimi, tez çalışması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını, bu tezin herhangi bir bölümünü bu üniversite veya diğer bir üniversitede başka bir tez çalışması içinde sunmadığımı, bu tezin planlanmasından yazımına kadar bütün safhalarda bilimsel etik kurallarına uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul edeceğimi beyan ederim.

03 / 09 / 2019

İmzası

Adı-Soyadı

Gizem ÇELEBİ ŞAHİN



ÖZET**YALIN BİR NESNELERİN İNTERNETİ
PLATFORMU GELİŞTİRİMİ**

ÇELEBİ ŞAHİN, Gizem

Yüksek Lisans Tezi, Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı: Doç. Dr. Rıza Cenk ERDUR

Eylül 2019, 87 sayfa

Nesnelerin interneti (Internet of Things), gelişen iletişim teknolojileri ile birlikte gün geçtikçe daha çok hayatımızda yer almaya başlamış olan bir kavramdır. Akıllı cihazlar, ortam algılayıcılar, kablosuz bağlantı teknolojileri bu alandaki gelişmeleri hızlandırmaktadır. Genel anlamıyla somut dünyada var olan tüm nesnelerin ve kavramların, dijital dünya üzerinde tasvir edilebilmesi ve bu yapıların birbirleri ile iletişime ve etkileşime girebilmesi, çeşitli servisler aracılığı ile yönetilebilmesi ihtiyacı ile birlikte nesnelerin interneti platformlarına ihtiyaç duyulmuştur. Çalışmada literatürde yer alan nesnelerin interneti platformu mimari model yaklaşımları ve aktif olarak kullanılmakta olan platform çözümlerinin mimari modelleri incelenmiştir. Bu model örneklerinden yola çıkılarak bir nesnelerin interneti platformunda mutlaka olması gereken katmanlar belirlenmiş, yalın bir çekirdek mimari oluşturulmuştur. Yalın çekirdek model yapısı ile gerçek hayatta çözüm isteğinde bulunabilecek iki farklı senaryo için iki farklı mimari opsiyon sunulmuştur. Yalın model ile küçük ölçekli çözümler yapılabileceği gibi büyük ölçekli platform sağlayıcılar için de kullanılabilir. Önerilen yalın model üzerine yapılan eklemeler ve yalın modelin farklı ölçeklerde kullanılırken sağladığı avantajlardan bahsedilmiştir.

Oluşturulan yalın model baz alınarak, açık kaynak kodlu node.js tabanlı Sails geliştirme ortamı ve MongoDB teknolojileri kullanılarak bir platform prototipi oluşturulmuştur. Prototipteki yapıların modeldeki hangi katmanlara karşılık geldiği belirtilmiştir. Geliştirme yapılırken kullanıcıların hızlı bir şekilde platforma cihazlarını tanımlayabilmeleri için ek servisler sağlanmıştır. İhtiyaçlar doğrultusunda yalın modelin hangi modüller ile genişletilebileceği, açık kaynak kodlu sistemler üzerinden örnekler ile belirtilmiştir.

Çalışmanın sonunda prototip platformda senaryo gerçekleştirmeleri yapılmıştır. Son bölümünde yalın yapının diğer ürünlerin modellerine göre nasıl konumlandığına ve avantajlarının neler olduğuna değinilmiştir. Ayrıca platform prototipinin farklı alanlarda yapılabilecek geliştirme önerilerine de değinilmiştir.

Anahtar Sözcükler: Nesnelerin interneti, nesnelerin interneti yalın mimari model, nesnelerin interneti platformu, akıllı çözümler.



ABSTRACT

LIGHTWEIGHT INTERNET OF THINGS

PLATFORM IMPLEMENTATION

ÇELEBİ ŞAHİN , Gizem

MSc in Computer Eng.

Supervisor: Assoc. Prof. Dr. Rıza Cenk ERDUR

Sep 2019, 87 pages

The Internet of Things is a concept which has started to take place in our lives with the developing communication technologies. Smart devices, sensors, wireless connection technologies accelerate the developments in this field. In general, all objects and concepts that exist in the concrete world can be depicted on the digital world and these structures can communicate and interact with each other, and the need to manage abstract objects through various services through internet platforms become necessary. In this study, the architectural model approaches of the internet of things platforms in the literature and the architectural models of the active platform solutions are examined. Based on these model examples, the layers that must be on the Internet of things platforms have been determined and a simple core architecture has been created. Two different architectural options are presented for two different scenarios that can request a real life solution with its simple core model structure. The additions on the proposed lean model and the advantages of the lean model when used at different scales are mentioned.

Based on the lean model, a platform prototype has been created using the open source node.js based sails development environment and MongoDB technologies. Which layers in the model correspond to the structures of the prototype has been explained. During the development, additional services have been provided to enable users to quickly onboard their devices to the platform. The modules with which the lean model can be expanded according to the needs are specified with open source options.

At the end of the study, prototype scenarios have been developed based on the lean model. Thus, the desired functionalities are realized with a simple platform. In the last part, how the lean structure is positioned according to the models of other products and the advantages are explained. In addition, the development proposals has been given for different areas of the platform realization.

Keywords: Internet of things, internet of things lightweight architecture, internet of things platform architecture, smart solutions.

ÖNSÖZ

Tez konusu, nesnelerin interneti çözümleri ve platformları kullanılırken ortaya çıkmıştır. Varolan platformların özellikleri ve sektörde çözüm bekleyen firmaların istekleri incelendiğinde bir yalın platform oluşturma gereksinimi görülmüştür. Literatür araştırması yapılmıştır. Derlemeler ve yalın bir platform oluşturmak için yapılmış olan çalışmalar incelenmiştir. Bu incelemeler ışığında platformun sahip olması gereken fonksiyonel yapı belirlenmiş ve katmanlar belirlenmiştir. Uygun teknolojiler ile bu katmanların geliştirimi yapılmıştır.

Yalın model hem çözüm odaklı hem de servis sağlayıcısı odaklı 2 farklı bakış açısı için örnek senaryolar ile desteklenmiştir.

Prototip bulut üzerine kurulmuş, örnek bir senaryo ile denenmiştir.

Çalışma bu haliyle çekirdek fonksiyonalliteye sahip, istenildiğinde genişletilebilen ve farklı modüllerle entegre edilebilen, hem bulut hem lokal sunuculara kurulabilen bir sistemdir.

İzmir

03/09/2019

Adı-Soyadı

Gizem ÇELEBİ ŞAHİN



İÇİNDEKİLERSayfa

İÇ KAPAK.....	ii
KABUL ONAY SAYFASI.....	iii
ETİK KURALLARA UYGUNLUK BEYANI	v
ÖZET.....	vii
ABSTRACT.....	ix
ÖNSÖZ	xi
ŞEKİLLER DİZİNİ.....	xix
TABLolar DİZİNİ	xxiii
KISALTMALAR DİZİNİ.....	xxv
1 GİRİŞ	1
2 GENEL BİLGİLER.....	3
2.1 Nesnelerin İnterneti Kavramının Gelişmesi	3
2.1.1 Radyo Frekans ile Tanımlama Teknolojisi (RFID).....	3
2.1.2 Makinalar Arası İletişim (M2M)	3
2.1.3 Nesnelerin İnterneti	5
2.1.4 Nesnelerin İnternetinin Geleceği.....	7

İÇİNDEKİLER (devam)

Sayfa

2.2	Literatür Taraması	8
2.2.1	IoT6 Referans Mimarisi	8
2.2.2	Açık Kaynak Kodlu Yalın bir Nesnelerin İnterneti Platformu ile Bağlam Bilinçli Uygulamalar Geliştirme	10
2.2.3	Akıllı Ev Sistemleri için Nesnelerin İnterneti	11
2.2.4	Nesnelerin İnterneti Derlemesi.....	11
2.2.5	Nesnelerin İnterneti için Yalın Mimari Gerçekleştirimi	12
2.2.6	Mobil Uygulama ile Yalın bir Nesnelerin İnterneti Platformu Sağlama	12
2.2.7	Nesnelerin İnterneti Platform Mimarilerinin Karşılaştırılması	13
2.3	Market Araştırması.....	14
2.3.1	IBM Watsons.....	14
2.3.2	Amazon AWS IoT Suite	15
2.3.3	Azure IoT	16
2.3.4	Thingworx	17
3	PLATFORM ÖNERİSİ.....	19
3.1	Platform Önerisinin Katman Yapısı	20

İÇİNDEKİLER (devam)

Sayfa

3.1.1	Entegrasyon katmanı	20
3.1.1.1	Http / Https	20
3.1.1.2	Mqtt	21
3.1.1.3	Coap	22
3.1.2	Veri katmanı	22
3.1.3	Yönetim katmanı	22
3.1.4	Çalıştırma katmanı	23
3.2	Önerilen Platformun Özellikleri ve Mimarisi	23
3.2.1	Platform entegrasyon katmanı.....	24
3.2.2	Platform veri katmanı.....	25
3.2.3	Platform yönetim katmanı	25
3.2.4	Platform çalıştırma katmanı	26
3.3	Platform Mimarisi ile Farklı Kurulum Opsiyonları	26
3.4	Platform Gerçekleştirimi	28
3.4.1	Kullanılan teknolojiler.....	28
3.4.1.1	Node.js	28

İÇİNDEKİLER (devam)

Sayfa

3.4.1.2	Sails Yazılım Geliştirme Ortamı	29
3.4.1.3	MongoDB.....	29
3.4.2	Platform Yazılımının İncelenmesi.....	31
3.4.2.1	Kontrolcüler	32
3.4.2.2	Modeller	37
3.5	Platform Genel Programlama Arayüzü Yazılımı	38
3.6	Platformun Test edilmesi ve Çıktıları	41
3.6.1	Kullanıcı Yaratma	41
3.6.2	Kullanıcıya ait Veritabanı Anahtarı Alma.....	43
3.6.3	Standart Prosedür ile Cihaz Ekleme.....	44
3.6.4	Programlama Arayüzü Başlatma.....	44
3.6.5	Özelleşmiş Programlama Arayüzü Yükleme	45
3.6.6	Programlama Arayüzünü Durdurma	46
3.6.7	Uygulama Yükleme.....	46
3.6.8	Kullanıcı Sil/Hesabı Dondurma	47
4	DURUM ÇALIŞMASI	48

İÇİNDEKİLER (devam)Sayfa

4.1	Atık Su Arıtma Tesisi Teknik Gezisi	48
4.2	Platform Kullanmanın Avantajları	49
4.3	Platformun Kullanılması	49
5	SONUÇ VE TARTIŞMA.....	52
6	ÖNERİLER	56
	KAYNAKLAR DİZİNİ	57
	KAYNAKLAR DİZİNİ (devam)	58
	TEŞEKKÜR.....	60
	ÖZGEÇMİŞ	61



ŞEKİLLER DİZİNİ

<u>Şekil</u>	<u>Sayfa</u>
2.1 Makinalar arası iletişimin ülkemizde kullanım alanları (BTK, 2013)	4
2.2 Dünya genelinde nesnelerin interneti trendi (Google Trends, 2019)	4
2.3 Nesnelerin interneti kavramının bileşenleri (Atzori,2010).....	5
2.4 IoT6 mimari modeli (Krcro et al., 2014)	9
2.5 Fernando ve arkadaşlarının mimari önermesi (Ribeiro, F. and Metrolho, J.,2016)	10
2.6 Flip mimari modeli (Malche, T. and Maheshwary, P.,2017)	11
2.7 Yalın nesnelerin interneti platformu mimarisi (Heo, Y.J, et. al., 2015)	12
2.8 Guth ve arkadaşlarının referans mimari modeli (Guth, J., et.al., 2016)	13
2.9 IBM Watsons nesnelerin interneti platform mimarisi (IBM Watsons, 2019)	15
2.10 Amazon nesnelerin interneti platform Mimarisi (Amazon AWS, 2019)	16
2.11 Azure IoT referans mimari modeli (Windows Azure, 2019)	16
2.12 Thingworx mimari modeli (Thingworx Case Study, 2019)	18
3.1 Platform Katmanları.....	20
3.2 MQTT protokolü mesajlaşma şeması (Mqtt Protokolü, 2019)	21
3.3 Konteyner yapısı (Docker, 2019)	23
3.4 Platform mimarisi.....	24

ŞEKİLLER DİZİNİ (devam)

<u>Şekil</u>	<u>Sayfa</u>
3.5 Opsiyon 1- Tekil kurulum modeli	27
3.6 Opsiyon 2- Servis sağlayıcı kurulum modeli	28
3.7 Yönetici Yazılım proje yapısı	32
3.8 Dizin yapısı	33
3.9 Yazılımdaki kontrolcüler ile URL eşlemeleri	38
3.10 Cihaz ekleme yöntemi opsiyonları	38
3.11 Cihaz verisini platforma gönderme servisi.....	39
3.12 Son cihaz verisini veritabanından alma.....	39
3.13 Belirli miktarda kaydı alma servisi	40
3.14 Belirli tarih aralığındaki verileri alma servisi.....	40
3.15 Platform kullanımı akış diyagramı	41
3.16 Kullanıcı ekleme servisi testi	42
3.17 Kullanıcı ekleme servisi hata mesajı.....	42
3.18 Yönetici veritabanına kullanıcı eklenmesi	42
3.19 Başarılı senaryo sonrası yaratılan veritabanı.....	43
3.20 Erişim anahtarı alma-Başarılı senaryo	43

ŞEKİLLER DİZİNİ (devam)

<u>Şekil</u>	<u>Sayfa</u>
3.21 Erişim anahtarı alma-Hatalı senaryo	43
3.22 Cihaz Ekleme-Başarılı senaryo	44
3.23 Programlama arayüzü başlatma	45
3.24 Programlama arayüzü yükleme	45
3.25 Programlama arayüzü durdurma	46
3.26 Uygulama yükleme servisi	46
3.27 Kullanıcı hesabının silinmesi/dondurulması	47
3.28 Kullanıcının veritabanındaki durumu.....	47
4.1 Atıksu arıtma süreci (Hidro dizayn, 2019).....	48



TABLÖLAR DİZİNİTabloSayfa

Tablo 3.1 MongoDB terimler eşlemesi 30

Tablo 5.1 Platformların Karşılaştırılması..... 52





KISALTMALAR DİZİNİ

Kısaltmalar

3PP	Third Party Product
4G	4th Generation
5G	5th Generation
ACK	Acknowledgement
API	Application Programing Interface
APP	Application
aaS	As a Service
BSN	Binary JSON
BTK	Bilgi Teknolojileri Kurumu
COAP	Constrained Application Platform
CPU	Central Processing Unit
DDOS	Distributed Denial of Service
ETSI	European Telecommunications Standards Institute
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Secure
IoT	Internet of Things

KISALTMALAR DİZİNİ (devam)Kısaltmalar

IPSO	IP for Smart Objects
Ipv4	Internet Protocol Version 4
Ipv6	Internet Protocol Version 6
ITU	International Telecommunication Union
İzsu	İzmir Su ve Kanalizasyon İdaresi
JSON	Javascript Object Notation
LPWAN	Low Power Wide Area Network
M2M	Machine to Machine
MQTT	Message Queuing Telemetry Transport
MVC	Model View Controller
NFC	Near Field Communication
NoSQL	Not Only SQL
ORM	Object Relational Mapping
PLC	Programmable Logic Controller
Pub/Sub	Publish/Subscribe
REST	Representational State Transfer

KISALTMALAR DİZİNİ (devam)Kısaltmalar

RFID	Radio Frequency Identification
SCADA	Supervisory Control and Data Acquisition
SDK	Software Development Kit
SQL	Standardized Query Language
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
WIFI	Wireless Fidelity
WISP	Wireless Identification and Sensing Platform



1 GİRİŞ

Nesnelerin İnterneti (Internet of Things- IoT) çeşitli ortamlarda bulunan cihazların birbirleri ile bağlantılı oldukları çözümler bütünüdür. Cihaz ve sensör çeşitlilikleri, çözüm önerileri, bağlantı teknolojileri her geçen gün gelişmekte ve kullanılabilirliği artmaktadır. Nesnelerin İnterneti kavramının son yıllarda popülerliğinin artmasıyla beraber çeşitli alanlarda geliştirilmiş uygulamalar hayatımıza girmeye başlamıştır. Akıllı binalardan, akıllı trafik sistemlerine, sınır güvenliğinden, sağlık sistemlerine, akıllı sayaçlara kadar çok farklı alanlarda her gün yeni çözümler üretilmektedir. Bu çözümler uygulamaya özgü yada farklı platform sağlayıcılarının servisleri ile geliştirilmiş olabilir. Farklı modüllerle zenginleştirilmiş olan platformlar bulunmaktadır. Bu platformları gereksinimlere göre pozisyonlandırmak belirli bir alan bilgisi gerektirmektedir. Varolan platform çözümleri incelendiğinde büyük bir bölümünün bulut çözümleri sağlayıcı şirketlerin çözümleri olduğu görülmektedir. Bulut sistemlerinde sunulan servisler müşterilerin veri güvenliği konusunda çekimser kalmasına sebep olmaktadır. Hem kurulum konusunda bulut bağımlılığı bulunmayan, hem de küçük veya büyük ölçekli çözümler için yalınlığı koruyan bir model oluşturmak amaçlanmaktadır.

Tez çalışmasında öncelikle mimari olarak temel fonksiyonaliteyi sağlamak amaçlanmıştır. Literatürdeki derlemeler ve varolan sıklıkla kullanılan platform çözümleri incelenmiştir. Bir platformun sahip olması gereken temel katmanlar ne olmalıdır sorusuna cevap aranmıştır. Bu sayede kolay anlaşılır yalın bir çekirdek yapı oluşturulabilecektir. Bu modelin gerçekleştirimi yapılırken kullanım ve fonksiyonalite açısından kullanıcılara hızlı bir başlangıç yapmalarını sağlamak için bazı temel fonksiyonaliteler hazırlanmıştır. Ayrıca 3. Parti sistemlerle entegrasyonun da kolay olması amaçlanmıştır.

Literatürde yapılmış olan çeşitli platform mimarisi yaklaşımları incelenmiştir. Guth ve arkadaşlarının ihtiyaca uygun bir platform seçmeyi kolaylaştırmak ve var olan platformlar arasında karşılaştırma yapmak için referans bir mimari model oluşturmaya çalışmışlardır. (Guth et al., 2016) Al Qaseemi ve arkadaşlarının çalışmasında ise IoT alanındaki çalışmaların hızlanabilmesi için standartlaştırmanın gerekliliğinden bahsedilmiştir. Var olan ticari platform çözümlerinden birinin standart sağlayıcı olarak baz alınabileceğini savunmuşlardır. (Al-Qaseemi, 2016) Heo ve arkadaşları yalın bir mimari model ile platform örneklemesi yapmışlardır. (Heo et al., 2015) Arka plan servislerinde kullanılan teknolojiler benzerlik gösterse de cihaz yönetimi için platform yaklaşımı çeşitli servisler ile genişletilebilir olması ve iş modeli bakımından mimari önermesi farklılık göstermektedir.

Akıllı kompleks cihazlardan basit ölçümler yapabilen düşük maliyetli sensörlere kadar farklı türdeki cihaz ve sensörlerin günlük hayatta kullanımının artmasıyla öncelikle dikey yönde özelleşmiş çözümler yapılmıştır. 2010'lu yılların başında M2M yani "Makinalar Arası İletişim" olarak başlayan teknolojiler sistemi zaman içerisinde bağlantı teknolojilerinin, bulut çözümlerinin ve büyük veri analiz teknolojilerinin gelişmesi ile "Nesnelerin İnterneti" kavramına evrilmiştir. Nesnelerin interneti kavramının somutlaştırıldığı, farklı türde cihaz verilerinin ortak bir ortamda

toplanıldığı ve yönetildiği; aynı zamanda kökten uca çözüm oluşturmak için bir ortam sağlayan platform yapıları oluşturulmuştur. Günümüzde yaygın olarak kullanılan IoT platformların çoğu (Microsoft Azure IoT Suite, Amazon AWS, Thingworx) yatayda uzanan, her tip kullanım senaryosuna uyum sağlayan iş modeline sahip çözümler sunmaktadır. Bu platformlar incelendiğinde her kullanım senaryosu ve büyüklüğe uygunluk için bazı dezavantajlara sahiptir. Mimari modelleri 4. bölümde incelendiği üzere çoğunlukla karmaşıktır. Aynı zamanda ortak bir model yaratma kaygısı çeşitli kısıtlamaları beraberinde getirmiştir. İş modellerindeki çeşitli yan ürün entegrasyonları da platform maliyetlerini arttırmaktadır, bu da IoT çözümlerinin gerçekleştiriminde sektörün çekimser kalmasına neden olmaktadır.

Platformun gerçekleştiriminden önce ilk olarak mimari model belirlenmiştir. İhtiyaçlar doğrultusunda genişletilebilecek merkezi bir referans mimari model ve farklı durumlar için farklı entegrasyonları ve genişletilmelere elverişli olan bir iş modeli mimarisi oluşturulmuştur. Kullanılabilecek teknolojiler araştırılıp uygun olanlar seçilmiştir. Çekirdek fonksiyonlitesinin gerçekleştirimi için arka plan yönetim servisleri oluşturulmuştur. Bu fonksiyonlilerin kullanılabileceği kullanımı kolay, platforma cihaz tanımlamayı kolaylaştıran servisler geliştirilmiştir. Ayrıca geliştiriciler özelleşmiş fonksiyonlar için uygulama geliştirme yapabilmekte ve bunları platforma yükleyip internete açabilmeleri amaçlanmıştır.

Oluşturulmuş olan platform hem bulut üzerinde hem de istenildiği zaman özel sunucularda konuşlandırılabilir. Bu yönüyle de var olan bulut tabanlı nesnelerin interneti platformlarından farklılık göstermektedir. Bulut servislerine bağımlılığı bulunmamaktadır. İstenildiği takdirde büyük veri ve çeşitli kimlik kanıtlama sistemleri ile bağlantılanabilir. Ölçeklendirme ve çeşitli çözümler için farklı kurulumlar yapılabilir. 5.3 nolu bölümde bunlara değinilecektir.

Tezde ilk olarak nesnelerin interneti kavramına tarihsel bir bakış yapılarak, teknolojinin nasıl bu günlere evrildiğini ve ileride neler getirebileceğinden bahsedilecektir. Sonrasında nesnelerin interneti ile duymaya alıştığımız ve platform yetkinliklerini de anlatırken bilinmesi gereken protokoller ve bağlantı teknolojileri tanıtılacaktır. Tezin gerçekleştirim kısmında platform mimarisinden ve kullanılan teknolojilerden bahsedilecektir. Platformun yazılım detaylarından bahsedilecek ve oluşturulmuş servislerin test edilmesine yer verilecektir. Son bölümde ise platformun bulut üzerindeki kurulmuş versiyonu ile örnek kullanım senaryosu üzerinde denenmesi yapılacaktır.

2 GENEL BİLGİLER

Bu bölümde ilk olarak 2.1’de nesnelerin interneti kavramının tarihsel süreçte nasıl oluştuğu inceleniyor. Bölüm 2.2 ve 2.3’te nesnelerin interneti ile ilgili yapılmış olan literatür ve market araştırmasına ait incelemeler yer almaktadır. Tez çalışması ile benzerlik gösteren çalışmaların benzer ve farklı yönleri vurgulanmıştır.

2.1 Nesnelerin İnterneti Kavramının Gelişmesi

2.1.1 Radyo Frekans ile Tanımlama Teknolojisi (RFID)

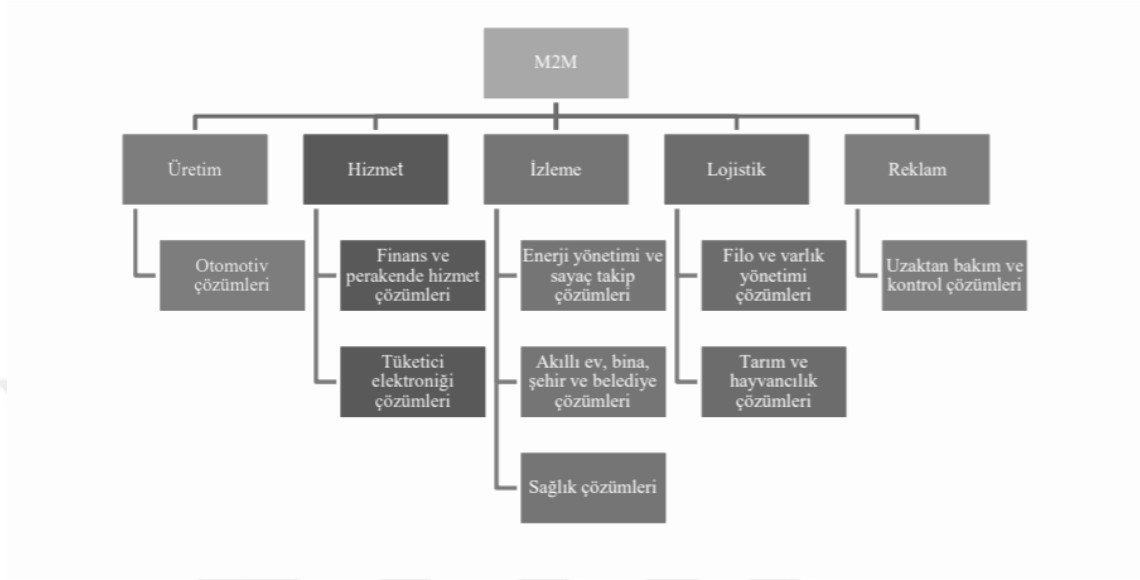
1948 yılında Silver ve Woodland in barkod sistemlerini geliştirmesiyle RFID teknolojisinin temelleri atılmış oldu, 1952 yılında patentini aldıkları bu teknoloji 20. Yüzyılın en önemli icatlarından olmuştur. Barkod sisteminin en büyük eksiklikleri çok kısıtlı miktarda veriyi tutabiliyor olması, tekrar kullanılabilir olmaması, hata oranının yüksek olmasıdır. 21. Yüzyılda elektronik teknolojisindeki gelişmeler ile veri depolama birimleri gelişmiş bu sayede veri limitasyonu ortadan kalkmıştır. RFID teknolojisi bu dönemde ortaya çıkarak yeni nesil elektronik barkod sistemleri olarak adlandırılabilir. Tekrar kullanılabilir olması, veri tanımlayıcılarıyla veri çakışması problemini ortadan kaldırması, yüksek veri depolama haznesi sayesinde tercih edilen bir teknolojiye dönüşmüştür. RFID, içerisinde mikroişlemci ve anten bulunan bir etiket taşıyan nesnenin, bu etikette taşıdığı bilgiler ve kullanılan kablosuz iletişim teknolojisi (RF sinyalleri) sayesinde hareketlerinin izlenebilmesine imkan veren bir teknolojidir. RFID sistemlerinde, kablosuz iletişim teknolojileri kullanılarak herhangi bir nesnenin otomatik olarak tanımlanabilmesi, izlenebilmesi, nesneye ait dinamik bilgilerin ya da verilerin oluşturulması, toplanması, ve yönetilmesi amacı güdülmektedir. (Yüksel ve Zaim, 2009)

RFID, kablosuz ağlar ve sensör teknolojilerinin üretim alanlarında yaygınlaşmasıyla makineler arası iletişim gelişmeye başlamıştır. Bu uygulamalar ve cihazlar nesnelerin interneti kavramının oluşmasına katkı sağlamışlardır. Makinaların kendi aralarında haberleşmesi ve birbirlerinden aldıkları verilere göre yeni aksiyonlara karar verebilir olmaları ile Makinalar Arası İletişimi dönemi başlamıştır.

2.1.2 Makinalar Arası İletişim (M2M)

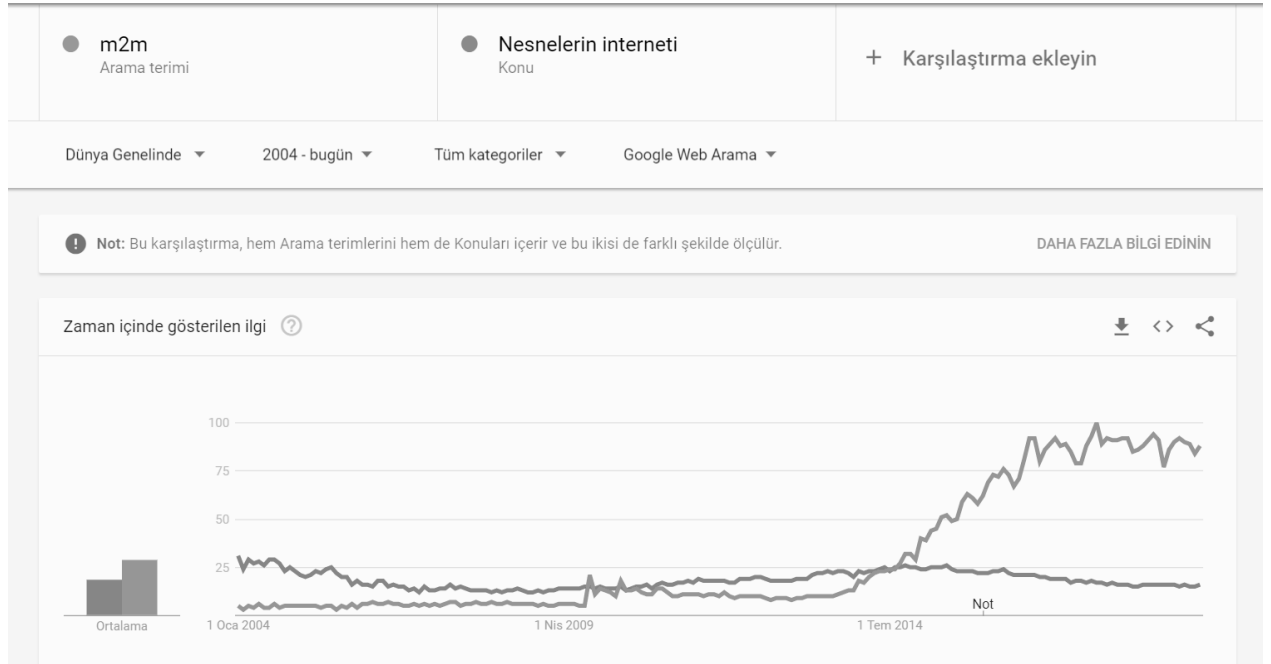
M2M kısaltmasıyla bilinen Makineler Arası İletişim (Machine to Machine Communication) iki ya da daha fazla makinenin önceden belirlenmiş kurallar ve protokoller ışığında kablolu ya da kablosuz iletişimi olarak adlandırılmaktadır. (BTK, 2013) Amaç insan etkileşimi olmadan birbirleri ile haberleşebilen cihaz sistemleri oluşturmaktır. Böylece insana bağlı hatalar en aza indirgenip üretim süreçleri, otomasyon süreçleri ve günlük yaşamımızdaki çoğu etkileşimin otomatize edilmesi hedeflenmiştir. Bu sistemlerin gelişmesi ve paralelinde gelişen mobil cihazlar, devre tasarımlarının küçülmesi kapsama alanı teknolojilerinin artması, tüm bu teknolojik gelişmeler ile

birbirleri ile iletişim cihazlar hizmet sektöründe, izlem, reklam ve lojistik uygulamalar ve üretim teknolojilerinde sıklıkla yer almaya başlamıştır.



Şekil 2.1 Makinalar arası iletişimin ülkemizde kullanım alanları (BTK, 2013)

2004'ten bu yana Google arama motorunun verilerine göre M2M ve Nesnelerin internet kavramlarına olan ilgi aşağıda gösterilmektedir.



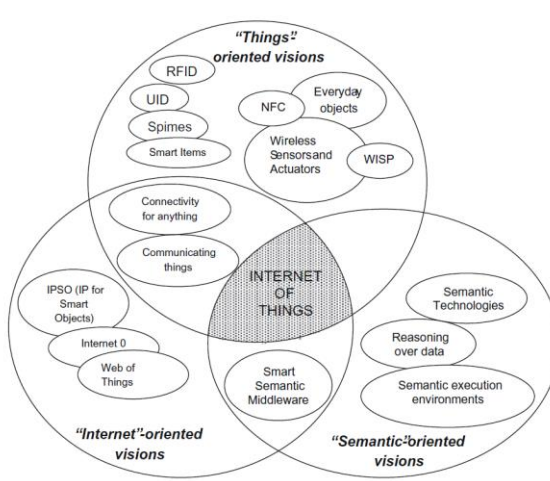
Şekil 2.2 Dünya genelinde nesnelerin interneti trendi (Google Trends, 2019)

Grafikte 2014'ten itibaren Nesnelerin İnterneti kavramının internet üzerinde ne kadar çok aratılmaya başlandığını görebiliyoruz. Bu kavramın bu kadar çok merak edilir ve aranır olmasından da anlaşıldığı üzere gün geçtikçe akıllı sistemler ve bu alana yapılan yatırımlar artmaktadır. Cihaz yetkinliklerinin artması ile zaman içinde cihaz ve aktörlerin haberleşebilir olması sağlanmıştır. Bu verilerin depolanması ve analiz edilebilmesi için yapılan çözümler nesnelerin interneti kavramı ile adlandırılmaya başlanmıştır.

2.1.3 Nesnelerin İnterneti

ITU'nun (Uluslararası Telekomünikasyon Komisyonu) yayımladığı 7. İnternet Raporunda belirtildiği üzere nesnelerin interneti geleceğin hesaplamalarını ve iletişimini gösteren bir çeşit teknolojik devrimdir ve bunun gelişimi nanoteknolojiden kablosuz algılayıcılara kadar farklı teknolojilerin gelişmesinden etkilenecektir. (ITU, 2005)

Nesnelerin interneti kavramı, benzersiz bir şekilde adreslenebilen nesnelerin standart protokoller üzerinden dünya çapında birbirleri ile bağlantılı olma durumudur. (European Commission, 2008)



Şekil 2.3 Nesnelerin interneti kavramının bileşenleri (Atzori,2010)

Nesnelerin interneti kavramı Atzori tarafından 3 bölümde incelenmiştir: Nesne odaklı, internet odaklı ve semantik odaklı. (Atzori, 2010) Nesnelerin interneti kavramı denilince akla ilk olarak nesne odaklı tanımlamalar gelmiştir. Nesne kavramı burada RFID tagları temsil etmektedir. Bu nesne odaklı nesnelerin interneti kavranımı RFID ve sensör teknolojileri alanında dünyaca ünlü akademik araştırma laboratuvar ağı Auto-ID Labs tarafından tanımlanmıştır. (Auto-ID Labs) RFID teknolojisinin yaygınlaştırılması ve endüstri standartlarının oluşturulması için çalışmışlardır. Nesnelerin takip edilebilmesi ve görünürlüğünün artırılması açısından bu çalışmalar önemlidir. Ancak nesnelerin interneti kavramı sadece bununla sınırlı değildir. RFID teknolojisinin yanı sıra

Yakın Alan İletişimi (NFC) ve Kablosuz Sensör ve Aktüatör Ağları (WISP) teknolojileriyle gerçek dünyanın dijital dünyada bir karşılığı oluşturulmaya çalışılmıştır. (Atzori, 2010)

Cihaz odaklı nesnelerin interneti tanımı zaman içinde internet odaklı bir tanım haline gelmiştir. Avrupa Komisyonunun tanımına göre nesnelerin interneti “Nesnelerin özgün tanımlamalar ve kişiselleştirmeler ile akıllı uzayları özel akıllı arayüzler ile yönetebilir olması durumudur” şeklinde tanımlanmıştır. (European Commision, 2008)

2008 yılında 25 şirketin ortaklaşa kurduğu Alliance forumu internet protokolü ile dünya üzerindeki tüm akıllı cihazların birbirine bağlanması için bir araya gelmiş ve IPSO (IP for Smart Objects) standartlarını oluşturmuştur. Cihazların birbiri ile bağlantılı olabilmesi için interneti kullanması, hem her yerden erişilebilir olmalarını hem de benzersiz bir adres ile tanımlı olmalarını sağlamaktadır.

Bağlantılı cihaz sayısının yıllar içinde artması ile oluşan verilerin kayıtlanması, gösterilmesi, verilerin aranması ve birbiri ile ilişkilendirilmesini zorlaştıracaktır. Bu nedenle semantik teknolojilere gereksinim doğmaktadır. (Atzori,2010)

Gelişen teknoloji ile birlikte farklı ölçümler yapabilen cihazların farklı bağlantı teknolojileri üzerinden veri gönderebilmesi, bu verilerin görüntülenmesi için uygulamaların yapılmasıyla birçok farklı alanda akıllı sistemler hayatımıza girmiştir. Önceleri bu alanda daha çok cihaz üreticileri geliştirdikleri kitler ile uyumlu, cihaz verilerini bir alanda toplayan uygulamalar ve sistemler geliştirerek aktif rol oynamışlardır. Bu çözümler dikey düzlemde uçtan uca çözümlerdir. Örneğin akıllı kombi üreten bir şirket üzerinden yola çıkacak olursak, kombinin verilerinin okunması ve kontrol edilebilmesi için üzerinde bir çalıştırıcının olması gerekmektedir. Önceleri Bluetooth ile ev içerisinde çeşitli uygulamalar ile kontrol edilebilen bu cihaz zaman içinde internete bağlanabilen modüller ile cep telefonu uygulamasıyla uzaktan kontrol edilebilir olmuştur. Kombin dıışarıdan kontrol edilebilir olması için verilerinin bir merkezde toplanılması gerekmektedir. Bu merkezi sistemin oluşturulması, kontrol uygulamasının hazırlanması tüm bu aşamaların kombi şirketi nezdinde yapılması dikeyde uçtan uca çözüm veren bu şirketin uygulama sürdürülebilirliğini de zorlaştırmaktadır.

Nesnelerin interneti kavramı sadece cihazdan veri okumak ve bu verileri uygulamalarda görüntülemek olarak basite indirgenmemelidir. Cihazların internete bir şekilde bağlanabilir olmaları da tek başına nesnelerin internet kavramının içini tam olarak dolduramamaktadır. Farklı özellikteki farklı üreticilerin ürettiği çeşitli protokolleri kullanan ve farklı veri modelleri olan nesnelerin birbirleri ile haberleşebilmeleri ve aynı zamanda bu sistemlerin sürdürülebilir sistemler olabilmesi ihtiyacı platformların ortaya çıkmasını sağlamıştır.

Nesnelerin interneti platformlarının amacı genel özellikleri itibari ile çeşitli veri üreticilerinden (cihaz, sensör, veri sağlayıcılar) verilerin gerçek zamanlı olarak alınıp belirlenen kurallar ve aksiyonlar çerçevesinde başka sistemler ile etkileşiminin sağlanması, verilerin güvenli yerlerde depolanması ve bu depolanan verilerden analizlerin elde edilmesidir.

2017 yılında IoT Analytics'in raporuna göre dünya üzerinde 450 farklı IoT Platformu bulunmaktadır. Bu devasa rakamdan da anlaşıldığı gibi günümüzün en revaçta olan alanlarından birisi artan akıllı cihazlar ve akıllı sistemlerin oluşturduğu pazarda yer almaya çalışan şirketlerin geliştirdikleri nesnelerin internet platformlarıdır.

Bu çalışmada nesnelerin interneti platformlarının en yalın anlamda nasıl olabileceği tasarlanıp ve prototip düzeyinde gerçekleştirilmiştir.

2.1.4 Nesnelerin İnternetinin Geleceği

Çeşitli yollarla birbirleri ve internet ile bağlı olan cihazlar arttıkça internet üzerindeki trafiğin de büyümesine neden olacaktır. Ayrıca bağlantı teknolojilerindeki mobilite, iletim hızı gibi faktörler de önemini arttırmaya devam edecektir. Bu nedenle nesnelerin internetinin yaygınlaşması ve kritik alanlarda kullanılabilmesi için 4G (4. Nesil) gibi mobil hizmetlerin yetersiz kalacağı öngörülmektedir. Bu nedenle 5G (5. Nesil) gibi mobil bağlantı teknolojileri geliştirilmektedir. 5G ile hayatımıza girmesi beklenen bazı senaryolar şöyle sıralanabilir:

- Uzaktan kontrol edilebilen ulaşım araçları
- Sürücüsüz/kontrolcüsüz araçlar
- Kritik uygulamalarla beraber gelişmiş akıllı trafik ve şehir uygulamaları
- Uzaktan yapılabilen ameliyatlar
- Uzaktan kullanılabilen insansız hava araçları

Nesnelerin interneti alanındaki gelişmeler devam ettikçe güvenlikle ilgili sıkıntıların da olması beklenmektedir. 2016 yılında nesnelerin interneti cihazlarının içine kolayca yerleştirilmiş korsan bir yazılım olan Mirai'nin yüklenmesiyle gerçekleşen DDOS (Dağıtılmış Hizmet Reddi) saldırısı yapması sonucu tüm dünya üzerinde DynDos şirketi ile ortak çalışan Netflix, Twitter, Spotify gibi şirketler internete bağlanma problemi yaşamıştır. Sadece Amerika'da bu sürede yaşanan zararın 7 milyar dolardan fazla olduğu düşünülmektedir. Nesnelerin interneti platformlarının ve bunlara bağlı cihazların güvenliği geliştirilmesi gereken alanlar arasındadır.

Akıllı cihazların hayatımızda daha fazla kullanılır olması için batarya sürelerinin de optimize edilmesi gerekmektedir. Ağa bağlı cihazların veri aktarım süresinde uyanık kalıp sonrasında uykuya geçtiği, belirli aralıklarla ağa kalp atışı şeklinde aktiflik göstergesi yolladığı yapılar ile optimizasyon sağlanmaktadır. Ayrıca LPWAN (Low power wide area network) ağı bağlantı teknolojileri ile cihazlar düşük enerji tüketimi ile küçük büyüklükteki verileri uzak mesafelere iletebilmektedir. Bağlantı teknolojisi opsiyonları gerçekleştirilecek çözümün ihtiyaçlarına bağlı olarak seçilmelidir.

2.2 Literatür Taraması

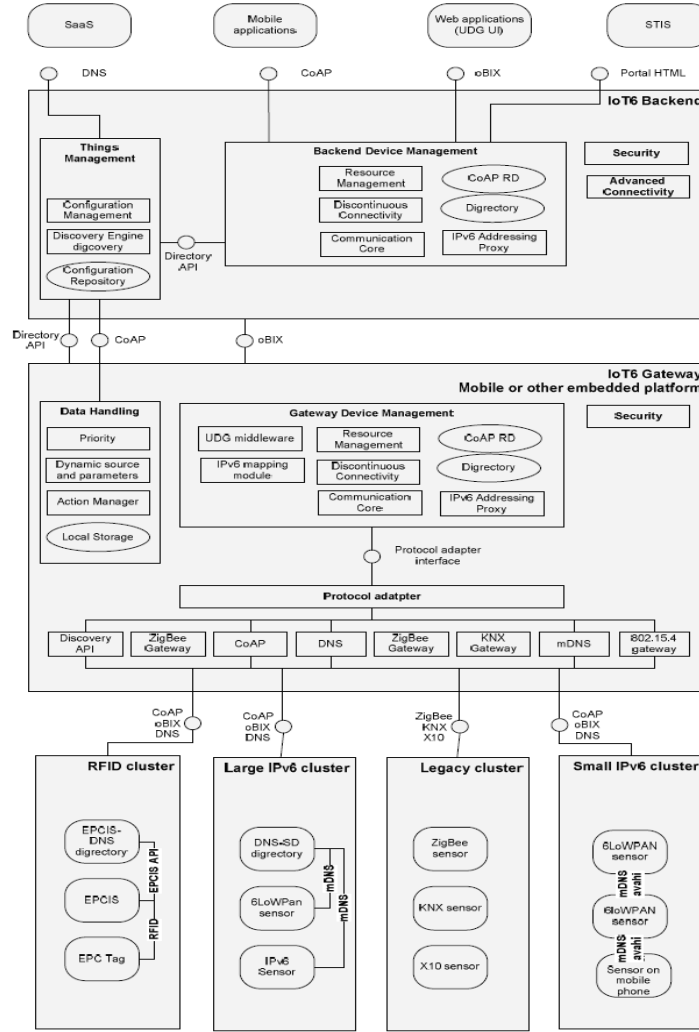
Literatür taramasında nesnelerin interneti alanında oluşturulmuş olan derlemeler, mimari model incelemeleri, yeni mimari model oluşturma çalışmaları incelenmiştir. Çalışmalar aşağıda belirtilen 3 grupta incelenmiştir.

- Derlemeler:
 - Nesnelerin İnterneti Derlemesi (Zdravković, M., 2016)
- Mimari model oluşturma çalışmaları:
 - Yalın bir mimari model oluşturma çabasında olanlar
 - Açık Kaynak Kodlu ve Yalın bir Nesnelerin İnterneti Platformu ile Bağlam Bilinçli Uygulamalar Geliştirme (Ribeiro, F. and Metrolho, J.,2016)
 - Nesnelerin İnterneti için Yalın Mimari Gerçekleştirimi (Heo, Y.J, et. al., 2015)
 - Nesnelerin Bağlanması ve Kontrolü: Mobil Uygulama ile Yalın bir Nesnelerin İnterneti Platformu Sağlama (Datta and Bonnet, 2015)
 - Nesnelerin İnterneti Platform Mimarilerinin Karşılaştırılması (Guth, J., et.al., 2016)
 - Ortak bir mimari model oluşturma çabasında olanlar:
 - IoT6 Referans Mimarisi (Krcó et al., 2014)
- Senaryo tabanlı nesnelerin interneti çözüm yaklaşımları:
 - Akıllı Ev Sistemleri için Nesnelerin İnterneti (Malche, T. and Maheshwary, P.,2017)

Bu çalışmanın oluşturulmasında rol oynayan çalışmalar yukarı da belirtilmiştir. Çalışmaların yararlanılan bölümleri, tez çalışmasında geliştirilen model ile arasındaki ortak ve farklı noktaların ayrıntıları bu bölümde açıklanmıştır.

2.2.1 IoT6 Referans Mimarisi

Krcó ve arkadaşları nesnelerin interneti platform mimari modellerini inceleyerek ortak bir platform mimarisi oluşturmayı hedeflemişlerdir. Çalışmada ETSI M2M ve OneM2M mimari modelleri, FI-WARE mimarisi ve IOT-A referans mimari modeli incelenmiştir. IoT6 modeli oluşturulmuştur. Ortak bir terminoloji yaratmak, birlikte çalışabilirliği arttırmak için mimari model oluşturmaya çalışılmıştır. Model oluşturulurken çeşitli kullanım senaryolarından alınan gereksinimleri, IoT-A referans modeli baz alınarak ve mühendislik beceriler kullanılarak şekil 2.4'te verilmiş olan yeni bir mimari model oluşturulmuştur.



Şekil 2.4 IoT6 mimari modeli (Krco et al., 2014)

Fonksiyonel boyutta bir model geliştirmişlerdir. Amaçları daha önceden geliştirilmiş olan ETSI, OneM2M ve FI-WARE mimari modelleri ile uyumlu çözümlerinde desteklenebildiği, aynı zamanda ipv6 destekleyen bir platform yapısı ortaya koymaktır. Bu nedenle ipv4 destekleyen geleneksel cihazlar ile ipv6 yi destekleyen yeni nesil cihazları ayrı kümelerde toplayabilecek bir mimari model önermişlerdir.(Krco et al., 2014)

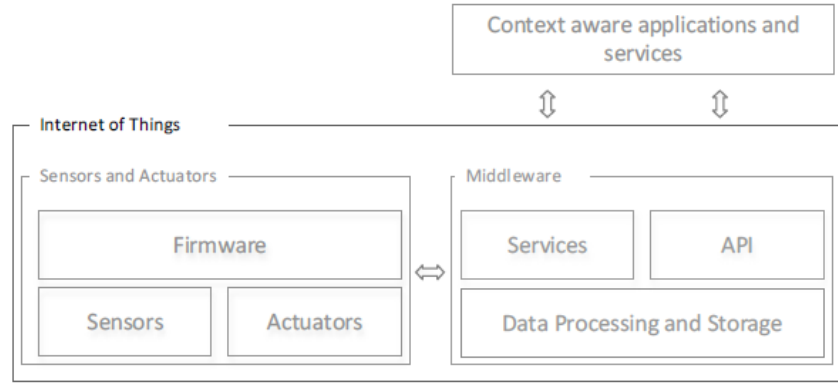
Krco'nun çalışmasındaki yapı önceden geliştirilmiş modelleri de destekleyebilmek için yalınlıktan uzaklaşmıştır. Cihaz ve bağlantı katmanı özelindeki model yapısı alt 2 ana bloğu oluşturmuştur. En üstte yer alan IoT6 Backend olarak adlandırılmış arka plan servislerinin sağlandığı yapı platform mimarisi yaklaşımı olarak bu tez çalışmanın kapsamı ile uyumludur. Temel olarak cihaz yönetimi modülüne ait servisler dış dünyaya servis olarak (aaS - as a Service) açılmaktadır. Cihaz yönetimine ait arka plan yönetim servisleri yönetim portalı üzerinden kullanılabildiği gibi uygulamalar tarafından da kullanılabilir.

2.2.2 Açık Kaynak Kodlu Yalın bir Nesnelerin İnterneti Platformu ile Bağlam Bilinçli Uygulamalar Geliştirme

Fernando ve arkadaşları nesnelerin interneti platformlarını ve referans mimari modelleri incelemişlerdir. Bu modellerden yola çıkarak bir mimari model hazırlamış ve onun gerçekleştirimini yapmışlardır. (Ribeiro, F. and Metrolho, J.,2016)

Çalışmada esneklik ve kolay anlaşılabilirlik esas alınmıştır. Bu ilkeler doğrultusunda model oluştururken cihaz bağımlı yada herhangi bir semantik modelleme yapılmamıştır. Şekil 2.5'te verilmiş olan modele göre geliştirilen platform üzerinde bazı senaryolar da gerçekleştirilmiştir. Mimarideki modüllerin görevleri şöyle sıralanabilir:

- Sensörler ve yapıcılar bloğu: Donanımsal olarak cihazları ve yazılımsal olarak onlardan gelen verileri veri işleme birimine gönderen, cihazı izleyen ve gerektiğinde veriyi manipüle eden yazılımı içeren modülü temsil etmektedir. Platform üzerine cihazların yönetici kullanıcı tarafından kaydı gerekmektedir. Platform bu aşamadan sadece raspberry pi ve arduino üzerindeki sensör verileri ile haberleşebilmektedir.
- Orta katman: Veri işleme ve depolama yapar. Aynı zamanda bu verilerin uygulamalar tarafından kullanılabilmesi için programlama arayüzleri aracılığı ile internet üzerinden servis edilmesini sağlamaktadır.



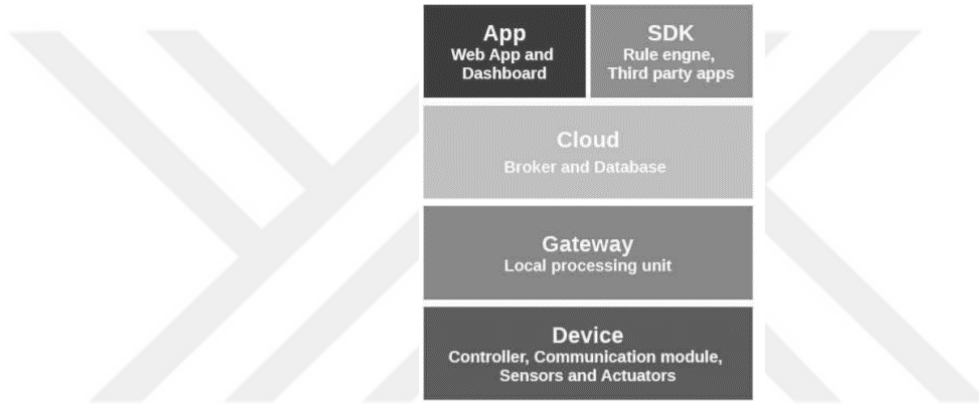
Şekil 2.5 Fernando ve arkadaşlarının mimari önermesi (Ribeiro, F. and Metrolho, J.,2016)

Çalışmada yeralan, var olan platformlar üzerinden yapılan merkezi model yapısı bu tez çalışmasında önerilen mimari model ile benzerlik göstermektedir. Bu tez çalışmasında ek olarak uygulamalar da platform üzerinden yönetilebilmektedir. Geliştirim esnasında kullanılan teknolojilere değinilmemiştir. Esneklik açısından incelendiğinde platformun bu haliyle sadece belli fonksiyonlileri gerçekleyen programlama arayüzüne sahip olduğu görülmektedir. Bu tez çalışmasında özgün servislerin de sistem üzerinde çalıştırılabilir olması esnekliği arttırmaktadır.

Ayrıca farklı büyüklüklerde ve gereksinimlerdeki çözümler için hedeflenen mimari yapı ile de farklı bir bakış açısı sunulmaktadır.

2.2.3 Akıllı Ev Sistemleri için Nesnelerin İnterneti

Akıllı ev uygulaması geliştirmek amacı ile açık kaynak kodlu bir sistem kullanılmak istenilmiştir. Hindistanda geliştirilmiş olan Flip platformu kökten uca nesnelerin interneti çözümü vermek üzere bazı senaryolar için geliştirilmiştir. Özel geliştirilmiş kartlar üzerindeki sensörler ile akıllı ev sistemi dışında tarım, sağlık ve lojistik uygulamalarını desteklemektedirler.(flip cloud, 2019) Platform mimarisi şekil 2.6’da gösterilmektedir.



Şekil 2.6 Flip mimari modeli (Malche, T. and Maheshwary, P.,2017)

Cihaz katmanında Arduino Nano kartını kullanan çözümde ağ geçidi modülü RaspberryPi kartıdır. Cihazlar ağ geçidi üzerindeki modüller sayesinde Wifi veya Bluetooth üzerinden veri gönderebilmektedir. Veriler bulut üzerindeki platform yapısına gönderilmektedir. Şekil 3.3’te Cloud olarak adlandırılmış yapı Node.js arka plan servislerini kullanarak mesaj kuyruğu yapısıyla verileri MongoDB üzerinde tutmaktadır. Akıllı ev sisteminde (Malche, T. and Maheshwary, P.,2017) kullanılmış olan bu sınırlı ama temel fonksiyonalityi sağlayan platform yapısı fikir ve teknolojiler açısından bu çalışmaya fikir vermiştir.

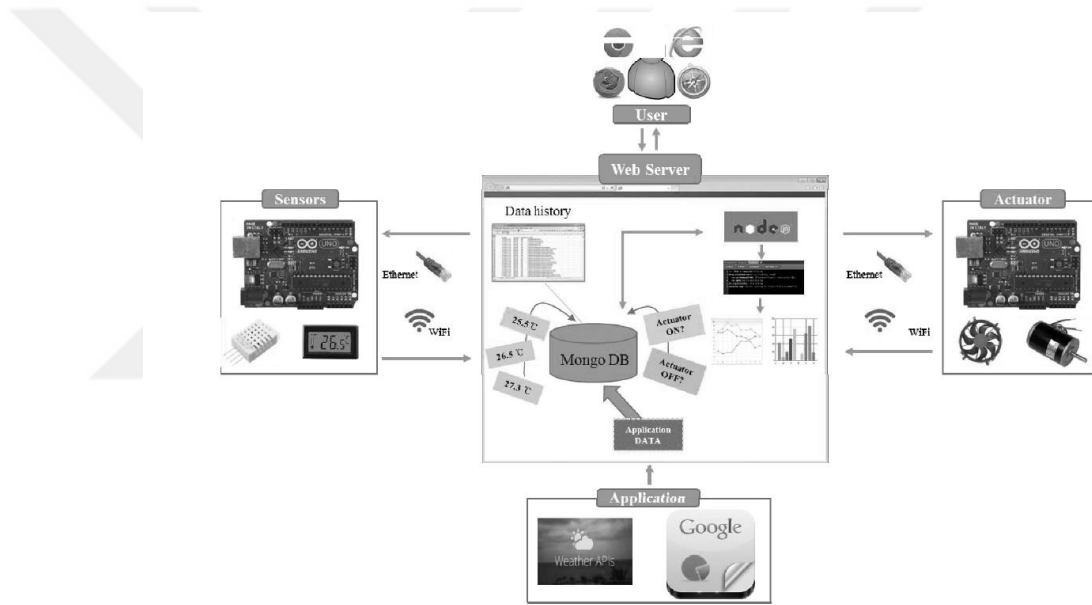
2.2.4 Nesnelerin İnterneti Derlemesi

Zdravković ve arkadaşlarının araştırması var olan bulut tabanlı nesnelerin interneti çözümleri ve avrupadaki standart enstitülerinin referans modellerini kapsamaktadır. Hızla gelişmekte olan nesnelerin interneti dünyasında var olan çözümlerdeki ortak noktalar ve araştırılması gereken konular analiz edilmiştir. Bu çalışma, tez çalışmasında çekirdek yapıda mutlaka bulunması gereken yapıların belirlenmesine yardımcı olmuştur. Marketteki platformlar arasında farklılığı ve tercih edilebilirliği belirleyen unsurun makinalar arası iletişim sağlayıcı ve yönetimi yapısının olduğu orta katman (middleware) olduğu belirtilmiştir. Veri görüntüleme, veri analizi, uygulama geliştirme gibi fonksiyonalityelerin başka sağlayıcılardan da alınabileceği gösterilmiştir. Zdravković

ve arkadaşlarının çalışması sektörde yer alan platform ürünleri ile ilgili bilgi vererek, bu tez çalışmasının odaklandığı çekirdek fonksiyonalityi oluşturma yaklaşımını desteklemektedir. (Zdravković, M., 2016)

2.2.5 Nesnelerin İnterneti için Yalın Mimari Gerçekleştirimi

Heo ve arkadaşları çalışmasında bir nesnelerin interneti platformu geliştirimi yapılmıştır. (Heo, Y.J, et. al., 2015) Platform dizaynı sadelik ön planda tutularak yapılmıştır ve bu tez çalışması ile benzer yapılara sahiptir. Veri depolama, görüntüleme, sanallaştırma, uygulama servisleri mimari yapının temelini oluşturmuştur. Mimari model şekil 2.7’de gösterilmektedir. Geliştirmede bu tez çalışması ile benzer olarak node.js ve mongoDB kullanılmıştır.



Şekil 2.7 Yalın nesnelerin interneti platformu mimarisi (Heo, Y.J, et. al., 2015)

Farklı olarak bu tez çalışmasında değişik ölçeklerdeki sistem çözümleri için uygulanabilir model yapısı sunulmuştur, ayrıca platformun özel geliştirilmiş servisleri de çalıştırabilmesi ile kullanıcılara kullanım senaryolarını gerçekleştirmek için esneklik sağlamaktadır.

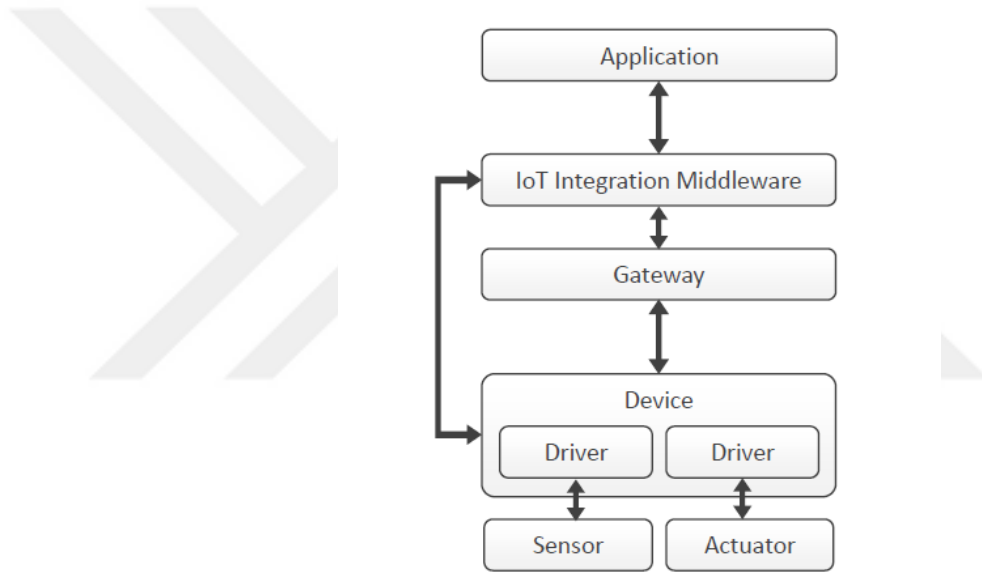
2.2.6 Mobil Uygulama ile Yalın bir Nesnelerin İnterneti Platformu Sağlama

Datta ve Bonnet’in çalışmasında nesnelerin interneti platformunun sahip olması gereken özellikler incelenmiştir. Bir model düşünülmüş ve geliştirilmiştir. Amaç mobil cihazlar üzerinde nesnelerin interneti cihazlarının kontrolünün sağlanabilmesidir. Bu nedenle önerilen platform yüklenmiş mobil cihazlarda bellek, CPU ve pil kullanımı metrikleri incelenmiştir. Akıllı ev kullanım senaryosu ile geliştirilmiş çözüm için ölçümler yapılmıştır. Bu çalışma nesnelerin interneti cihazlarının yönetiminin mobil cihazlar seviyesinde tutarak dikeyde bir çözüm sistemi

sunmuştur. Bulut sistemlerinde de kullanılabilir olma ya da ortak bir platform oluşturma fikri güdülmemiştir. Bu tez çalışmasına bir nesnelerin interneti çözümünün sahip olması gereken katmanları nelerdir, konusunda destek olmuştur. (Datta and Bonnet, 2015)

2.2.7 Nesnelerin İnterneti Platform Mimarilerinin Karşılaştırılması

Guth ve arkadaşlarının çalışmasında platform karşılaştırmalarının sağlıklı bir şekilde yapılabilmesi için ortak bir terminoloji geliştirilmesi gerektiğine değinmiştir. Bu nedenle referans olarak kullanılabilecek ortak özelliklerin barındırıldığı bir mimari model oluşturulmaya çalışılmıştır. Model şekil 2.8’de gösterilmektedir. Modelin yapıtaşları bu tez çalışmasındakiler ile katmanlama açısından benzerlik göstermektedir.



Şekil 2.8 Guth ve arkadaşlarının referans mimari modeli (Guth, J., et.al., 2016)

Bu çalışmada bahsedilen “Nesnelerin İnterneti Entegrasyon Orta Katmanının” (IoT Integration Middleware) tez çalışmasındaki “Entegrasyon katmanı” ile fikir olarak benzerlik göstermektedir. Genel olarak belirlenmiş referans model yapısı çalışmada değinilen ticari modeller içerisinde incelenerek, nasıl kullanıldığına bakılmıştır.

2.3 Market Araştırması

Son yıllarda çoklukla kullanılan nesnelerin interneti platform çözümlerine ait mimari modeller incelenmiştir. Çalışmada bu modeller üzerinde yalınlaştırma yapılarak gerekli merkezi fonksiyonaliteye odaklanılmıştır.

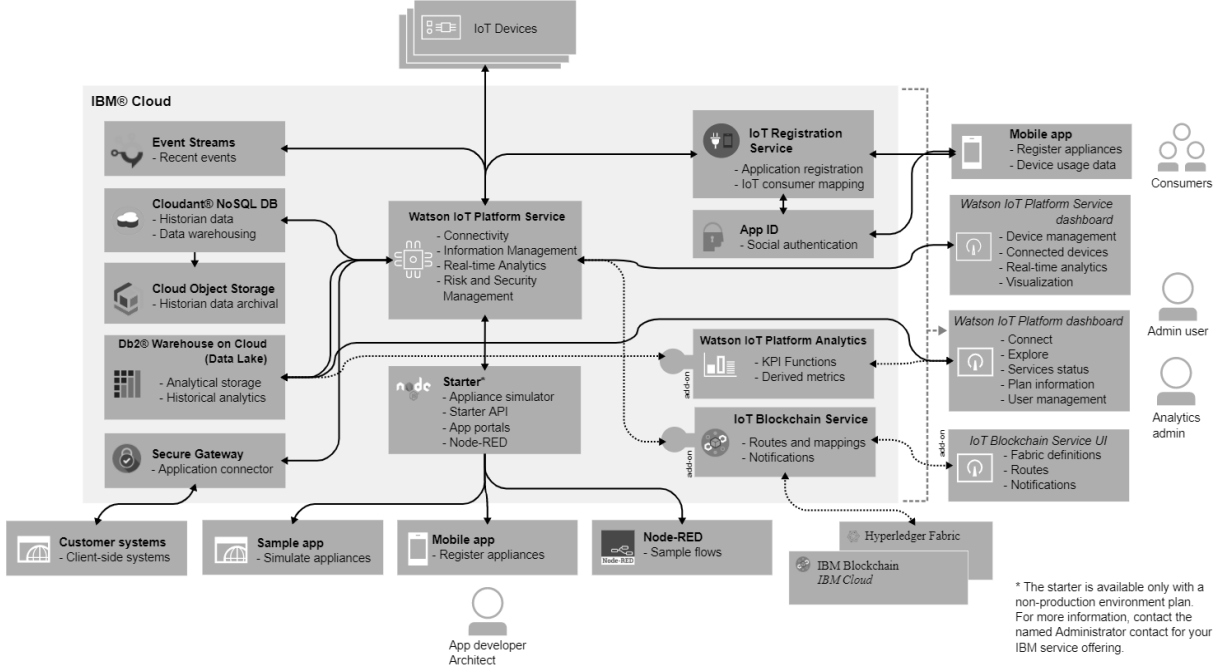
2.3.1 IBM Watsons

IBM şirketi bulut üzerinden Watsons adıyla nesnelerin interneti platformu çözümü sağlamaktadır.(IBM Watsons, 2019) IBM bulut servisleri ile sağlanan başlıca modüller ve görevleri listelenmiştir:

- Watson Platform Servisi: Cihazların bağlandıkları bir mesaj kuyruğu yapısıdır.
- Kayıt Modülü: Kayıt ekranları ve servislerinden oluşan cihazları ve kullanıcılarını bağdaştıran modüldür.
- Başlangıç Kiti: Yalnızca demo sürümünde bulunan bu modül ile “Watson Studio” programlama arayüzü üzerinden uygulama ve servis üretmeyi sağlar.
- Bulut NoSQL Veritabanı: Bulut üzerinde gerçek zamanlı verilerin tutulması için kullanılan NoSQL veritabanı modülüdür.
- Db2 Bulut Veri Ambarı: İlişkisel verilerin tutulması için kullanılan modüldür.
- Bulut Obje Deposu: Yapısal olmayan verilerin uzun süreli tutulması için kullanılabilecek veri deposudur.
- Bulut Uygulama Erişim Anahtarı: Mobil ve web uygulamalarına Google+ ve Facebook hesapları üzerinden erişimi sağlayan modüldür.
- IBM Bulut için Güvenli Geçit : IBM bulut uygulaması üzerinden başka bulut yada sunucu üzerindeki verinin alınmasını sağlayan güvenli bir geçit modülüdür.

Blok zinciri ve analitik ek modülleri de mevcuttur. Watsons’a ait mimari model yapısı şekil 2.9’da gösterilmektedir.

IBM® Watson™ IoT Platform



Şekil 2.9 IBM Watsons nesnelerin interneti platform mimarisi (IBM Watsons, 2019)

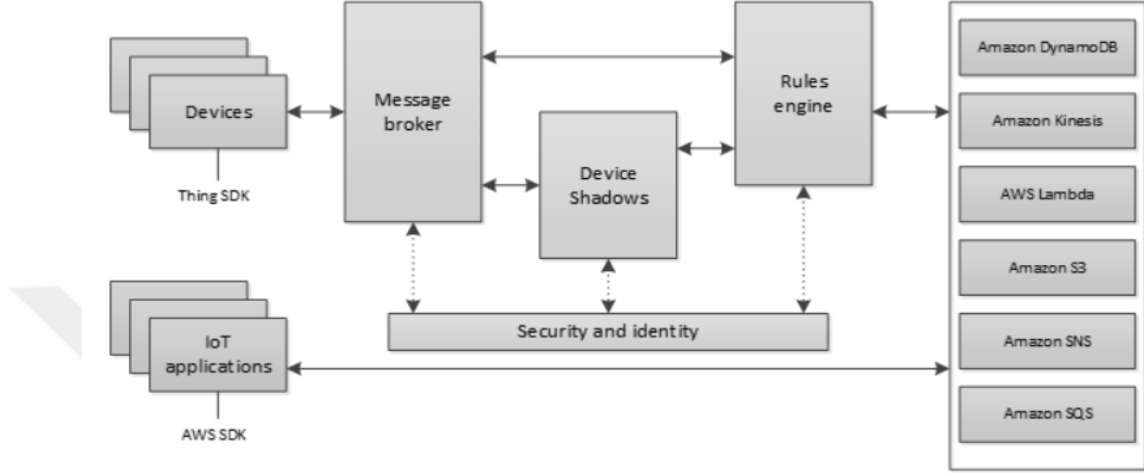
2.3.2 Amazon AWS IoT Suite

AWS IoT Core, Amazon'un nesnelerin interneti platformunun özel adıdır. Cihazların bulut üzerinden uygulamalar ve diğer cihazlar ile iletişimini kolay ve güvenli bir şekilde sağlaması için geliştirilmiştir. AWS IoT Core, altyapı yönetimi gerekmeksizin bağlı cihazlar tarafından oluşturulan verileri toplamaya, işlemeye, analiz etmeye ve bunlardan yola çıkarak harekete geçmeye yönelik IoT uygulamaları oluşturmak için AWS Lambda, Amazon Kinesis, Amazon S3, Amazon SageMaker, Amazon DynamoDB, Amazon CloudWatch, AWS CloudTrail ve Amazon QuickSight gibi Amazon hizmetlerini kullanmayı da kolaylaştırır. (Amazon AWS, 2019)

Şekil 2.10'da verilen mimari model incelendiğinde platform yapısının başlıca öğelerden oluştuğu görülmektedir:

- **Mesaj Kuyruğu**: Mesajların bir kaynaktan başka bir kaynağa mesajın sırası geldiğinde iletilmesini sağlayan yapıdır. Yayımcı ve abone (Pub/Sub) mantığı için gerekli bir modüldür.
- **Güvenlik ve Erişim**: Platform güvenliğinin sağlandığı ve erişim haklarının kullanıcı rollerinin tanımlandığı modüldür.
- **Kural Motoru**: Platforma bağlı cihazlar ve uygulamalar için belirlenen kurallar çerçevesinde bildirimler ve alarmlar hazırlamayı sağlar.
- **Cihaz Gölgesi**: Bu modül ile cihazların platform üzerindeki bir anlamsal yedeğinden bahsedilmektedir, bu anlamsal yapı cihazların durum bilgisini barındırmaktadır.

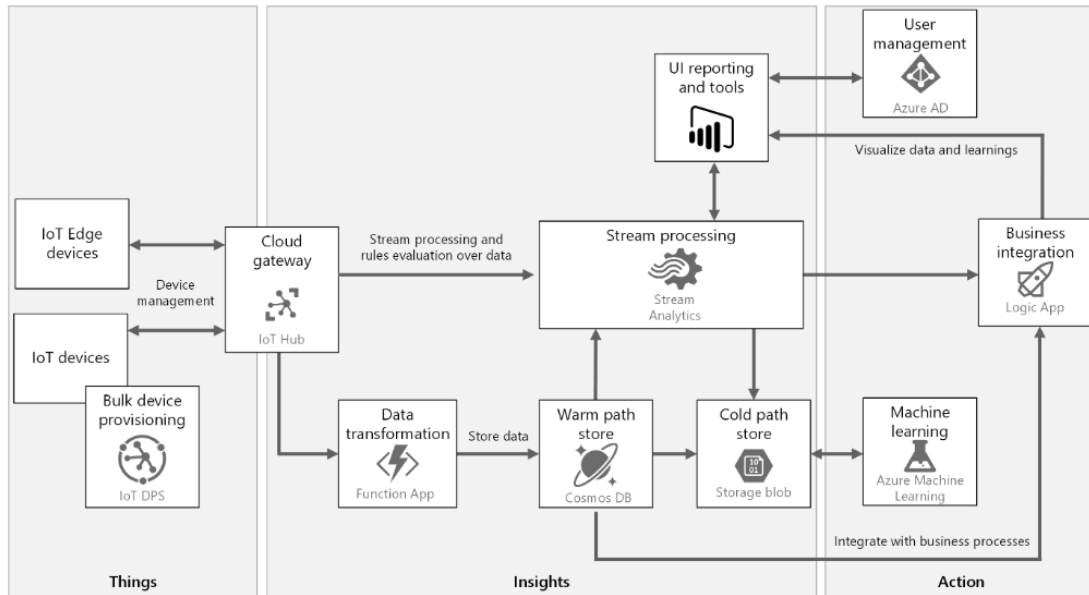
- Amazon Bulut Servisleri: Veri merkezi olarak buluttan hizmet veren DynamoDB, veri akışı analizi için Kinesis bulut servisi, çalıştırma ortamı için Lambda bulut servisi gibi çeşitli modüller Amazon'un bulut servisleri arasında yer almaktadır.



Şekil 2.10 Amazon nesnelerin interneti platform mimarisi (Amazon AWS, 2019)

2.3.3 Azure IoT

Windows nesnelerin interneti için geliştirmiş olduğu servisler bütününe Azure IoT Suite adını vermiştir. Şekil 2.11'deki şemada bir nesnelerin interneti uygulamasının Azure IoT platform servislerini kullanarak oluşturulabilecek bir referans mimari paylaşılmıştır.



Şekil 2.11 Azure IoT referans mimari modeli (Windows Azure, 2019)

Kullanılan modüllerin görevleri aşağıda verilmiştir:

- IoT Hub: Bulut üzerinde cihazların bağlanabildikleri ve yönetilebildikleri yapıdır aynı zamanda bir mesaj kuyruğudur.
- Akış Analizi: Cihazlardan gelen akım verilerini kullanım senaryolarına uygun olan veri depolarına yönlendirilmektedir. Aynı zamanda gerçek zamanlı raporlama ve analitik modülleri ile de iletişime geçebilmektedir.
- CosmosDB: Gerçek zamanlı sistemler için kullanılabilecek veritabanı opsiyonudur.
- Storage Blob: Platform verilerini uzun süreli depolanmasında kullanılan teknolojidir, veriler yığın işlemede kullanılır. Bu veriler uzun vadede makina öğrenmesi için geliştirilmiş Azure modülü ile de iletişime geçebilmektedir.
- Function App: Cihazlardan gelen veriler için bir veri transformasyonu gerekiyorsa bunu gerçekleştiren yapıdır.

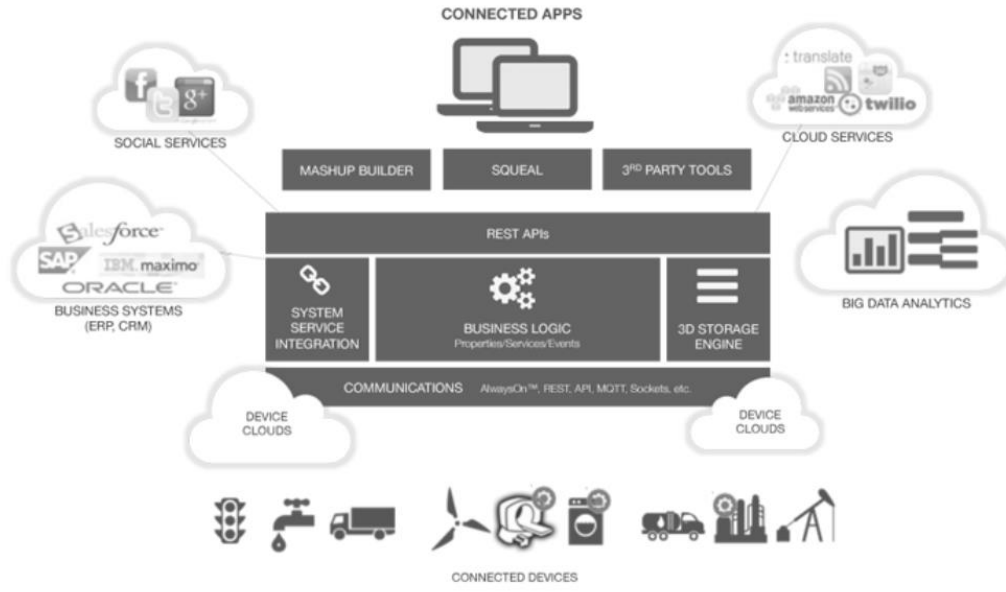
Platform yapısı çeşitli aksiyonları oluşturmak için farklı sistemlerle iletişime geçebilir. İş süreci modülleri ile Logic App çözümü sayesinde entegre olabilmektedir. (Windows Azure, 2019)

2.3.4 Thingworx

Thingworx, PTC şirketine ait uçtan uca nesnelerin interneti çözümü geliştirmeye olanak sağlayan bir sistemler bütünüdür. Bu yapı lokal sunucularda kurulabileceği gibi bulut üzerinde de servis edilebilmektedir. Sistem tomcat üzerinde çalışmaktadır. Gömülü H2 depolama ve PostgreSQL gibi farklı veritabanı modülü ile kurulum opsiyonları mevcuttur. Ayrıca uzaktaki veritabanları ile de iletişime geçebilmektedir. Kodlamayı minimuma indirerek platform servisleri ve görsel modüller kullanılarak nesnelerin interneti web uygulamaları oluşturmak için geliştirilmiştir. Sistem üzerinde her soyutlama nesneler ile özdeşleştirilmiştir. Nesne grupları, modelleri belirlenir. Servis geliştirmeye olanak sağlar. Bu sayede istenilen algoritma akışı sağlanabilir. Servisler REST çağrılar olarak dış sistemlerden kullanılabilir, 3. şahıs sistemlerle entegrasyon bu sayede sağlanabilir. Servisler ve modeller görsel modüller ile web çözümlerine dönüşür. Sürükle bırak modeli sayesinde dizaynı kolaylaştırır. Geliştirilen uygulama yine Thingworx üzerinden sunulmaktadır.

Bu nesnelerin interneti platformu uçtan uca çözümler için kolaylıkla kullanılabilen bir yapıya sahiptir. Ancak cihaz yönetimi, servis yönetimi konusunda platform yapısına sahip değildir.

Thingworx'e ait mimari model yapısı şekil 2.12'de verilmiştir.



Şekil 2.12 Thingworx mimari modeli (Thingworx Case Study, 2019)

3 PLATFORM ÖNERİSİ

Önceki bölümlerde nesnelerin interneti dünyasında yapılan çalışmalar ve markette söz sahibi ürünlerin mimari modelleri incelenmiştir. Sonraki bölümlerde ise bu incelemelerin ışığında nesnelerin interneti platformlarının mutlaka sahip olması gereken özellikler düşünülerek oluşturulmuş katman yapısından bahsedilecek, gerçekleştirimi yapılan platformun mimari modeline değinilecektir. Gerçekleştirimde kullanılan teknolojiler ve yazılıma ait bilgilendirmeler yapılacaktır. Sistemin sağladığı yönetimsel servislerin kullanımlarına ait REST sorguların ekran görüntüleri de paylaşılacaktır.

Tez çalışması öncesinde ilgili referans bir mimari oluşturma, merkezi fonksiyonaliteyi yalın bir şekilde sağlama amacıyla yapılmış çalışmalar incelenmiştir. Bu çalışmalara 2. bölümde değinilmiştir.

Tez çalışmasında platform yapısı cihaz katmanından bağımsız olarak düşünülmüştür. Yazılımsal olarak bağlantı katmanı protokollerini desteklemek ve verileri uygulama katmanına kadar yönetilebilir bir şekilde depolayarak taşımak amaçlanmıştır. Geliştirme kısmında ise incelenen çalışmalardan farklı olarak kullanıcıların belli kısıtlamalara maruz kalmadan hızlı bir şekilde geliştirme yapabilmelerine olanak sağlamak amaçlanmıştır. Özelleşmiş çözümlere ait programlama arayüzlerini (API) yükleyebilecekleri ve bunlar ile cihazları yönetebilecekleri bir yapı kurmak amaçlanmıştır. Kullanıcılar uygulamalarını ve servislerini platform üzerinden internete açabilirler. Ayrıca cihazlarını hali hazırda sunulan fonksiyonaliteler ile monitor ve kontrol edebilirler.

Çekirdek mimari model hem heterojen ve büyük ölçekli, hem de küçük ölçekli çözüm ihtiyaçları için önermelere sahiptir. Farklı teknolojik yapıların entegrasyonu yapılabilir. Farklı alanlara ait çözümler küçük çaplı ayırık çekirdek mimarilerde tutulup beraber veya ayrı yönetilebilir. Bu kullanım önerileri mimari modelin anlatıldığı bölümde ayrıntılandırılacaktır.

İncelenen sistemlerden farklı olarak oluşturulan prototip mimari hem bulut hem de lokal sunuculara kurulabilmektedir. Birçok ticari nesnelerin interneti platformunun problemi olan lokal sunucu kurulumu problemini yaşamamak için gerçekleştirimde kullanılan teknolojilerde bulut servislerine bağımlılık yoktur.

3.1 Platform Önerisinin Katman Yapısı

Yapılan literatür taramasında ve nesnelerin interneti marketinde sıklıkla kullanılan çözümler incelenmiştir. Yalın bir nesnelerin interneti platformunun mutlaka sahip olması gereken katmanlar Şekil 3.1’deki gibi yorumlanmıştır:



Şekil 3.1 Platform katmanları

3.1.1 Entegrasyon katmanı

Farklı protokollerle haberleşebilen cihazların Nesnelerin İnterneti platformuna bağlanabilmeleri için bu protokolleri destekleyebilen bir entegrasyon katmanı olması gerekmektedir. Bu katman cihazlar için platformun giriş kapısı olarak adlandırılabilir. Cihaz verileri çeşitli işlemlere ve kurallara bağlı olarak işlenebilir. Cihaz verileri mesaj kuyruklarında bekletilerek tanımlanmış dinleyicilere yönlendirilebilir. Bu yapılar çözümün gereksinimlerine bağlı olarak eklenebilir. Bu katmandan sonra platform içerisinde protokol bağımsız bir ortam oluşmaktadır. Entegrasyon katmanında yaygın kullanılan protokollere ait açıklamalar aşağıda belirtilmiştir.

3.1.1.1 Http / Https

HTTP, genel internet iletişim kurallarını belirleyen uygulama katmanı protokolüdür. Taşıma katmanında TCP veya UDP protokolünü kullanmaktadır. Güvenlik protokolleri ile desteklenmiş versiyonu ise HTTPS olarak adlandırılır. Bir sunucu ile bir alıcısının arasındaki istekler ve cevaplar bu protokole göre gerçekleştirilir. HTTP nesnelerin interneti platformları için alıcı verici yapısında istek-cevap yöntemi ile çalışır.

Öncelikle istemci sunucu ile TCP bağlantısı kurar. 3'lü el sıkışma sağlanır. HTTP protokolünde 2 çeşit mesajlaşma bulunmaktadır, bunlar istek ve cevap şeklinde adlandırılır.

HTTP istek mesajları TCP ile bağlantı kurulduktan sonra alıcının sunucudan istekte bulunduğu mesajlardır. Belirli metodlar yardımı ile veri iletimi yapılmaktadır. Bunlar GET, POST,

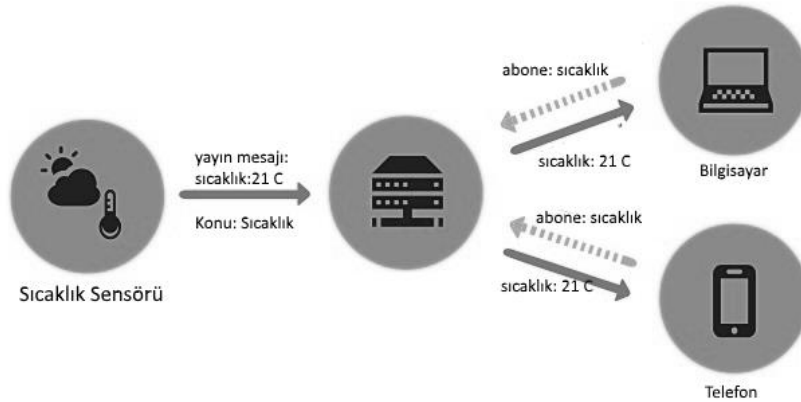
DELETE, PUT, HEAD, CONNECT, OPTIONS olabilir. Nesnelerin interneti alanında en çok GET ve POST metodları kullanılmaktadır.

GET metodu ile sunucudan bir veri objesi alınmak istenildiği belirtilmektedir. Protokole göre sunucudan gelen cevap bir ön bilgi ve gövde bilgisi içerir. POST metodu ise veriyi istek mesajı ile sunucu tarafına ulaştırılmak için kullanılır. Sunucu tarafından gelen cevaplarda durum belirten bir http kodu ve sunucu tarafından belirlenmiş uygun mesajları içeren gövde bilgisi bulunmaktadır. Sunucu ile istemci arasındaki bağlantı 2 farklı şekilde kurulabilir: kalıcı bağlantı ve çoklu bağlantı. Kalıcı bağlantıda istemci ile sunucu arasında bağlantı kurulduktan sonra bağlantı süre aşımına uğrayana kadar veri aktarımı için bağlantı açık tutulur. Çoklu bağlantı yapısında ise istemci ile sunucu arasında veri aktarımı bilgi aktarımı süresinde aktif tutulmaktadır. Bu sayede sunucunun başka istemciler ile daha verimli bir şekilde haberleşmesi amaçlanmaktadır.

3.1.1.2 Mqtt

Makineler arası iletişim için geliştirilmiş bir mesajlaşma protokolüdür. Yalın yapılı ve düşük kaynak tüketimi nedeniyle nesnelerin internet uygulamalarında ve platformlarında sıklıkla kullanılmaktadır. Mesaj boyutunda küçük olması sayesinde ağırlıklı olarak verimli kullanılmasını sağlamaktadır. Bu protokol yayın - abone yapısında haberleşme sağlamaktadır. Taşıma katmanında TCP protokolünü kullanmaktadır. (Mqtt, 2019)

Mesajlaşma prensibine göre mesajı yayınlayan bir yayıncı diğer aboneleri arayacaktır. Asenkron bir protokoldür, bu sayede mesaj beklenirken iletişim engellenmemektedir. Bu protokole mesaj trafiğini yöneten bir “mesaj dağıtıcısı” (broker) yapısına ihtiyaç duyulmaktadır. Dağıtıcı mesajları yayıncıdan alır ve abonelere iletir. Mesaj gönderimi yapılırken gönderiyi alacak aboneler belirlenirken konu (topic) filtreleri kullanılır. Dağıtıcı yapısı sayesinde alıcılar ve abonelerin birbirini tanımasına gerek yoktur.



Şekil 3.2 MQTT protokolü mesajlaşma şeması (Mqtt Protokolü, 2019)

Birçok farklı mesaj dağıtıcısı opsiyonu bulunmaktadır. Bu çözümlerden RabbitMQ gibi açık kaynaklı bir mesaj dağıtıcısı yapısı platforma entegre edilebilir.

3.1.1.3 Coap

Nesnelerin interneti alanında kullanılan küçük ve düşük kapasiteli cihazlar için tasarlanmış bir uygulama katmanı protokolüdür. HTTP ile benzerlik göstermektedir. HTTP TCP yada UDP üzerinden haberleşebiliyor iken, COAP yalnızca UDP protokolü üzerinden haberleşebilmektedir. Makinalar arası iletişimde kullanıldığı için cihazlar hem alıcı hem sunucu rolünde olabilmektedir. COAP protokolünün 2 farklı modu bulunmaktadır: CON ve NON. CON modunda iken mesajlaşma yapılırken aynı mesaj kodu ile bir mesaj alındığına dair bilgilendirme mesajı (ACK) beklenir. NON modunda ise mesajı alan taraftan herhangi bir alındı bilgisi beklenmez.

HTTP protokolüyle benzer olarak GET, PUT, POST, DELETE gibi temel komutları COAP protokolünde de kullanılmaktadır.

3.1.2 Veri katmanı

Nesnelerin interneti platformunda cihaz ve uygulama verilerinin yanı sıra platform verilerinin saklanabilmesi için bir veri katmanının bulunması gerekmektedir. Veri katmanının cihaz verileri için model bağımsız olması geliştiriciler için kolaylık sağlayabilir. Farklı cihazlardan ve çözümlerden farklı formatlarda veriler gelebilir. Bu nedenle cihaz verisi için ilişkisel olmayan veritabanlarının kullanılması daha doğru olacaktır. Platform kullanıcılarına, konfigürasyonlara ait veriler için de ilişkisel veritabanı kullanılmak istenebilir. Bu gibi durumlarda hem NoSQL hem de SQL veritabanlarının kullanıldığı hibrit çözümler oluşturulur.

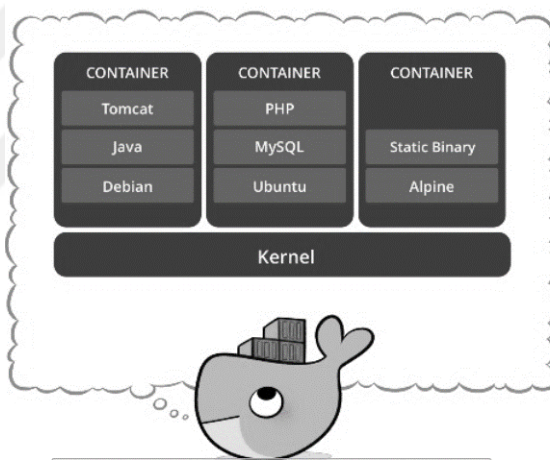
3.1.3 Yönetim katmanı

Yönetim katmanı platform kullanıcıları ve platform modülleri arasında orkestrasyon yapan katmandır. Hem cihazların servislerinin yönetimi hem de platform kullanıcılarının yönetimi için gereklidir. Cihazların yönetimi için servislerin başlatılması, durdurulması, cihazlara komut gönderilebilmesi gibi işlemler yapılabilmelidir. Platform kullanıcılarının erişim haklarına bağlı olarak servislerinin durdurulması gerekebilmektedir. Servislerin yönetimi ile kastedilen platforma bağlanmayı sağlayan çeşitli servislerin erişilebilirliklerinin yönetilmesi ve yeni servisler ile dış dünyadaki uygulamalara verilerin yayınlanmasının kontrolüdür. Platformu kullanan kullanıcıların yönetilmesi kapsamında da kullanıcıların platforma tanıtılması, hesaplarının oluşturulması, gerektiğinde dondurulması, kullanıcılarla servislerin ve cihazların eşleştirilmesi gibi işlemleri yapılabilir olmalıdır.

3.1.4 Çalıştırma katmanı

Servislerin ve/veya uygulamaların çalıştırılıp dış dünyaya belli adresler üzerinden açıldığı katmandır. Bu katman üzerinde çalıştırılan servisler depolanan yada direkt cihazdan alınan verileri farklı sistemlere, cihazlara veya uygulamalara servisler aracılığıyla aktarılabilir. Çalıştırma katmanı farklı teknolojilerle geliştirilmiş servis ve uygulamaları destekleyebilir olmalıdır.

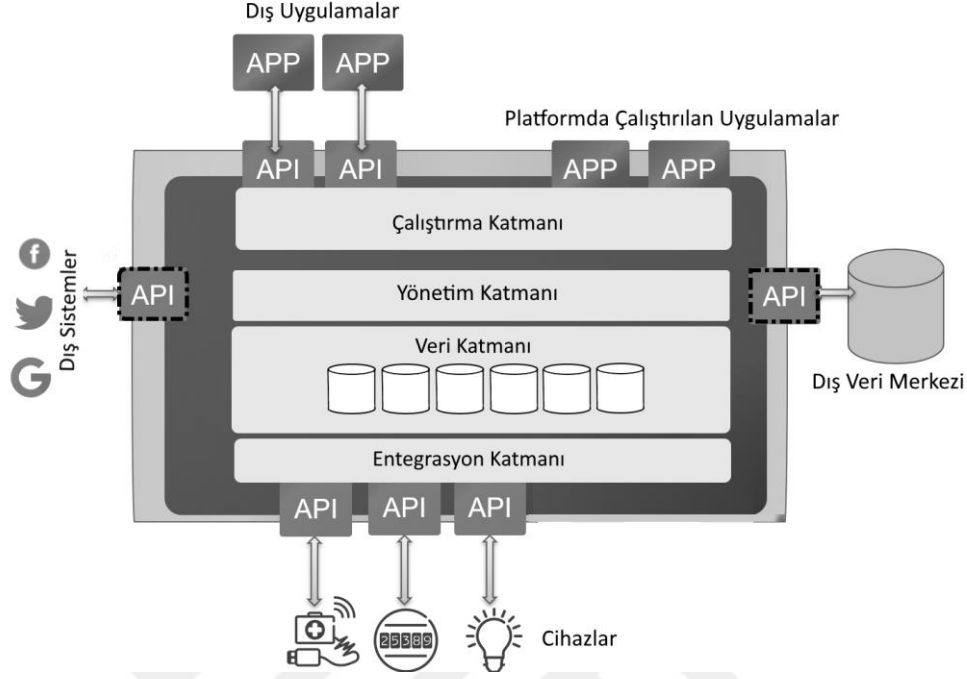
Bulut çözümü üzerindeki farklı çalışma ortamları sağlamak için konteyner yapısı kullanılabilir. Platform, kullanıcıların konteyner imajlarını yüklemelerine izin verilebilir. Konteyner yapısı ile şekil 3.3'te görüldüğü gibi aynı anda farklı farklı teknolojiler kullanılarak geliştirilmiş uygulamalar platform tarafından çalıştırılabilir. Konteyner imajları platforma portal üzerinden yüklenebilir, yönetim katmanı aracılığı ile güvenlik ve erişim kontrollerinden sonra ayağa kaldırılıp erişime açılabilir. Konteyner yapısı platforma esneklik ve güvenlik sağlar.



Şekil 3.3 Konteyner yapısı (Docker, 2019)

3.2 Önerilen Platformun Özellikleri ve Mimarisi

Bölüm 3.1'de açıklamaları yapılmış olan fonksiyonel katmanların ışığında yalın bir model oluşturulmuştur. Bu modelin fonksiyonel katmanları nasıl gerçekleştirdiği, kullanıcılara hangi özellikleri sunduğu bu bölümde incelenecektir. Oluşturulan platform modeli şekil 3.4'te verilmiştir.



Şekil 3.4 Platform mimarisi

Önerilen platformun özellikleri aşağıdaki gibi listelenmektedir:

- Veri modeli kısıtlaması yapmadan her modelde verinin kayıtlanmasını sağlayan veri katmanına sahiptir.
- Kullanıcıların kendi uygulama geliştirme arayüzlerini (API) entegre edebilecekleri bir platformdur. Bu esneklik platformun cihazların bağlantı protokollerini desteklemek için yeni modüllerle genişletilmesini kolay hale getirir.
- Yönetim katmanı sayesinde cihazların, cihazlara ait uygulama programlama arayüzlerinin (API), çeşitli uygulamaların platform üzerinde yaratılabilir ve sunulabilir.
- İhtiyaca göre şekillendirilebilecek servis odaklı mimari ile tasarlanmıştır, istenildiği takdirde lokal sunucu yada bulut üzerinde konuşlandırılabilir bir platformdur.

3.2.1 Platform entegrasyon katmanı

Platform mimarisine bakıldığında entegrasyon katmanını cihazlara ait uygulama programlara arayüzlerinin oluşturduğu görülmektedir. Platform cihazların entegrasyonu ve dış dünya sistemleri (3PP) ile entegrasyon için farklı programlama arayüzlerini destekleyebilir. Cihaz kullanıcılarına istedikleri takdirde cihaz verilerine erişebilmeleri için belli fonksiyonliler sunan programlama arayüzleri vermektedir. Platform gerçekleştiriminde bu özellikten “Genel Programlama Arayüzü” olarak bahsedilecektir. Ayrıca platform kullanıcıları kendi istedikleri fonksiyonliler ve cihazlarına özgü protokoller için istedikleri bağlantıyı sağlayabilecek programlara arayüzlerini de

platforma yükleyip çalıştırabilmektedirler. Bu sayede platformun esnekliği sağlanmaktadır, entegrasyon kısıtlamalarını da yok etmektedir. Platform gerçekleştiriminde bu yapı “Özelleştirilmiş Programlama Arayüzü” olarak isimlendirilmiştir.

Platform gerçekleştiriminde HTTP protokolü üzerinden haberleşebilen cihazlar için genel programlama arayüzü üzerinden aşağıda belirtilen fonksiyonalteler sağlanmıştır:

- Cihazdan veri gönderim servisi (PushData)
- Depolanmış cihaza ait son veriyi alma servisi (GetLatestData)
- Depolanmış cihaz verisinden belirtilen son N miktarda veriyi alma (GetDataWithCount)
- Depolanmış cihaz verisinden belirli bir tarih aralığı içinde olan verileri alma (GetDataWithInterval)

Özelleştirilmiş Programlama Arayüzleri sayesinde platform diğer protokollerini de destekleyebilir. (MQTT, CoAp, Web-socket vs.) Bu sayede kural motorlarının da entegrasyonunu desteklemektedir.

3.2.2 Platform veri katmanı

Platformun veri katmanı planlanırken veri modeli bağımlılığından arındırılması amaçlanmıştır. Bu nedenle veri katmanında ilişkisel olmayan veritabanı teknolojilerinden MongoDB kullanılarak oluşturulmuştur. Platforma bağlanabilen her bir nesne, uygulama yada dış sistem için ayrı bir veritabanı oluşturulması planlanmıştır. Bu sayede farklı kullanıcıların verileri kendilerine ait erişim anahtarları ile sadece kendileri tarafından kullanılan veritabanında tutulmuş olur.

Servis isimleri ve port bilgileri gibileri gibi platform yönetiminde kullanılan veriler için ise performansın da iyi olabilmesi için lokal disk veritabanı kullanımı yapılmaktadır.

Platform istenildiği takdirde çeşitli analitik çözümler ve büyük veri modülleri ile entegre edilebilir.

3.2.3 Platform yönetim katmanı

Platform yönetim katmanında platform ile etkileşen aktörler için farklı yönetim servisleri bulunmaktadır. Bu aktörler cihazlar, platform kullanıcıları ve uygulamalar olabilir. Bu yönetim servisleri bir portal üzerinden sunulabilir. Bu katmanın amacı platformu oluşturan modeller arasında orkestrasyonu sağlamaktır. Başlıca sağlanan yönetim katmanı servisleri şöyle sıralanabilir:

- Cihazların platforma tanıtılması
- Cihazın platform erişiminin hayat döngüsünün yönetimi
- Programlama arayüzlerinin hayat döngüsü yönetimi

- Platform kullanıcılarının yönetimi
- Cihazların kuzey yönü bağlantı noktalarının tayini (northbound interaction)

3.2.4 Platform çalıştırma katmanı

Platform çalıştırma katmanı, servislerin ve uygulamaların dış dünyaya açılabilir olması için gerekli olan bir çalıştırma ortamını ifade etmektedir. Bu katman sayesinde hazırlanan programlama arayüzleri cihaz ve sistem bağlantılarını çalışır ve ulaşılabilir halde tutmaktadır.

Platform prototipi kullanıcılarına node.js teknolojisi kullanılarak geliştirilmiş uygulama ve servisleri çalıştırma ve dış dünyaya sunma imkanı vermektedir.

Platforma kullanıcı tanımı yapılmasıyla beraber, platform yönetim servisleri bu kullanıcıya ait programlama arayüzü ve uygulamalarını yükleyip çalıştırabileceği bir dizin yapısı oluşturmaktadır. Node.js ile oluşturulmuş projeler servisler aracılığı ile platform üzerinde dizin yapısına gönderilmektedir. Platform tarafından kullanıcıya özel belirlenen port numarası üzerinden de özelleştirilmiş servisler kullanıcıların verdiği programlama arayüzü ismi ile ulaşılabilir hale gelmektedir. Ayrıca cihazlar için hali hazırda platform tarafından verilen servisler (genel programlama arayüzü servisleri) kullanılıyorsa, bu yapı kullanıcının dizinine kopyalanır ve kullanıcıya özgü port numarası üzerinden ortak erişim yolları aracılığı ile sunum sağlanmaktadır.

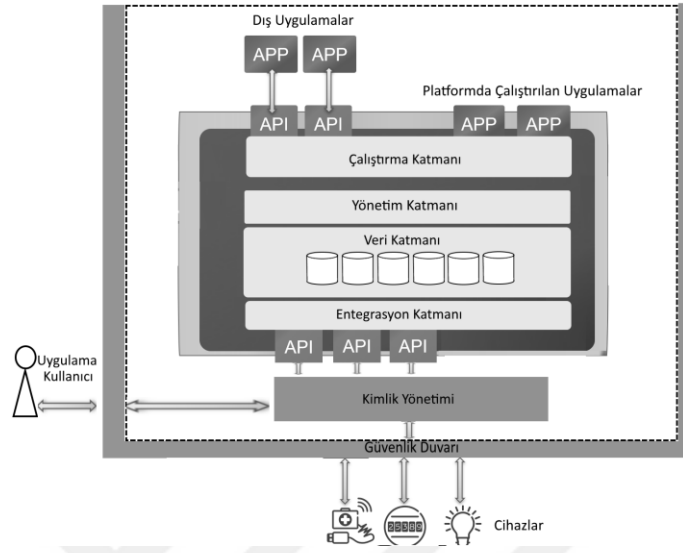
3.3 Platform Mimarisi ile Farklı Kurulum Opsiyonları

Platform çekirdek yapısı dolayısıyla farklı ölçeklendirme opsiyonlarını sunmaktadır. Kullanım senaryosuna bağlı olarak 2 farklı modellemeden bahsedilecektir:

Opsiyon 1

Örnek senaryoda, çekirdek sistem bir fabrikada üretimde kullanılan bantlardaki sensörler ile gelen verileri depolayıp bu verilerin servisler aracılığı ile hat üzerindeki çeşitli ekranlarda ve çalışanların tabletlerinde görüntülenebilmesini istenmektedir. Fabrika verilerinin kendi bünyesindeki sunucularda tutulmasını ve verisinin dış dünya ile paylaşılmasını istememektedir.

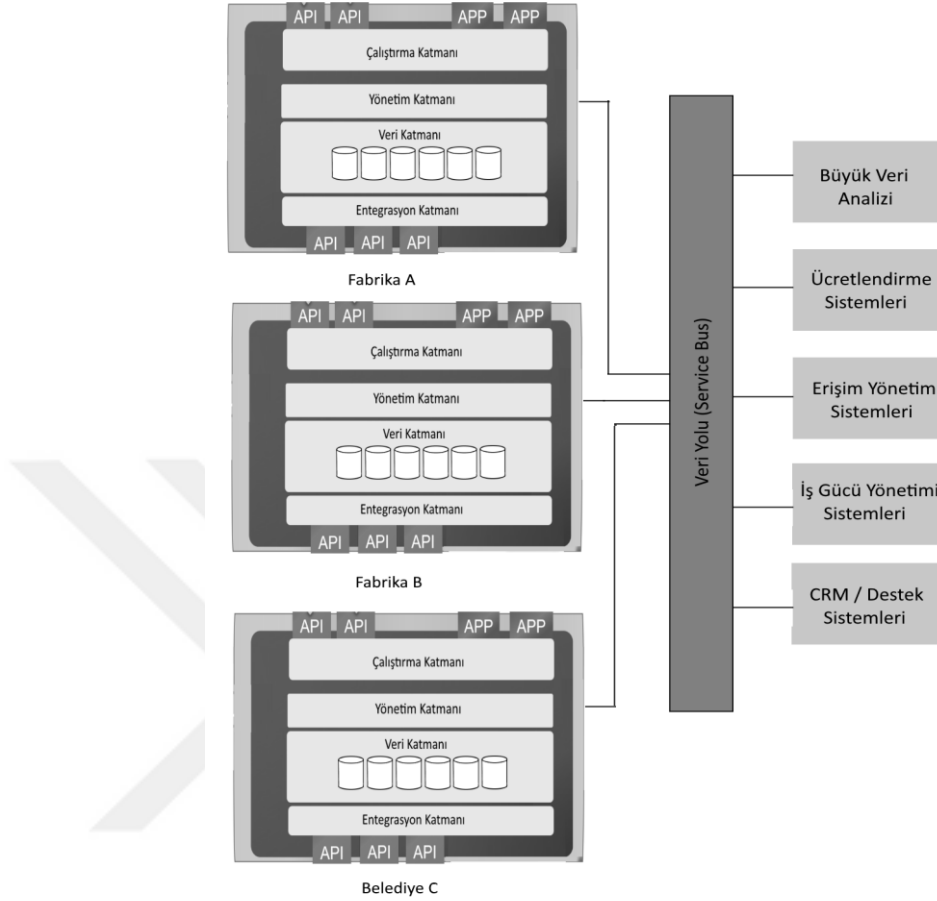
Kurulum fabrika sunucularına yapılabilir olmalıdır. Fabrikada kullanılan cihazların ürettikleri verilerin boyutu ve sıklığına göre sunucuların bellek, ön bellek, işlemci hızı gibi özellikleri belirlenmelidir. Platform kullanıcılarının platforma bir kimlik yönetimi kontrolünden sonra geçmesi gerekir. OpenAM gibi açık kaynak kodlu bir yapı modele entegre edilebilir. Güvenlik gereğiyle de sistem bir güvenlik duvarı ile korunmalıdır. Sadece lokal ağ üzerinde açık bir sistem kurulumu düşünülebilir. Veri miktarına göre veritabanı boyutu seçilir. Bu şekli ile sistem Şekil 3.5'teki gibi kurulabilir. Yük dengelemesi ve yedekleme opsiyonları modele dahil edilmemiştir.



Şekil 3.5 Opsiyon 1- Tekil kurulum modeli

Opsiyon 2

Örnekte B2B2C (Business to business to customer) yapısı yani şirketten şirkete şirketten müşteriye satış için bir model belirlenecektir. Senaryoda bir servis sağlayıcı şirket iş ortaklarına nesnelerin interneti platformu sağlayacaktır. İş ortakları 3 farklı alanda faaliyet gösteren aynı zamanda rekabet içinde olan şirketlerdir. Bu şirketler kendilerine özgü kontrol portalına ve veri depolama yapılarına sahip olmak istemektedir. Servis sağlayıcı ise bu şirketlerin verilerinden onların izinleri doğrultusunda analizini yapmak istemektedir. Servis sağlayıcı ayrıca platform kullanımını ücretlendirme ve platform ile ilgili son kullanıcılara müşteri desteği verecektir. Tüm kullanıcılar için de kimlik yönetimi yapısı platform sağlayıcının kontrolündedir. İş ortaklarının istediği özerk yapı için çekirdek mimarinin çoğul biçimde kullanılması önerilebilir. Örnek model şekil 3.6'da verilmiştir. Çekirdek mimari yapısı ile her şirketin veri ambarı ayrı ve kendine özgüdür. Şirketlerin isteğine göre çekirdek mimari kendi sunucularına yada buluta kurulabilir.



Şekil 3.6 Opsiyon 2- Servis sağlayıcı kurulum modeli

3.4 Platform Gerçekleştirimi

Platform gerçekleştiriminde kullanılan teknolojiler ve yazılımsal yapılar bu bölümde açıklanmaktadır.

3.4.1 Kullanılan teknolojiler

3.4.1.1 Node.js

Node.js javascript tabanlı bir açık kaynak bir programlama dilidir. Farklı çalışma ortamlarında çalıştırılabilir (cross-platform). Node.js sayesinde javascript dili tarayıcı tarafı programlama dilinden sunucu tarafı uygulama programlama diline dönüşmüştür. (node.js, 2019) Performans açısından güçlü bir dildir. LinkedIn şirketinin verilerine göre yazılım teknolojisi olarak Rails'ten Node.js diline geçmesiyle performansın bazı işlemlerde 20 kat artarken, sunucu maliyetleri ise 10'da bir oranında azalmıştır. Node.js olaya dayalı asenkron fonksiyonlar tarafından işletilmektedir. Eş zamanlı milyonlarca bağlantıyı yönetebilecek şekilde ölçeklendirilebilir bir

yapısı vardır. Milyonlarca cihazın aynı anda iletişime geçmek istediği bir dünyada platform kapasitelerinin de buna elverişli olması gerekmektedir. Bu nedenle node.js nesnelerin interneti platformları için iyi bir teknolojik tercih olmaktadır.

3.4.1.2 Sails Yazılım Geliştirme Ortamı

Sails, Node.js tabanlı açık kaynak bir yazılım geliştirme ortamıdır. Kurumsal düzeyde uygulama geliştirmeye olanak sağlayan ortam Model-Bakış-Kontrolcü (Model-View-Controller / MVC) tasarım desenini kullanmaktadır. Diğer MVC tasarım desenlerini destekleyen dillere göre veri merkezli geliştirilmiş yapısıyla modern bir web uygulaması geliştirme ortamı sağlamaktadır. Özellikle gerçek zamanlı mesajlaşma uygulamaları gibi uygulamaların geliştirilmesinde de sıklıkla tercih edilmektedir.

Sails geliştirme ortamı Express.js desteğiyle önyüz geliştirmeye olanak vermektedir. Uygulama Programlama Arayüzü (API) desteği sayesinde otomatik olarak Restful JSON modelleme yapmaktadır. Bu sayede HTTP ve websocket desteği sunmaktadır. Sunucu tarafı programlama desteği ile binlerce kullanıcıyı yönetebilmekte, HTML sayfalarını sunabilmekte ve veritabanı ile iletişime geçebilmekte ve uygulama programlama arayüzü çalıştırabilmektedir. (Sails.js, 2019)

Sails geliştirme ortamının en önemli özelliklerinden birisi REST servislerin rotalarını otomatik olarak bağlamasıdır. Config/routes.js altında belirlenmiş servis rotalarını otomatik olarak kontrolcü sınıflar altındaki servis geliştirimlerine bağlar.

3.4.1.3 MongoDB

MongoDB farklı çalıştırma ortamlarında kullanılabilen döküman tabanlı bir veritabanıdır.

Nesnelerin interneti alanında kullanılacak platform gereksinimleri düşünüldüğünde, cihaz ve kullanım senaryosu çeşitliliği nedeniyle veri modeli esnekliği veri depolama teknolojisi seçiminde en önemli kriterlerden biri olmaktadır. Bu nedenle NoSQL veri teknolojileri incelenmiştir.

MachinaResearch'un 2014 yılında yaptığı çalışmaya göre Nesnelerin İnternetini Makineler arası haberleşme teknolojisinden ayıran temel özellik verinin düzensizleşmesi ve miktarının artmasıdır. Bu nedenle de veri analizinin de büyük veri depolama teknolojilerinin desteği ile nesnelerin interneti tanımı oluşmuştur. Rapora göre veri depolama teknolojisinin gereksinimleri listelenmiştir (Emil, 2014):

- Farklı cihaz ve veri modellerini destekleyebilme
- Esnek ve kolaylıkla başka ihtiyaçlara göre modellenip adapte edilebilir olması
- Analitik yetkinlikler

Nesnelerin interneti platformlarında sıklıkla tercih edilen SQL veritabanlarının kullanılmamasının nedeni, verinin SQL veritabanlarında olduğu gibi düzenli yapıda olmamasıdır. Bu düzensiz yada yarı düzenli veriler grafik tabanlı (Neo4J, Virtuoso), anahtar-değer tabanlı (Riak, DynamoDB), geniş kolon formatlı (Accumulo, Cassandra, HBase) yada daha sıklıkla tercih edilen Doküman eğilimli (MongoDB, CouchDB) NoSQL teknolojileri ile saklanabilmektedir. Bu teknolojiler arasında MongoDB'nin incelemeye başlanmasının başlıca nedenleri açık kaynaklı olması geliştirmeye elverişli olması, ölçeklendirmeye elverişli olması, sektörde de sıklıkla kullanım örneğinin bulunmasıdır. MongoDB'nin diğer NoSQL veritabanlarına göre avantajları şöyle listelenebilir (MongoDB,2019):

- Sorgu(query) desteği: Birçok NoSQL çözümü veriye sadece anahtarlar (key) üzerinden erişme olanağı sağlarken, MongoDB istenilen alanlara ve belirli aralıklara (range query) göre, ayrıca düzenli ifadelerle (regular expression) de sorgulama imkanı sunmaktadır.
- İkincil(secondary) indeks desteği: İstenilen alanlara göre sorgulamanın yanı sıra, bu alanları ikincil indeks olarak tanımlayabilmek, bu da veriye daha performanslı erişim imkanı sağlamaktadır.
- Komut veren-Bağımlı Replika Desteği: Yazma ve okuma işlemlerini ayrı sunuculara yönlendirebilme, komut veren sunucu erişilemez olduğunda bir bağımlı sunucuyu komut veren sunucu olarak çalıştırabilme
- Paylaştırma: Büyük ölçekli verilerin sunucular arasında paylaştırılması özelliği bulunmaktadır.
- Dağıtık veri işleme
- Pek çok yazılım dili için sürücü (driver) desteği

MongoDB'de her kayıt doküman olarak tutulmaktadır. Veriler JSON benzeri Binary JSON (BSN) formatında saklanmaktadır. Tanımlamaların ilişkisel veritabanları ile karşılaştırması aşağıdaki tabloda verilmiştir (MongoDB, 2019):

Tablo 3.1 MongoDB terimler eşlemesi

İlişkisel Veritabanı Terimleri	MongoDB Terimleri
Database	Database
Tablo	Koleksiyon
Satır	Doküman
Sütün	Alan
İndeks	İndeks
Birincil Anahtar	Birincil Anahtar

3.4.2 Platform Yazılımının İncelenmesi

Platform yönetim yazılımı iki ana kontrol yazılımdan oluşmaktadır. Proje Sails.js yazılım çatısı üzerinde geliştirilmiştir. Kullanıcılara ve cihazlara ait veriler MongoDB üzerinde tutulmaktadır.

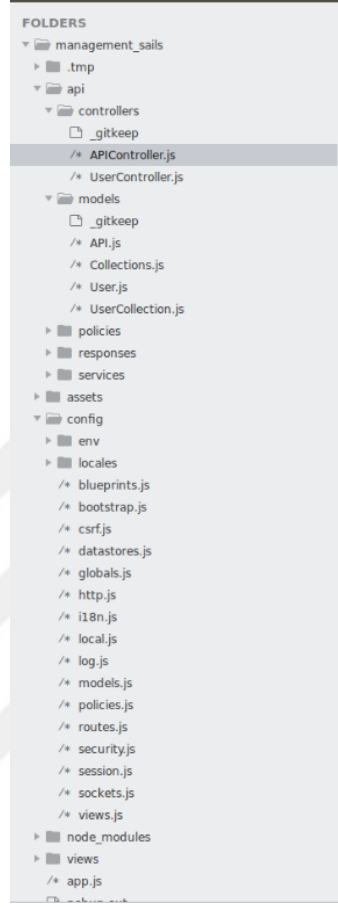
Kontrol bölümlerinden biri platform kullanıcılarını yönetmek için oluşturulmuştur. Başlıca fonksiyonalteler şöyle sıralanabilir:

- Kullanıcı yaratma
- Kullanıcı hesap durumunu değiştirme
- Kullanıcı bilgilerini getirme
- Tüm kullanıcıları listeleme
- Kullanıcıya ait veritabanı anahtarını getirme

Kullanıcıların cihazlarını kontrol etmek amacıyla yaratılmış kontrol bölümü aşağıdaki fonksiyonaltelere sahiptir:

- Cihazlar için platform tarafından verilen standart fonksiyonlara sahip bir uygulama programlama arayüzü yaratımı (API)
- Cihazlar için özelleşmiş fonksiyonlara sahip programlama arayüzü yaratımı (API)
- Programlama arayüzü başlatma
- Programlama arayüzünü durdurma
- Programlama arayüzünü deaktive etmek
- Programlama arayüzü detaylarını getirme

Yönetim yazılımı Sails.js proje yapısı şekil 3.7’de verilmiştir:



Şekil 3.7 Yönetici yazılım proje yapısı

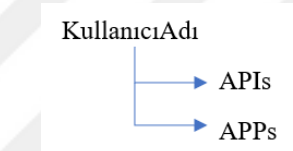
3.4.2.1 Kontrolcüler

Sails.js platformunda yaratılan projeler Model-Bakış-Kontrolcü (MVC) tasarım desenini kullanır. Bu yapıya göre Kontrolcüler “Controllers” klasöründe bulunmaktadır. Kontrolcülerde modellerle iletişime giren ve istemciye gerekli cevapları hazırlayan javascript yazılım parçaları bulunmaktadır. “APIController.js” cihazlar için tanımlanan programlama arayüzlerinin yönetim fonksiyonlarının yer aldığı yapıdır. “UserController.js” ise kullanıcı yönetim fonksiyonlarını kapsar. Bu yapı ile platformun yönetim katmanı gerçekleşmiş olur. Kontrolcüler servisler gibi davranırlar. Ana proje içerisinden buraya gelen program akışında REST çağrılarının gerekli değişkenleri kontrol edilir, gerekli hata mesajları hazırlanır. Fonksiyonalliteye göre programın yapması gereken adımlar bu bölümde tanımlanmıştır.

UserController.js

Kullanıcı yaratma operasyonları için takip edilen adımlar şöyle sıralanabilir:

- Servis özgün kullanıcı adı kullanılarak çağrılır. Kullanıcı adı ile daha önceden bir hesap oluşturulup oluşturulmadığı kontrol edilir ve oluşturulmadıysa akışa devam edilir.
- Yaratılan kullanıcı için standart programlama arayüzü kullanılmak istenildiğinde erişim için kullanılacak olan bir port numarası belirlenir. Kullanıcı ve port eşlemesi lokal disk üzerinde saklanır.
- Kullanıcının cihaz ve uygulama verilerini depolaması için MongoDB üzerinde o kullanıcı için bir veritabanı tahsis edilecektir. Bu veritabanı için bir anahtar oluşturulur.
- Yaratılan kullanıcı bilgileri MongoDB üzerindeki Admin veritabanında kayıt altına alınır.
- Yaratılan kullanıcıya ait sunucu üzerinde uygulamalarını ve programlama arayüzlerini ait kodlamaları tutacağı ve çalıştırabileceği dosya izinleri yaratılır. Oluşturulan izinler şekil 3.8'deki gibidir:



- Yaratılan kullanıcıya platforma ait standart fonksiyonalliteyi kullanmasını sağlayacak programlama arayüzüne ait kaynak kodlar “APIs” dizinine kopyalanır. Bu programlama arayüzüne ait kod parçasındaki konfigürasyonlar kullanıcıya ait dizine konulmadan önce kullanıcıya özel verilmiş port numarası ile değiştirilir.
- Sails projesinin üzerindeki yerel disk yapısı ile yönetim servislerinin konfigürasyonları kayıt altına alınmaktadır. Sistemin çalışma performansını arttırmak amaçlı konfigürasyon verileri için bu yerel disk kullanılmıştır. Bir kullanıcı yaratıldığında yerel disk üzerindeki “collections” dosya veritabanında kayıt altına alınmaktadır. Bu sayede o kullanıcıya ait port bilgisi sistemde kayıt altına alınmış olur.
- Tüm işlemler tamamlandıktan sonra yeni yaratılan kullanıcıya ait kullanıcı adı ve veritabanı anahtarı platform arayüzüne dönecek şekilde tasarım yapılmıştır.

Kullanıcıya ait hesap durumunu değiştirmek için takip edilen adımlar şöyle sıralanabilir:

- Servis çağrılırken kullanıcı adı ve durum verisi belirtilir. Kullanıcı aktif ise deaktif, kullanıcı deaktif ise aktive etme akışı gerçekleştirilecektir.
- Yönetici veritabanına bağlanılır ve “Users” koleksiyonundaki kayıt bulunur, durum bilgisi değiştirilir.

Kullanıcıya ait bilgileri getirmek için işletilen adımlar şöyle sıralanabilir:

- Özgün kullanıcı adı ile servis çağrılır.
- Kullanıcı adı yönetici veritabanında aranır. Kayıt bulunursa durum ve port bilgisi geri döndürülür. Kayıt bulunamazsa hata mesajı döndürülmektedir.

Kullanıcıların hepsini getirmek için işletilen adımlar şöyle sıralanabilir:

- Yönetici veritabanına bağlanılır. “Users” koleksiyonunda kayıt var ise tüm kayıtlar döndürülür. Kayıt yok ise hata mesajı oluşturulur.
Akışa ait kodlamalar aşağıdaki gibi yapılmıştır:

Kullanıcıya ait veritabanı erişim anahtarını getirmek için işletilen adımlar şöyle sıralanabilir:

- Servis özgün kullanıcı adı ile çağrılır.
- Servis çalışmasının hızlı olması için kullanıcı adı öncelikle lokal disk veritabanındaki kayıtlarda aranmaktadır. Burada kayıt yok ise veritabanı ile iletişime girilmeden hata mesajı ile süre. Sonlandırılır.
- Kullanıcı adı lokal disk kayıtlarında var ise, yönetici veritabanına bağlanılır. “Users” koleksiyonundan erişim anahtarı verisi döndürülür.

APIController.js

Platformun sağladığı, standart fonksiyonaliteyi sahip programlama arayüzünün yaratılmasını ait akış adımları şöyle sıralanabilir:

- Servis özgün kullanıcı adı ile çağrılır.
- Servis çağrılırken bu programlama arayüzünü kullanacak olan cihaz ile ilgili tanımlanmak istenen bilgiler de gönderilir. Cihaz bilgileri bağlantı hiyerarşisine uygun olarak sensör-cihaz-ağ geçidi bilgileri ile kayıtlanabildiği gibi, yalnızca nesne adı ile de oluşturulabilir.
- Kullanıcı ilk oluşturulurken programlama arayüzlerinin kaynak kodların yüklenmesi için özgün kullanıcı adı ile oluşturulmuş dizinin altında “APIs” dizini oluşturulmuştur. Bu dizinin erişim yolu kullanıcı adı bilgisi ile kaynak kod içerisinde oluşturulur.
- Tüm kullanıcılara ait programlama arayüzlerine ait konfigürasyon verilerinin tutulduğu lokal disk veritabanı yolu belirtilir
- Bir kullanıcıya ait standart fonksiyonaliteyi kullanan tüm programlama arayüzleri için konfigürasyon verileri kullanıcının dizinine kopyalanan standart programlama arayüzünün içerisindeki lokal disk veritabanında tutulacaktır. Bu veritabanı lokasyonu da belirtilir.
- İlk olarak kullanıcı adının sistem lokal disk veritabanında olup olmadığı kontrol edilir. Yok ise hata mesajı döndürülerek süreç sonlandırılır. Var ise akışa devam edilir.
- Kullanıcı var ise lokal disk veritabanındaki konfigürasyon bilgisinden kullanıcıya ait port numarası alınır.

- Lokal disk üzerinden kullanıcının daha önce bu isimle oluşturduğu başka bir programlama arayüzü olup olmadığı kontrol edilir. Programlama arayüzü ismi özgün olmalıdır. Eğer kayıt bulunursa hata mesajı gönderilir ve akış sonlandırılır, bulunmazsa akışa devam edilir.
 - Sistem lokal disk veritabanına yeni API tanımlaması eklenir. Sonrasında bu kullanıcıya ait tüm programlama arayüzü isimleri sıralı bir şekilde alınır. Standart fonksiyonalitye sahip olan programlama arayüzleri “isGenericApi” parametresi ile belirlidir. Bu kullanıcıya ait programlama arayüzleri listesi kullanıcıya ait dizinde oluşturulan standart API klasöründeki lokal disk db ye yedeklenir.
 - Kullanıcıya ait “APIs” dizini altındaki “Standart_API” kaynak kodu çalıştırılır.
 - Programlama arayüzüne ait tanımlama bilgisi yönetici veritabanına bağlanılarak “Definitions” koleksiyonuna eklenir.
 - Servisin cevabı olarak oluşturulmuş olan programlama arayüzüne ait erişim bilgilerini içeren bir JSON yapısı döndürülür.
- Akışa ait kodlamalar aşağıdaki gibi yapılmıştır:

Uygulamanın veya özelleştirilmiş fonksiyonalitye sahip programlama arayüzünün yaratılmasını sağlayan akış adımları şöyle sıralanabilir:

- Servis özgün kullanıcı adı ile çağrılır.
- Bir cihaz programlama arayüzüne ait verilmek istenen tüm bilgiler servis çağrısında gönderilebilir.
- Bu servis ile hem özelleşmiş programlama arayüzü hem de uygulama yükleme işlemleri yapılabilmektedir. İki süreç farklı dizinlerde gerçekleştirilir. Bu ayrım servis çağrımında gönderilen “appApiName” değişkenine bakılarak karar verilir. Bu değişken tanımlı geliyorsa uygulama, gelmiyorsa programlama arayüzü dizinine gidilir.
- Lokal disk veritabanından daha önceden bu isim ile bir tanımlama yapıp yapılmadığı kontrol edilir. Var ise süreç sonlandırılır, yok ise devam edilir.
- Yönetici veritabanına bağlanılır ve standart olmayan programlama arayüzleri listelenir. Uygun port numarası bulunur.
- Kullanıcının yüklemek istediği programlama arayüzü bir portal aracılığı ile sunucuya yüklendiği varsayılmaktadır. Yüklendiği lokasyon “UploadedAPIs” klasörüdür. Buraya programlama arayüzü ismi ile yüklenmiş olan kaynak kod zip dosyası, kullanıcıya ait dizine çıkartılır.
- Kaynak kod içerisindeki server.js dosyasındaki port değişkeni belirlenen port numarası ile değiştirilir.
- Programlama arayüzü bilgileri yönetici veritabanındaki “Definitions” koleksiyonuna da eklenir.
- Kullanıcı ve port bilgisi aynı zamanda lokal disk veritabanına da kayıt edilmektedir.
- Kullanıcıya ait programlama arayüzü direkt olarak kullanıcı dizinindeyse yalnızca port bilgisi kayıtlamaları ve yönetici veritabanı kayıtları yapılır.
- İki durumda da bu programlama arayüzüne erişim için kullanılacak olan adres bilgisi, oluşturulmuş koleksiyon adı ve kullanıcı adı verileri döndürülür.

Platformun sağladığı, programlama arayüzü silme servisine ait akış adımları şöyle sıralanabilir:

- Servis özgün kullanıcı adı ile çağrılır. Silinmek istenen programlama arayüzü ismi, tipi ve standart programlama yüzü olup olmadığını belirten “isGenericApi” parametresi servis çağrımında gönderilmelidir.
- Silinmek istenen programlama arayüzünün lokal disk veritabanı içinde olup olmadığı kontrol edilir. Varsa bu programlama arayüzünün port bilgisi alınır.
- Bu port numarası ile çalışmakta olan bir süreç olup olmadığı kontrol edilir. Eğer çalışan bir süreç varsa bu süreç durdurulur.
- Programlama arayüzüne ait olan veritabanı kayıtlarında durum bilgileri deaktive durumuna çekilir.

Platformun sağladığı, programlama arayüzü çalıştırma servisine ait akış adımları şöyle sıralanabilir:

- Servis özgün kullanıcı adı ile çağrılır. Silinmek istenen programlama arayüzü ismi, tipi ve standart programlama yüzü olup olmadığını belirten “isGenericApi” parametresi servis çağrımında gönderilmelidir.
- Çalıştırma servisi hem uygulama hem programlama arayüzü için ortak tasarlanmıştır. Tip bilgisine göre farklı dizinlere yönlendirme yapılmaktadır.
- Programlama arayüzünün varlığı kontrol edilir.
- Lokal disk veritabanı üzerinden programlama arayüzüne ait port bilgisi alınır.
- Port üzerinde programlama arayüzü çalışıp çalışmadığına bakılır. Çalışmıyorsa programlama arayüzü o port üzerinde ayağa kaldırılır ve durum bilgisi mesajı paylaşılır.

Platformun sağladığı, programlama arayüzü durdurma servisine ait akış adımları şöyle sıralanabilir:

- Servis özgün kullanıcı adı ile çağrılır. Durdurulmak istenen programlama arayüzü ismi servis çağrımında gönderilmelidir.
- Durdurulmak istenen programlama arayüzü varlığı lokal disk veritabanı üzerinden kontrol edilir. Böyle bir kayıt yoksa hata mesajı döndürülür, varsa akışa devam edilir.
- Lokal disk üzerinden kayıtlı port bilgisi alınır.
- O port üzerinde çalışan süreç yoksa sürecin çalışmadığına dair bir bilgilendirme mesajı verilir. Çalışan süreç var ise süreç durdurulur.

Platformun sağladığı, programlama arayüzüne ait bilgileri listeleme servisine akış adımları şöyle sıralanabilir:

- Servis özgün kullanıcı adı ile çağrılır. Detayları istenen programlama arayüzü ismi servis çağrımında gönderilmelidir.
- Öncelikle bu programa arayüzünün varlığı lokal disk veritabanından kontrol edilir. Böyle bir kayıt yoksa hata mesajı döndürülür. Var ise akışa devam edilir.

- Lokal disk üzerinden programlama arayüzüne ait port bilgisi alınır.
- Port bilgisi ile programlama arayüzünün çalışıp çalışmadığı kontrol edilir.
- Programlama arayüzünün MongoDB üzerindeki koleksiyonları üzerinden son kayıt zamanı ve kayıtların boyutu bilgileri alınır.
- Alınan tüm bilgiler oluşturulan JSON ile cevap olarak döndürülür.

3.4.2.2 Modeller

Model klasöründe veritabanı ile iletişime geçerken kullanılan veri modelleri belirlenir. Sail.js platformu Waterline kütüphanesini kullanmaktadır. Waterline sayesinde veritabanı teknolojisinden bağımsız olarak oluşturulan yapı ile farklı veritabanlarına bağlantıya olanak sağlamaktadır.(Waterline,2019) Bu yapılar Obje İlişkilendirme (ORM) modelini kullanır. Proje yapısında “Models” klasöründe tanımlanan modeller aşağıda detaylı incelenmiştir:

API.js: Cihazlar için verilen programlama arayüzlerine ait veritabanında tanımlanma verilerinin yapısını göstermektedir. Bu programlama arayüzünü kimin kullandığı, platform standart fonksiyonalitesine sahip program parçası ile yaratılıp yaratılmadığı bilgileri tutulmaktadır. Bağlanan obje için bağlı bulunduğu sensörler, cihazlar, geçiş kapısı gibi yapılara ait bilgiler de var ise tutulmaktadır. apiType verisi işletilen program parçası bir programlama arayüzü ise “api”, bir uygulama ise “application” değerini almaktadır.

Collection.js: Platform kullanıcısının sisteme bir cihaz bağlaması isteği sonucunda bu cihaz verilerinin tutulması için bir MongoDB koleksiyonu tanımlanır. Bu açılan koleksiyona ve onun kullanıcısına ait kurulan yapının kaydının tutulması için Collection.js içindeki model kullanılarak yönetim verisi tutulur. Bu kullanıcı-koleksiyon ilişkisinin aynı zamanda programlama arayüzüne ulaşım için bir de port bilgisi bulunmaktadır. Sağlanan programlama arayüzünün platformun standart fonksiyonalitelerini kullanarak mı oluşturulduğu yada platforma dışarıdan özel geliştirilmiş bir programlama arayüzü yazılımı mı eklendiğini belirten “isGenericApi” değişkeni bulunmaktadır.

User.js: Platform kullanıcılarına ait kayıtların düzenli bir şekilde tutulabilmesi için bu model kullanılmıştır. Kullanıcıya platform üzerinde o kişiye özgü bir port numarası ile standart programlama arayüzlerine erişim hakkı verilmektedir. Bu nedenle port numarası bilgisi önemlidir.

UserCollection.js: Oluşturulmuş olan tüm programlama arayüzleri kayıt altına alınmaktadır. Bu kayıtlar içinde koleksiyon adları kullanılmaktadır. Koleksiyon adı özgün olmalıdır.

Config/routes.js : Bu bölümde programa ait erişilebilecek olan URL’ler ile program içindeki kontrolcüler ve onların fonksiyonaliteleri arasında bir eşleştirme yapılmaktadır.

```

'post /ManagementApi/User/create' : 'UserController.create',
'post /ManagementApi/User/changeStatus' : 'UserController.changeStatus',
'get /ManagementApi/User/get' : 'UserController.get',
'get /ManagementApi/User/getAll' : 'UserController.getAll',
'post /ManagementApi/User/getDbKeyForUser' : 'UserController.getDbKeyForUser',

'post /ManagementApi/Api/createStandardApi' : 'ApiController.createStandardApi',
'post /ManagementApi/Api/createCustomApi' : 'ApiController.createCustomApi',
'post /ManagementApi/Api/deleteApi' : 'ApiController.deleteApi',
'post /ManagementApi/Api/startApi' : 'ApiController.startApi',
'post /ManagementApi/Api/stopApi' : 'ApiController.stopApi',
'post /ManagementApi/Api/getApiDetails' : 'ApiController.getApiDetails'

```

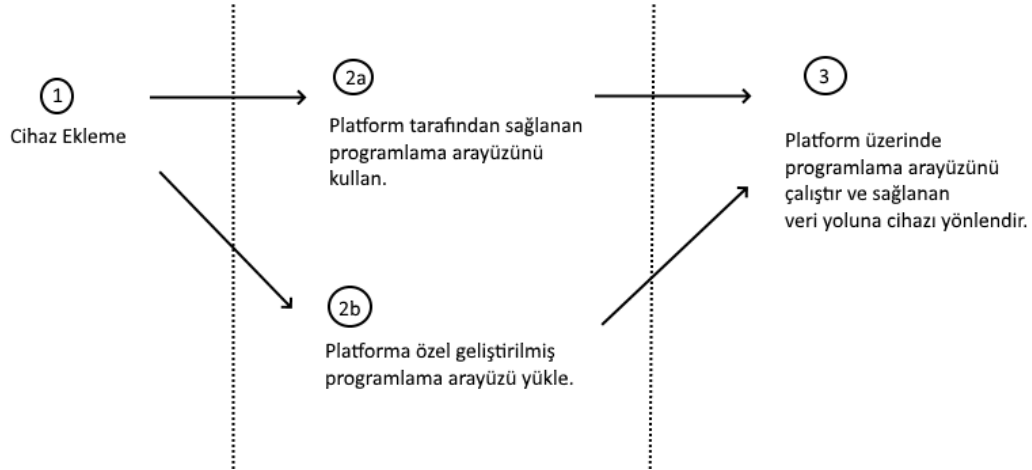
Şekil 3.9 Yazılımdaki kontrolcüler ile URL eşlemeleri

App.js: Program için başlangıç noktasıdır. Proje bu dosya üzerinden ayağa kaldırılmaktadır.

3.5 Platform Genel Programlama Arayüzü Yazılımı

Yalın platform gerçekleştirmesinde kullanıcılara cihazlarını programlama arayüzü geliştirme yani kodlama gereğinde bulunmadan en çok kullanılan servisleri içeren bir programlama arayüzü sağlanmaktadır. Böylece kullanıcılar platformu kullanmaya cihaz tanımlamalarını yapar yapmaz başlayabilirler. Kullanıcıların teknik olarak programlama bilmelerine gerek olmadan cihazlarını platforma bağlamaları amaçlanmıştır.

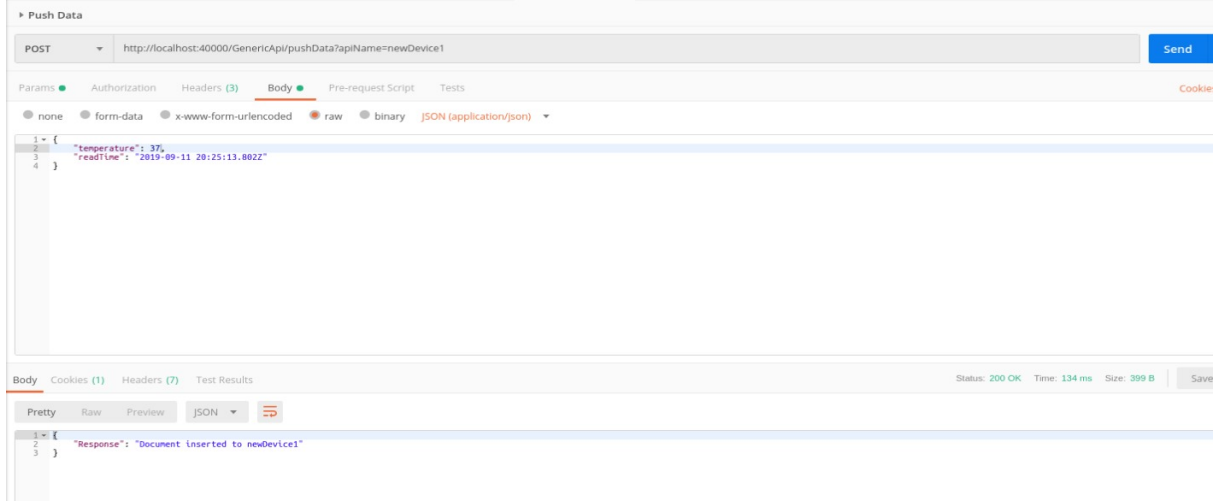
Kullanıcı cihazını platforma bağlamak istediğinde karşılaşması planlanan akış şekil 3.10'daki gibidir:



Şekil 3.10 Cihaz ekleme yöntemi opsiyonları

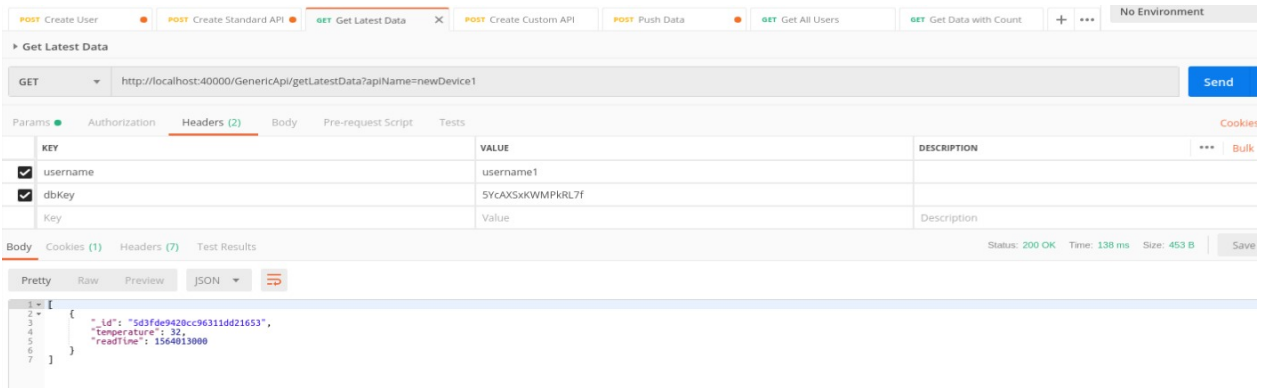
Akış üzerindeki 2a adımı seçildiğinde standart programlama arayüzü olarak adlandırılan yapı ile kullanıcılara sağlanan servisler listelenmiştir:

- **Cihazdan platforma veri gönderme:** HTTP protokolü üzerinden POST metodu ile cihaz üzerindeki verinin platforma gönderilmesini sağlayan bir servistir. Standart programlama arayüzünün çalıştırılması ile birlikte ilgili port numarası ve “/pushData” veri yolu ile kullanıcı için oluşturulmuş mongoDB veritabanındaki cihaz koleksiyonuna erişir ve veriyi burada kaydeder.



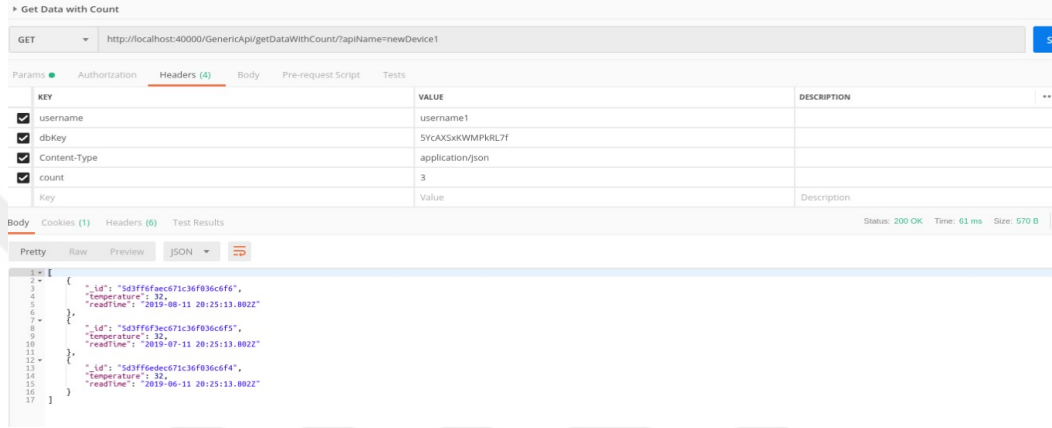
Şekil 3.11 Cihaz verisini platforma gönderme servisi

- **Platformdan kayıtlı cihaza ait son veriyi alma:** HTTP protokolü üzerinden GET metodu ile kullanıcıya ait veritabanında tutulan cihaza ait koleksiyondaki son kayıtlı veriyi getiren bir servistir. Standart programlama arayüzünün çalıştırılması ile birlikte ilgili port numarası ve “/getLatestData” veri yolu ile erişim sağlanmaktadır.



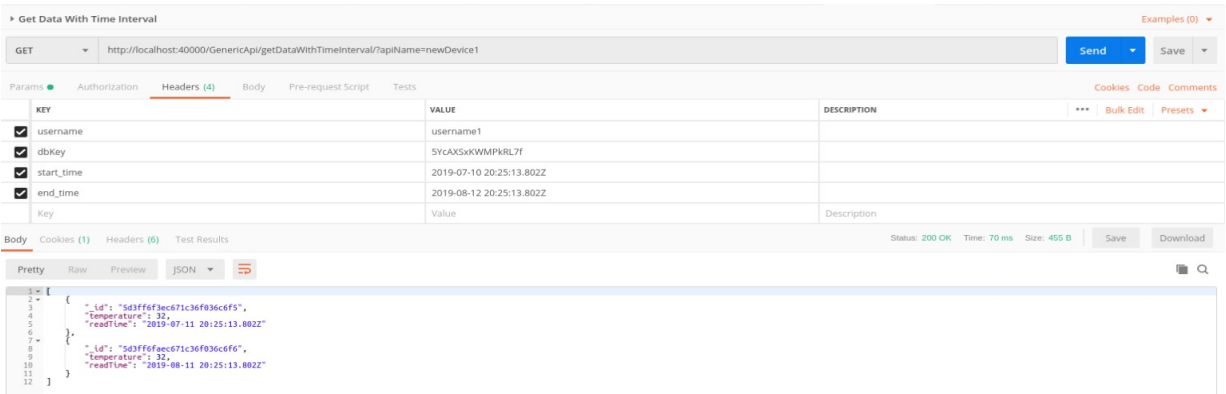
Şekil 3.12 Son cihaz verisini veritabanından alma

- **Platformdan kayıtlı cihaza ait belirli miktarda veriyi alma:** HTTP protokolü üzerinden GET metodu ile kullanıcıya ait veritabanında tutulan cihaza ait koleksiyondaki kayıtlı verilerden istemci tarafından parametrik olarak belirlenen miktardaki veriyi son eklenenden başlayarak getiren bir servistir. Standart programlama arayüzünün çalıştırılması ile birlikte ilgili port numarası ve “/getDataWithCount” veri yolu ile erişim sağlanmaktadır. Sayı belirten parametresi “count” ile belirtilmiştir. Ekran görüntüsü şekil 3.13’te verilmiştir.



Şekil 3.13 Belirli miktarda kaydı alma servisi

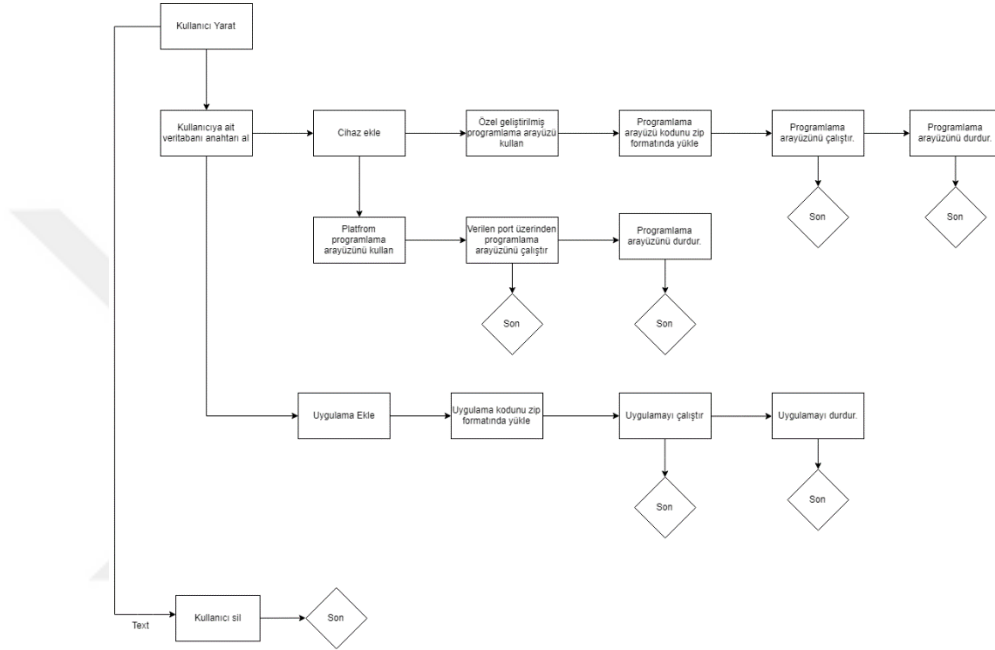
- **Platformdan kayıtlı cihaza ait verilerden belli bir tarih aralığında olanları alma:** HTTP protokolü üzerinden GET metodu ile kullanıcıya ait veritabanında tutulan cihaza ait koleksiyondaki kayıtlı verileri getiren bir servistir. Verileri getirirken istemci tarafından belirtilecek olan tarih parametrelerini kullanarak ona göre mongoDB’ye sorgu yapmaktadır. Standart programlama arayüzünün çalıştırılması ile birlikte ilgili port numarası ve “/getDataWithInterval” veri yolu ile erişim sağlanmaktadır. Ekran görüntüsü şekil 3.14’te verilmiştir.



Şekil 3.14 Belirli tarih aralığındaki verileri alma servisi

3.6 Platformun Test edilmesi ve Çıktıları

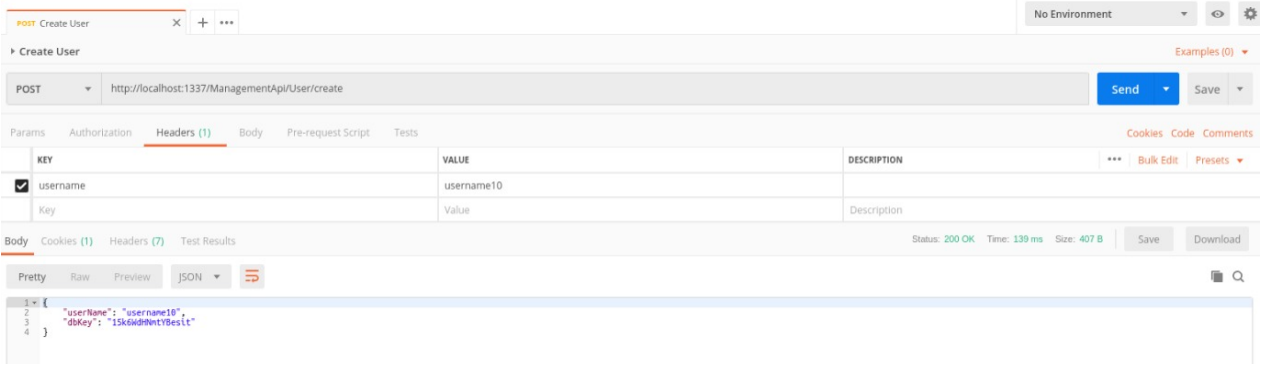
Platform gerçekleştirimine dair sağlanan servislerin postman istemcisi üzerinden testleri yapılmıştır. Bu testlerin yapılması için gerekli olan parametreler, veri yolları ve sisteme ait çıktıları gösteren servis çalıştırmaları bölümün devamında ekran görüntüleri ile paylaşılmıştır. Servislerin çalıştırılması sırasında aşağıdaki akış izlenmiştir:



Şekil 3.15 Platform kullanımı akış diyagramı

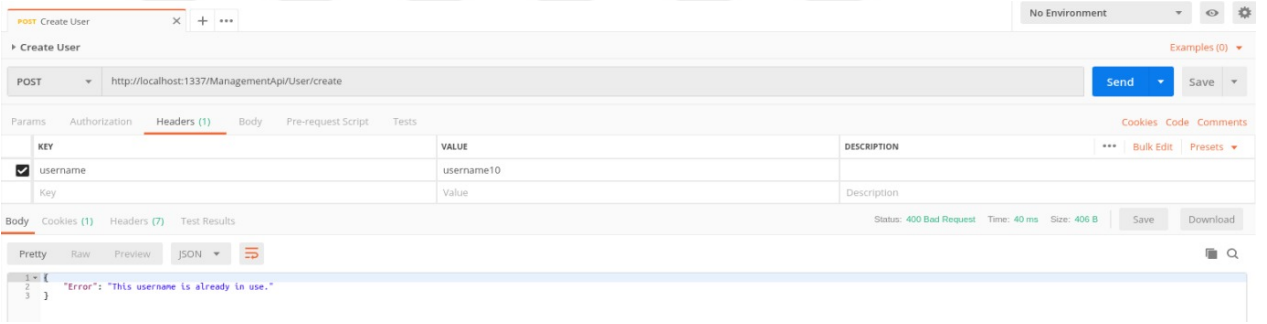
3.6.1 Kullanıcı Yaratma

Kullanıcı yaratmak için en öncül değişken özgün bir kullanıcı adıdır. Kullanıcıya ait kayıtlanmak istenen verilere göre yeni bilgiler bu servise parametre olarak eklenebilir, bu veriler bir portal aracılığı ile kullanıcıdan istenilebilir.



Şekil 3.16 Kullanıcı ekleme servisi testi

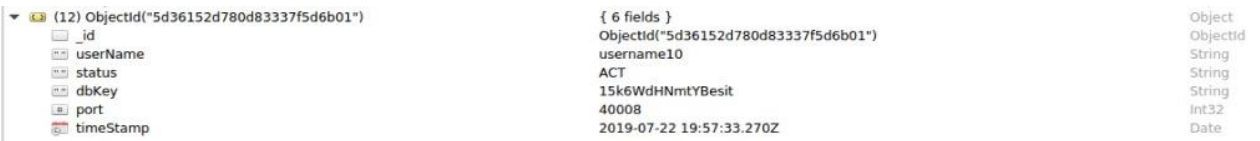
Kullanıcı adı özgün bir değişkendir. Eğer kullanıcı adı daha önceden kullanılmış bir veri ise sistem hata verir:



Şekil 3.17 Kullanıcı ekleme servisi hata mesajı

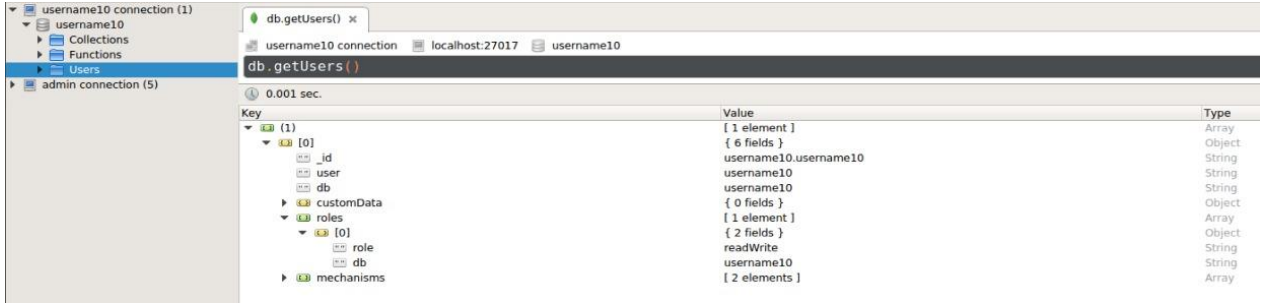
Kullanıcı ekleme işlemi başarı ile sonuçlandığında Kullanıcı adı ve oluşturulmuş olan DBKey erişim anahtarı bilgisi kullanıcıya geri dönmektedir.

Kullanıcı bilgisi Yönetici veritabanındaki Kullanıcılar Koleksiyonuna eklenir:



Şekil 3.18 Yönetici veritabanına kullanıcı eklenmesi

Kullanıcıya ait MongoDB üzerinde veritabanı da yaratılır:



Şekil 3.19 Başarılı senaryo sonrası yaratılan veritabanı

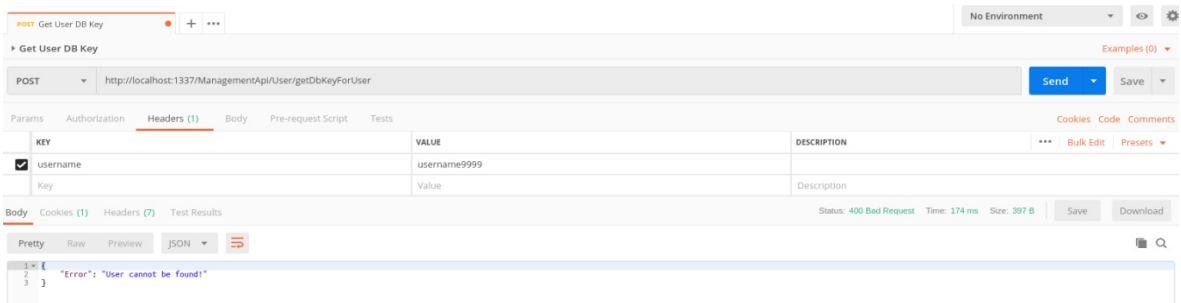
3.6.2 Kullanıcıya ait Veritabanı Anahtarı Alma

Kullanıcıya ait oluşturulmuş olan veritabanı için bir erişim anahtarı gerekmektedir. Bu erişim anahtarının getirildiği ayrı bir servis daha yaratılmıştır.



Şekil 3.20 Erişim anahtarı alma-Başarılı senaryo

İstenilen kullanıcıya ait veri bulunmuyorsa servis 400 hata mesajını döndürür:



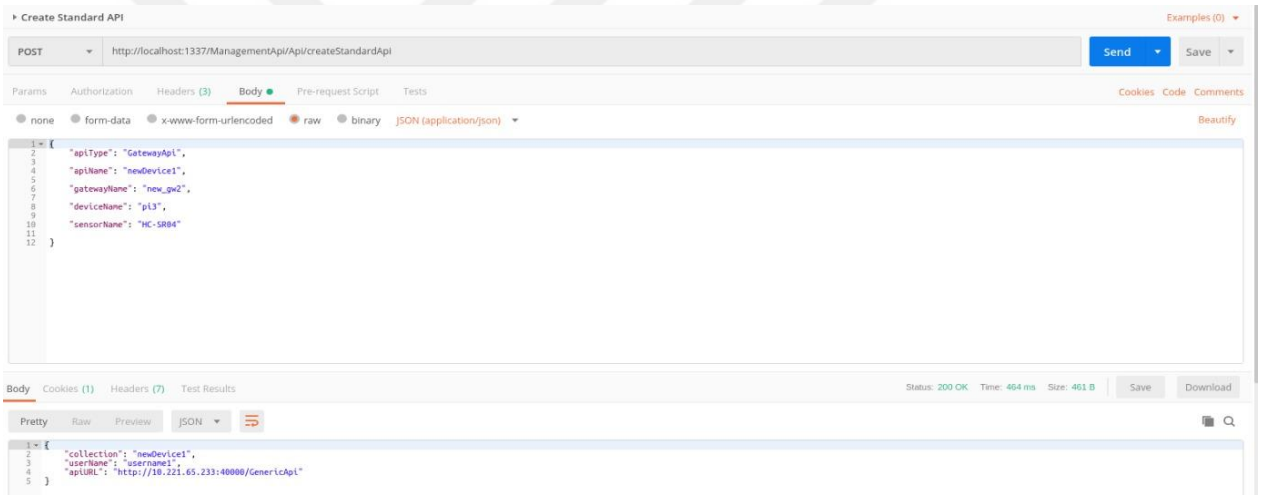
Şekil 3.21 Erişim anahtarı alma-Hatalı Senaryo

3.6.3 Standart Prosedür ile Cihaz Ekleme

Cihaz ekleme yöntemlerinden standart prosedür ile ilerleme opsiyonu seçildiğinde platform arka planda kullanıcıya standart servislerin sunulduğu bir programlama arayüzü sağlar.

Kullanıcılar cihazlarına ait hiyerarşik yapıyı yani isterlerse cihaz, sensör, ağ geçidi gibi tanımlamaları da yapabildikleri gibi isterlerse sadece özgün bir nesne adı belirleyerek de sisteme kayıt olabilirler.

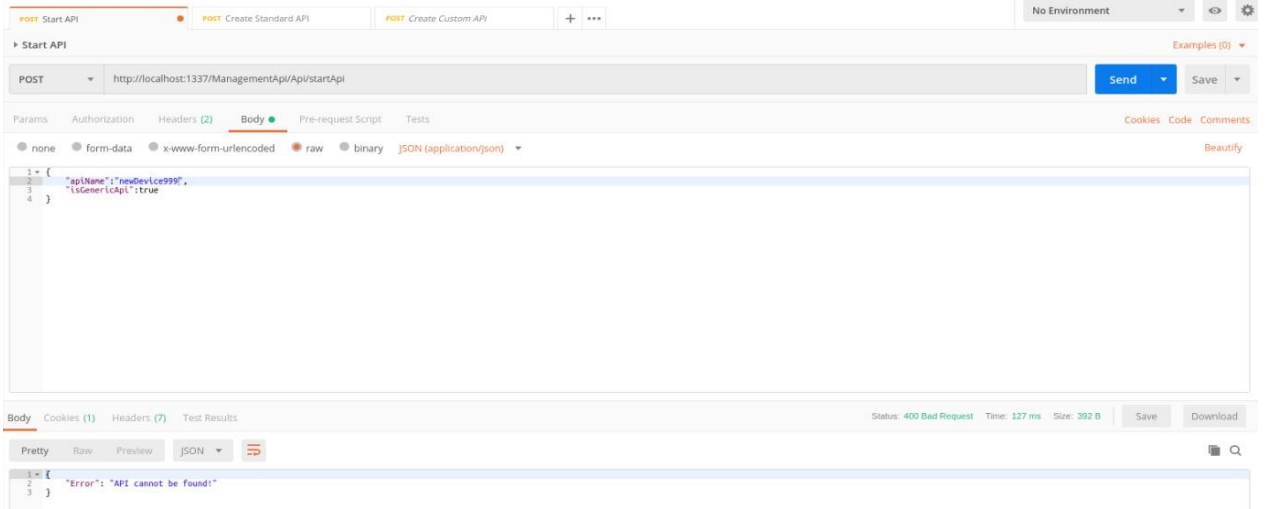
Başarılı senaryoda cihaz ekleme servisi çalıştırıldığında kullanıcıya cihazına erişim için kullanacağı bir adres bilgisi dönmektedir. Ayrıca daha sonra bu cihazın verilerine erişmek için kullanacağı gerekli değişkenlerden biri olan “ApiName” değişkeni de bu servisin cevap yapısında bulunmaktadır.



Şekil 3.22 Cihaz ekleme-Başarılı senaryo

3.6.4 Programlama Arayüzü Başlatma

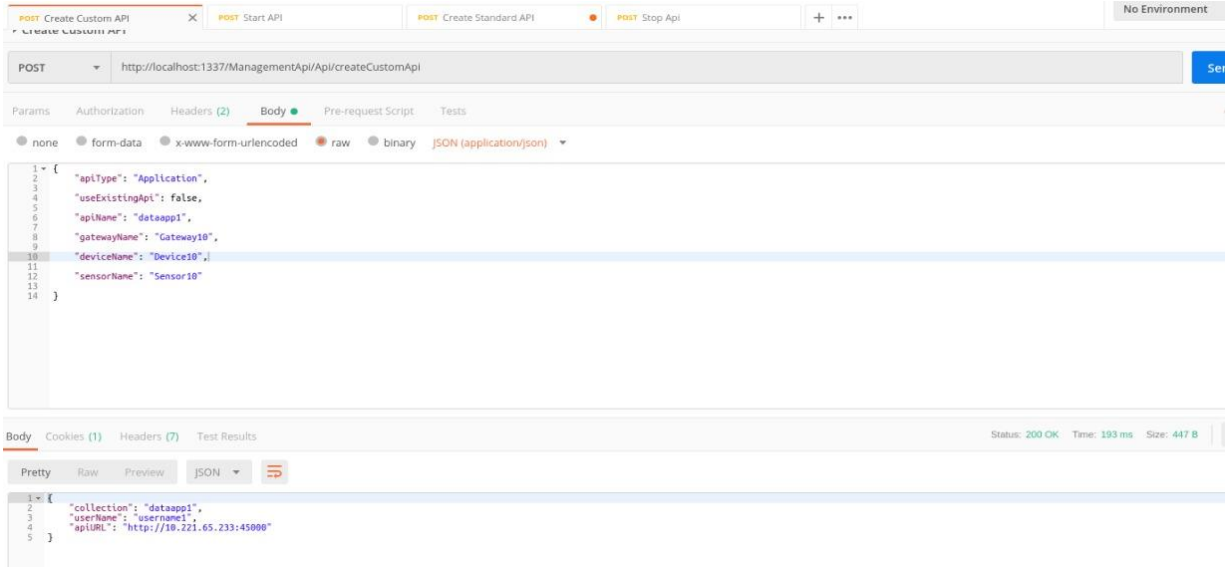
Platform üzerine yüklenmiş olan API’lerin yada platformun sunduğu standart API’nin aktifleştirilmesi için bu servis ile başlatılması gerekmektedir. Bu servis sonucunda eğer başarılı bir mesaj alınıyorsa programlama arayüzü cihazlar tarafından kullanılabilir olmaktadır.



Şekil 3.23 Programlama arayüzü başlatma

3.6.5 Özelleşmiş Programlama Arayüzü Yükleme

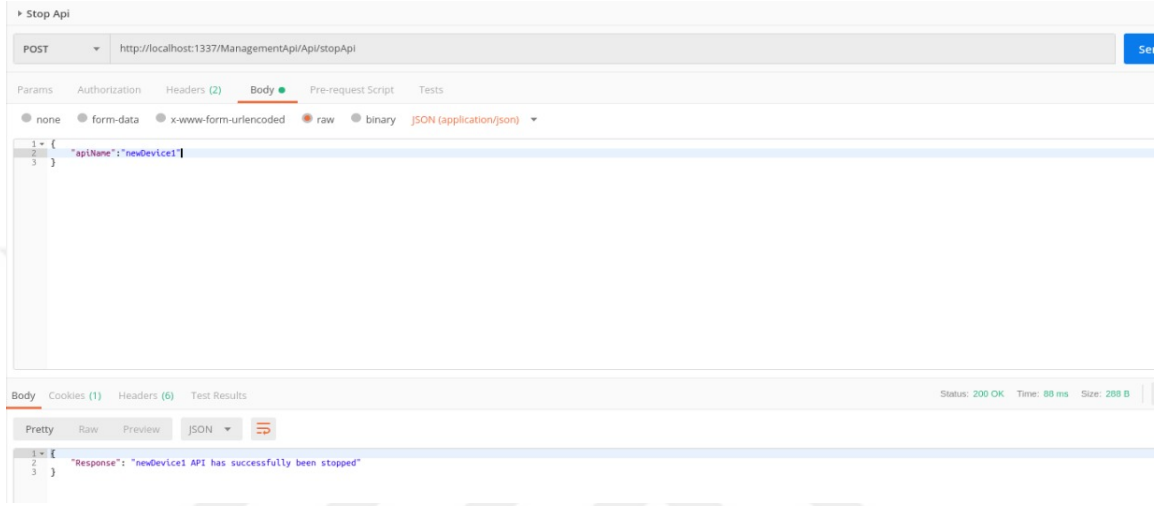
Platform üzerine özelleşmiş bir programlama arayüzü eklemek için yüklenen projenin Node.js projesi olma koşulu vardır. Bu proje dosyası platform sunucusuna arayüz aracılığı ile yüklenmiş olduğu kabul edilmiştir.



Şekil 3.24 Programlama arayüzü yükleme

3.6.6 Programlama Arayüzünü Durdurma

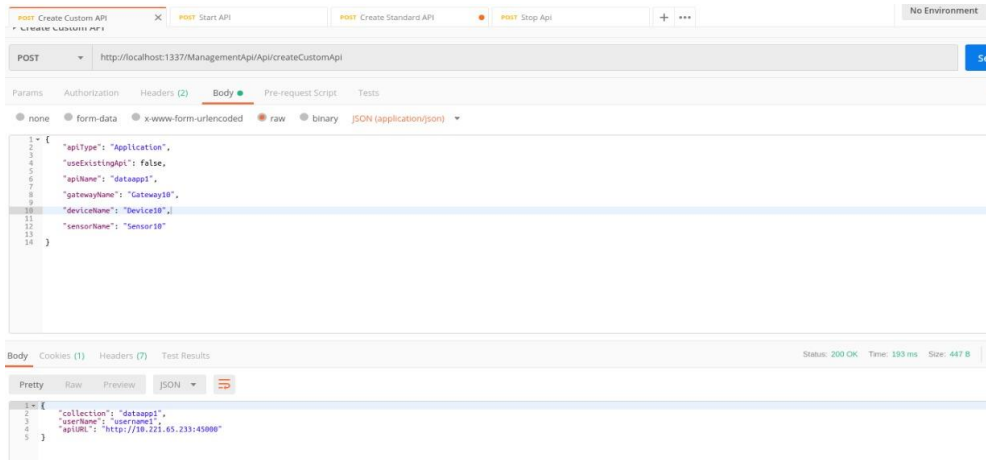
Platform tarafından sağlanmış olan uygulama programlama arayüzlerinin (API) durdurulması için şekil 3.25'te verilen /stopApi servisi çalıştırılmalıdır. Bu servis işlemi tamamladığında, API'nin durumuna göre durduruldu ya da zaten durdurulmuş bir cevap döndürmektedir.



Şekil 3.25 Programlama arayüzü durdurma

3.6.7 Uygulama Yükleme

Kullanıcı platforma uygulama yüklemek istediğinde uygulamasına ait proje dosyasının bir node.js projesi olması gerekmektedir. Bu proje dosyası platform sunucusuna arayüz aracılığı ile yüklenmiş olduğu kabul edilmiştir.

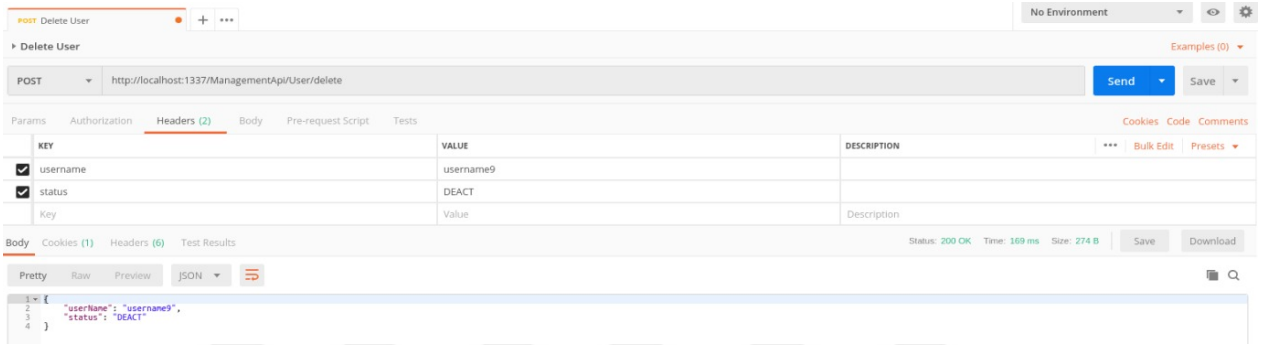


Şekil 3.26 Uygulama yükleme servisi

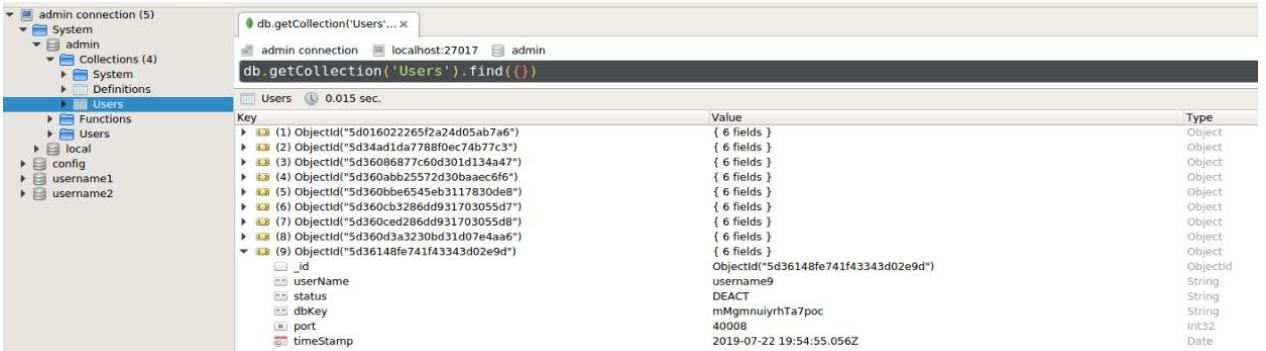
Uygulamayı çalıştırma ve durdurma servisleri uygulama programlama arayüzü servisleri ile ortaktır.

3.6.8 Kullanıcı Sil/Hesabı Dondurma

Platformda tanımlı kullanıcılar, kullanıcıların cihazları ve bunlara ait veriler hesaba erişim dondurulduğunda sistemden silinmezler, veriler sonraki analizlerde kullanılabilir ayrıca kullanıcı tekrardan sisteme erişim isteyebilir. Bu nedenle kullanıcıya ait bilgilerin tutulduğu koleksiyonda kullanıcı durumu deaktif durumuna atanmaktadır.



Şekil 3.27 Kullanıcı hesabının silinmesi/dondurulması



Şekil 3.28 Kullanıcının veritabanındaki durumu

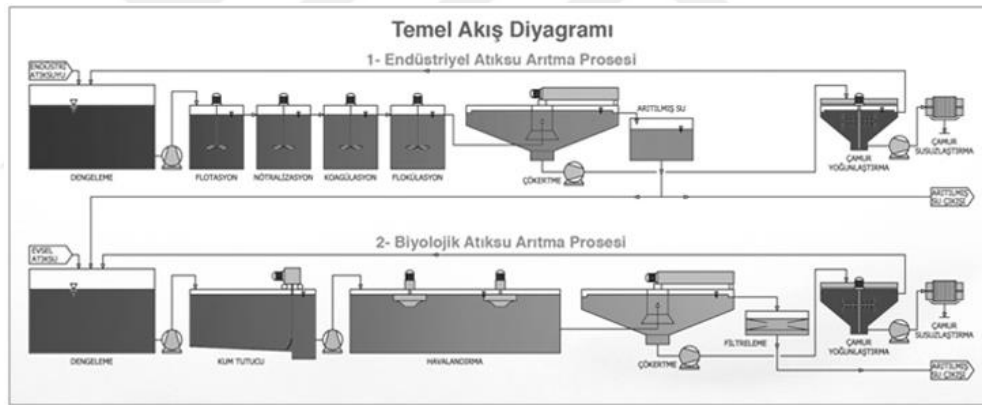
4 DURUM ÇALIŞMASI

4.1 Atık Su Arıtma Tesisi Teknik Gezisi

İzmir’de bulunan Çiğli Atık Su Arıtma Tesisi , İzmir Körfezi'nin atıksu kirliliğinden kurtarılması amacı ile Büyük Kanal Projesi kapsamında inşa edilmiştir. Arıtma süreci, biyolojik olarak fosfor ve azot gideren ve daha kaliteli çıkış suyu elde edilebilen "ileri biyolojik arıtma" yöntemine göre tasarlanmıştır. (İzsu, 2019)

Tesis yönetimi izni ile Ege Üniversitesi Çevre Bilimleri Araştırma Merkezi yüksek lisans öğrencisi Nazım Şahin eşliğinde teknik gezi yapılmıştır.

Arıtma tesisinde mekanik arıtma yapan ince ızgaralar, kum tutucular, çökeltme tankları; biyolojik arıtma yapan bio-fosfor havuzları, havalandırma havuzları ve çamur toplama tankları bulunmaktadır. Çamur çürütme ve kurutma ünitesi ile de son ürün oluşturulmaktadır. (İzsu, 2019)



Şekil 4.1 Atıksu arıtma süreci (Hidro dizayn, 2019)

Yapılan gezide tesiste çalışan elektrik teknisyeni mevcut sistemdeki ölçümler ve görüntülemelerde sistemler ile ilgili bilgi vermiştir. Birimlerdeki ölçümler PLC (Programmable Logic Controller) sistemleri ile yapılmaktadır. Bu bilgiler ethernet kabloları ile kumanda merkezindeki sunuculara gelmektedir. Günlük elektrik tüketimleri, tesislerin giriş - çıkış debileri, tesis giriş hazne seviyeleri, havalandırma havuzlarındaki oksijen değerleri ve dekantör çamur debi değerleri ile süreç için kritik olan motorların durum bilgileri SCADA sistemi ile izlenmektedir. SCADA, “Merkezi denetleme kontrol ve veri toplama” sistemlerine verilen addır. Tesiste kullanılan sistemler tesisin ihtiyaçlarını karşılamaktadır. Süreçlerle ilgili teknolojik sistemlerden kaynaklı bir problemten bahsedilmemiştir. Yalnızca sunucu maliyetleri, kontrol ve izlemeye kullanılan sistemlerin maliyetleri ve yeni teknolojilere adaptasyon sürecinde yaşanan sıkıntılardan bahsedilmiştir.

Sunucu maliyetlerinin azaltılması için tüm dünyada bulut teknolojilerinin kullanılması artmıştır. Yapılan teknik gezide alınan geri bildirimlerden yola çıkarak daha düşük maliyetli bir

çözüm nasıl sağlanabilir fikri ile nesnelerin interneti platform prototipinin kullanılması düşünülmüştür.

4.2 Platform Kullanmanın Avantajları

Platform bölüm 3.3'te bahsedilen kurulum opsiyonlarından opsiyon 1'e bağlı kalınarak kurulmuştur. Böyle bir tesiste platform kullanımı sayesinde veriler merkezi bir yerde tutulur hale gelmiştir. Platform kullanımı ile farklı farklı senaryolar bir çatı altında toplanmıştır. Platform sayesinde izleme arayüzlerine ve kontrol ünitelerine uzaktan da erişilebilir. Sisteme yeni bir eklenti olduğunda yada kullanılan sistemler üzerinde bir değişiklik olduğunda yalnızca o sistemi ilgilendiren servisin değiştirilmesi yeterli olacaktır. Böylece uygulamaların ve süreçlerin bakımı da kolaylaşacaktır. Uygulamalar ve uygulama arayüzleri de platform üzerinden kullanıcılara servis edilebilecektir. Bulut kurulumu sayesinde yerel sunucu ve bakım maliyetleri azalmış olacaktır.

4.3 Platformun Kullanılması

Platform sunucu maliyetlerini minimize etmek için bulut üzerine kurulumu yapılmıştır. Amazon ec2 bulut sistemi kullanılarak Linux tabanlı bir sunucu kiralanmıştır.

Çekirdek platform kurulumu için gerekli adımlar aşağıda listelenmiştir:

- Npm kurulumu
- Node.js kurulumu
- Sails.js kurulumu
- MongoDB kurulumu
 - MongoDB'yi Auth modunda çalıştırma
 - MongoDB'ye admin kullanıcı belirleme
- Platform yazılımının sunucuya aktarımı ve çalıştırımı

Senaryo -1: Kaba İzgara İzleme Sistemi

Atıksu arıtma tesisinin ilk aşamasında atıksu içerisindeki katı atıkların ızgara sistemi ile mekanik olarak temizlenmesi için kaba ızgara yapısı kullanılmaktadır. Bu ızgaraların katı atıklarca dolması durumunda süzme işleminin devamı için bir tırmık sistemi uyarılır, ızgara temizlenir ve atıksu akışı devam eder. Bu sistemin prototipinin gerçekleştirilinde ızgara çevresindeki debi ölçer sensörlerden veri geldiği düşünülmüştür.

Prototip için aşağıdaki sensörler kullanılmıştır:

- Debi ölçme sensörü: Debi ölçüm verileri simüle edilmiştir.
- Servo motor: Izgara tırmığının kontrolü için aktüator görevindedir.
- Raspberry pi2: Motor cihazını internet üzerinden erişilebilir olmasını sağlamaktadır.

Bu kullanım senaryosunun gerçekleştirimi için aşağıdaki servisler sırası ile çalıştırılmıştır:

- İzsu için kullanıcı yaratma servisinin çağırılması
 - Kullanıcı adı ve kullanıcıya ait veritabanı anahtarı bilgisinin alınması
- Debi ölçme sensörü verileri için platform üzerinde cihaz ekleme servisinin çağırılması
 - Standart programlama arayüzü servisinin çağırılması
 - Oluşturulan API'nin çalıştırılma servisinin çağırılması
- Motor modülünü yönetmek için platform üzerinde cihaz ekleme servisinin çağırılması
 - Özelleştirilmiş programlama arayüzü servisinin çağırılması
 - Oluşturulan API'nin çalıştırılma servisinin çağırılması
- İzleme önyüzünün platform üzerine yüklenmesi
 - Özelleştirilmiş programlama arayüzü servisinin “APP” parametresi ile çağırılması
 - Oluşturulan uygulamanın çalıştırılma servisinin çağırılması

Debi ölçer sensörler için platformun genel programlama arayüzü kullanılarak servisler yaratılmıştır. Tırmık kontrolü için ise özelleştirilmiş programlama arayüzü kullanılarak tırmığa çalışma komutu gönderen bir servis oluşturulmuştur.

Izgaranın sıkışıklık durumu platform üzerinde konuşlandırılmış bir önyüz üzerinden debi ölçerlerden gelen verilere göre izlenebilmektedir. Operatör gerekli gördüğünde arayüz üzerinden tırmığı kontrol ederek sistemin alarm verip otomatik olarak aksiyon almasından önce temizleme işlemini gerçekleştirebilmektedir.

Senaryo - 2: Havuzlar Arası İletim

Havalandırma havuzlarında bulunan atıksu, arıtma sürecinde ön çökeltme havuzlarına aktarılacaktır. Havalandırma havuzlarının su seviyesi ve giriş debisine göre 3. veya 4. ön çökeltme havuzu da işletilmeye başlanır.

Bu yapının prototipi için aşağıdaki sensörler ve modüller kullanılmıştır:

- Uzaklık ölçüm sensörü: Havuz su seviyesini ölçmek için kullanılmıştır.
- Servo motor: Ön çökeltme havuzlarına dağıtım için kullanılmıştır.
- Raspberry pi2: Bu modül üzerine bağlanan uzaklık ölçüm sensörü ve motor cihazının internet üzerinden erişilebilir olmasını sağlamaktadır.

Yapılan prototipte Raspberry pi üzerinde çalışan bir python yazılımı bulunmaktadır. Bu yazılım periyodik aralıklarla uzaklık ölçüm sensörü verilerini alıp internet üzerinden platform tarafından sağlanmış programlama arayüzü adresine göndermektedir.

Bu kullanım senaryosunun gerçekleştirimi için aşağıdaki servisler sırası ile çalıştırılmıştır:

- İzsu için yaratılmış olan kullanıcı hesabı kullanılmaktadır.
- Uzaklık sensörü verileri için platform üzerinde cihaz ekleme servisinin çağırılması
 - Standart programlama arayüzü servisinin çağırılması
 - Oluşturulan API'nin çalıştırılma servisinin çağırılması
- Motor modülünü yönetmek için platform üzerinde cihaz ekleme servisinin çağırılması
 - Özelleştirilmiş programlama arayüzü servisinin çağırılması
 - Oluşturulan API'nin çalıştırılma servisinin çağırılması

- İzleme önyüzünün platform üzerine yüklenmesi
 - Özelleştirilmiş programlama arayüzü servisinin “APP” parametresi ile çağırılması
 - Oluşturulan uygulamanın çalıştırılma servisinin çağırılması

Senaryo – 3: Reaktör İçi pH Kontrolü

Atıksu arıtma sürecinde ortaya çıkan aktif çamur yoğunlaştırılıp biyogaz reaktörleri içerisinde bekletilerek biyogaz elde edilmektedir. Reaksiyonda sıcaklık, ph, gaz yüzdeleri ve basınçları izlenmelidir. Aktif çamurun niteliğine bağlı olarak bu parametrelerin değerleri değişebilmektedir. Bu reaksiyonda görev alan bakteriler ph değişimlerinden kolaylıkla etkilenebilmekte bu da süreci olumsuz etkilemektedir. Asit baz dengesinin optimal koşullarda olabilmesi için reaktöre aktüatörler ile asidik veya bazik derişim eklenebilmektedir.

Bu yapının prototipi için aşağıdaki sensörler ve modüller kullanılmıştır:

- PH ölçüm sensörü: pH değerini ölçmek için kullanılmıştır.
- Sıcaklık ve nem ölçüm sensörü: Reaktör içindeki sıcaklık ve nem değerlerini ölçmek için kullanılmıştır.
- Servo motor: Asidik veya bazik çözelti ekleme aktüatörü olarak kullanılmıştır.
- Raspberry pi2: Bu modül üzerine bağlanan pH ölçüm sensörü ve motor cihazının internet üzerinden erişilebilir olmasını sağlamaktadır.

Bu kullanım senaryosunun gerçekleştirimi için aşağıdaki servisler sırası ile çalıştırılmıştır:

- İzsu için yaratılmış olan kullanıcı hesabı kullanılmaktadır.
- pH sensörü verileri için platform üzerinde cihaz ekleme servisinin çağırılması
 - Standart programlama arayüzü servisinin çağırılması
 - Oluşturulan API'nin çalıştırılma servisinin çağırılması
- Sıcaklık ve nem verileri için platform üzerinde cihaz ekleme servisinin çağırılması
 - Standart programlama arayüzü servisinin çağırılması
 - Oluşturulan API'nin çalıştırılma servisinin çağırılması
- Asidik/Bazik çözelti ekleme motorunu yönetmek için platform üzerinde cihaz ekleme servisinin çağırılması
 - Özelleştirilmiş programlama arayüzü servisinin çağırılması
 - Oluşturulan API'nin çalıştırılma servisinin çağırılması
- İzleme ve kontrol önyüzünün platform üzerine yüklenmesi
 - Özelleştirilmiş programlama arayüzü servisinin “APP” parametresi ile çağırılması
 - Oluşturulan uygulamanın çalıştırılma servisinin çağırılması

5 SONUÇ VE TARTIŞMA

Nesnelerin interneti platform çözümleri ve literatürdeki mimari modeller incelenmiştir, yapılan incelemeler ve kullanım esnasındaki deneyimler sonucunda yalın bir model oluşturulmuştur. Bu model ile bir platformun nesnelerle iletişim için sahip olması gereken fonksiyonallikler düşünülmüş ve bu yapının farklı ölçekler ve talepler için nasıl kullanılabileceği önerilmiştir.

Platformun geliştirimi yapılmış ve örnek senaryolar ile denenmiştir. Platformu kullanmanın avantajları bu bölümde açıklanacaktır.

- Platform aracılığı ile ekstra geliştirme yapmadan cihaz verileri platformda depolanmaya başlar ve bu veriler uygulamalarda sağlanan servisler ile kullanılabilir. Standart programlama arayüzünün sağladığı fonksiyonlar ile kullanıcılar kolaylıkla cihazlarını platforma bağlayıp verilerini uygulamalarda görebilirler. Diğer platformlarda olduğu gibi veri modeli belirleme, servis yazma ve çalıştırma gibi aşamalar standart programlama arayüzü sayesinde atlanmış olur.
- Platform kullanılarak aynı cihazların bağlı olduğu farklı uygulamalar ortak servisler aracılığı ile oluşturulabilir. Cihazlar ve sensörler kullanılarak istenilen uygulamalar platform kullanılmadan da yapılabilirdi. Uygulamalar platform kullanılmadan geliştirilmek istenilseydi her bir uygulama için ayrı cihaz bağlantı protokolü destekleyen bir yazılım modülü, cihazın verileri için uygulamaya özel bir veritabanı kurulumu, cihazın verilerinin gösterileceği bir arayüz gerekmekte idi. Platform kullanımı ile cihazlara ait veriler platform üzerinde tutulur ve bu veriler programlama arayüzleri aracılığı ile merkezi bir veri katmanından farklı farklı uygulamalara servis edilebilir. Platform kullanımı ile cihaz bağlantısı platform tarafından sağlanan standart programlama arayüzü sayesinde yapılır. Geliştiriciler veritabanı teknolojisi seçimi, kurulumu ile ilgilenmezler. Platform tarafından sağlanan standart fonksiyonallite ile de kodlama yapılmadan servisler aracılığı ile istenilen yere veriler aktarılabilir. Eğer farklı senaryolar kullanılacaksa da yalnızca veri manipülasyonunu yapan bir servis geliştirimi ile uygulamanın beklediği şekilde veriler dış dünyaya açılabilir. Bu servis yapısı da platform üzerinde tutulur. Tüm servisler farklı farklı uygulamalarda tekrar tekrar kullanılabilir. Bu sayede aynı cihazlar farklı senaryolar için farklı uygulamalarda kullanılmış olur. Platform sayesinde yazılım tekrar kullanılabilir küçük bloklara ayrılmış olur. Uygulamaların da bakımı kolaylaşmış olur.
- Platform veri modeli bağımsızdır. Geliştiriciler cihaz verilerini manipüle edip, platformun kabul edeceği veri formatlarını uygulamak zorunda değildirler. Veriler JSON formatında istenildiği veri modeli kullanılarak platform üzerinde kayıt edilebilir. XML formatı için dönüştürücü içeren bir servis özelleşmiş programlama arayüzü olarak yüklenerek platformun desteklemesi sağlanabilir.

- Platform bulut servislerinden bağımsızdır.
Platform bulut sistemi sağlayıcıların servislerinden bağımsız olarak geliştirilmiştir. Bu nedenle hem bulut üzerinde hem de yerel sunucularda konuşlandırılabilir.
- Platform hem ürün olarak hem de servis sağlayıcılar için servis (as a service - aaS) olarak konumlandırılabilir.
Platform çekirdek yapısı gereği bölüm 3.3'te anlatıldığı gibi hem tekil olarak ihtiyaçlara cevap verirken, istenildiği takdirde opsiyon 1'deki hali ile ölçeklendirilebilir. Servis sağlayıcı olarak kullanılmak istenildiğinde ise opsiyon 2'de gösterildiği gibi farklı isteklerde bulunan kullanıcılar için ayrı çekirdek kurulumlar servis sağlayıcının merkezi sistemlerine bağlanabilir.
- Platform kullanıcılarına hızlı bir şekilde cihazlarını bağlamalarına olanak verir.
Platform kullanıcıları alan bilgisi gerekmeden hızlı bir şekilde cihazlarını platforma tanımlayıp verilen standart fonksiyonallite ile platformu kullanmaya başlayabilirler. Platforma cihaz eklenirken sorulan programlama arayüzü tipi isteğine göre eğer standart programlama arayüzü servisi kullanılacaksa herhangi bir kodlama yapmadan listelenen servislere sahip olur:
 - Cihaz verisini platforma gönderme
 - Son kayıtlanmış cihaz verisini platformdan uygulamaya alma
 - Kayıtlı cihaz verisinin geçmişe dönük bir kısmını alma
 - Son n sayıdaki kaydı alma

Bu servisler ile hem cihaz verisi platforma kaydedilir hem de bu veriler farklı uygulamalar içerisinde grafiklerde ve analizlerde kullanılabilir.
- Platform kullanıcılarına, servislerini kontrol etme olanağı tanır.
Platformun yönetim servisleri sayesinde cihaz ve uygulama bağlantıları için geliştiricilerce tanımı yapılmış standart uygulama arayüzleri ve özelleşmiş programlama arayüzleri için başlatma ve durdurma operasyonları sunulmuştur. Bu servisler ile uygulama geliştiriciler uygulama son kullanıcılarının veriyi görüntülemesini kontrol altında tutmaktadır. Örneğin ödemesini yapmamış bir kullanıcı için bu servislerin durdurulması mümkündür. Ödeme yapıldıktan sonra cihaz verilerini uygulamaya gönderen servis yönetim servisleri aracılığı ile yeniden çalışır hale getirilebilir.
- Platform, platform kullanıcılarını kontrol etmeye olanak tanır.
Platform yönetim servisleri ile platformu kullanan kullanıcıların hesapları aktive ve deaktive edilebilir. Bu sayede sistemden ayrılmış bir kullanıcının hesabı dondurulabilir.
- Platform üzerinde uygulamalar konuşlandırılabilir.
Platform kullanıcıları node.js ile geliştirilmiş uygulamalarını platform üzerine yükleyebilir. Bu sayede kullanıcılara ait hem cihazlar hem programlama arayüzleri hem de uygulamaları merkezi bir platform üzerinden yönetilebilir. Uygulamaların dış dünyaya açılımı platform

üzerinden kontrol edilebilir. Kullanıcılar uygulamaları için ayrı bir sunucu ayarlamak ile uğraşmazlar.

- Platform esnektir.
Platforma geliştiriciler tarafından farklı görevleri yerine getirmek üzere geliştirmiş node.js tabanlı programlama arayüzleri yüklenebilir. Bunun için özelleşmiş programlama arayüzü yükleme, başlatma servisleri kullanılır. Platform bu sayede yeni bir bağlantı protokolünü, yeni bir modül entegrasyonunu destekleyebilir hale gelir.
- Platform kurulumu basittir.
Platform kurulumu için belli bir komut sırasının kullanılması yeterlidir. Kullanıcılar ek modüllerin neler olması gerektiğini ve bunların kurulumu için neler yapılması gerektiğini düşünmekle zaman kaybetmezler. Çekirdek kurulumu yapıp cihazlarını bağlayabilir ve bu verileri uygulamalarında gösterebilirler, istenilirse entegrasyon düşünülen modüller sonradan özelleşmiş servisler ile entegre edilebilir.

Market araştırması bölümünde incelenen nesnelerin interneti platformu ürünlerindeki modeller ile önerilen mimari model karşılaştırıldığında aşağıdaki sonuçlar elde edilmektedir:

- Önerilen model incelenen platform ürünlerinin modellerinin de kapsamındadır.
- Önerilen model incelenen ürünlerin modellerinin bir alt kümesi gibi nitelendirilebilir.
- Önerilen modele istenildiği takdirde gerekli modüllere servisler aracılığı ile entegre edilebilir.
- Önerilen modelin çekirdek yapısı çoklu kurulum opsiyonu ile farklı kullanım senaryolarında özerk bir alan sağlar.
- Önerilen modelin bulut sistemler ile bir bağımlılığı yoktur bu nedenle kapsam istenildiği gibi tutularak kurulum da istenilen şekilde yapılabilir.
- Önerilen model hem büyük hem küçük hem de kendi içerisinde kolarize olmuş küçük sistemler olarak kullanılabilmeyle önermektedir.

Çalışmada incelenen platform çözümleri ile tez çalışması platformu karşılaştırmalı tablosu tablo 5.1'de verilmiştir:

Tablo 5.1 Platformların karşılaştırılması

Kriterler	Tez Çalışması	Thingworx	IBM Watsons	Azure IoT	Amazon AWS
Entegrasyon Katmanı	+	+	+	+	+
Veri Katmanı	+	+	+	+	+
Yönetim Katmanı	+	-	+	+	+
Çalıştırma Katmanı	+	+	+	+	+
Lokal Kurulum	+	+	-	-	-
Bulut Kurulum	+	+	+	+	+
Esneklik	+	±	+	+	+
Kolay Kurulum	+	+	-	-	-

Tablo 5.1’de incelenen kriterlere bakılacak olursa, öncelikle önerilen platformun fonksiyonel katmanlarının market çözümlerinde de olup olmadığına bakılmıştır. Yönetim, entegrasyon, çalıştırma ve veri katmanlarının neredeyse hepsinin ortak olduğu görülmüştür, yalnızca yönetim katmanı Thingworx çözümünde bulunmamaktadır. Thingworx üzerinde bağlanan cihazların, kullanıcıların yönetilebileceği bir yapı bulunmamaktadır. Tez çalışmasının yönetim servisleri ile cihaz ve kullanıcı hayat döngüleri yönetimi mümkündür. Bunun yanı sıra incelenen diğer kriterlere bakılacak olursa lokal sunuculara kurulum açısından tez platform çözümü ve Thingworx çözümü lokal kurulumu desteklemektedir.

Esneklik kriteri incelenirken, bulut servisleri ile çözüm veren market çözümlerinin modüler ve farklı gereksinimler için farklı çözümler ile geliştirilebilir oldukları görülmektedir. Tez çalışması da özelleşmiş programlama arayüzü yükleme özelliği ile entegrasyonları mümkün kılar ve platformun esnek olmasını sağlar. Thingworx çözümü de diğer sistemler ile bağlayıcılar aracılığı ile entegre olabilir ancak iletişim protokolü kısıtlamaları mevcuttur.

Kurulum için bulut çözümleri ihtiyaca göre farklı modüller önermektedir, bu modüllerin ne zaman hangi durumlar için gerekli olabileceğinin araştırılması ve gereksinimin netleştirilmesi kurulum aşamasını uzatmaktadır. Thingworx çekirdek yapısı ile hızlı bir şekilde kurulabilmektedir. Tez çalışması da bir komut dizisi ile kurulabilen yapıdadır.

6 ÖNERİLER

Gelecek dönemde platform üzerinden geliştirilmesi planlanan durumlar bu bölümde özetlenmiştir:

- Platformun kullanılabilirliğinin artırılması için öncelikle bir portal/ön yüz yazılımı geliştirilmesi gerekmektedir.
- Platform cihaz bağlantısı katmanında HTTP protokolü üzerinden geliştirilmiş servisler ile denenmiştir. Gelecek dönemde farklı bağlantı protokollerini desteklemek için platformun özelleşmiş programlama arayüzlerini çalıştırabilme özelliği sayesinde CoAP, MQTT ve webSocket node.js kütüphaneleri kullanılarak cihazlar ile iletişebilir hale getirilebilir.
- Platformun güvenlik gereksinimleri istenilen senaryolara göre uygulanabilir. Platforma erişim güvenlik duvarı ile sağlanabilir.
- Kullanıcıların erişim hakları bir erişim kontrolcüsü ile kontrol edilebilir.
- Veritabanı erişim anahtarları gibi koruma şifreleme algoritmaları ile korunabilir.
- Depolanan verilerin bir büyük veri çözümü ile entegre olabilmesi için bir büyük veri teknolojisinden yararlanılabilir. Sistemde kullanılan mongoDB teknolojisi ile kolaylıkla entegre olabilen Apache Spark ürünü kullanılabilir.

KAYNAKLAR DİZİNİ

Al-Qaseemi, S. A., Almulhim, M. F., Almulhim, H. A. and Chaudhry, S. R., 2016, IoT architecture challenges and issues: Lack of standardization, future technologies conference, *IEEE Conference Publications*, 731-738 pp.

Amazon AWS Platformu, 2019, (<https://aws.amazon.com/tr/iot-core/>) (Erişim: 9 Mayıs 2019)

Atzori, L., Iera, A., and Morabito, G., 2010, The internet of things: A survey, *Computer Networks*, 15 pp.

Auto-Id Labs, 2019, <http://www.autoidlabs.org/> (Erişim: 5 Mayıs 2019)

Bilgi Teknolojileri ve İletişim Kurumu, 2013, “Makinalar Arası iletişim Raporu”, <https://www.btk.gov.tr/uploads/pages/slug/makineler-arasi-iletisim-m2m.pdf> (Erişim: 9 Mart 2019)

Bilgi Teknolojileri ve İletişim Kurumu, 2014 “Elektronik Haberleşme Sektöründe Teknolojik Gelişmeler ve Eğilimler”, <https://www.btk.gov.tr/uploads/pages/slug/elektronik-haberlesme-sektorunde-teknolojik-gelismeler-ve-egilimler.pdf> (Erişim: 12 Mart 2019)

BTK, 2019, Bilgi ve iletişim teknolojileri terimleri kılavuzu, <https://www.btk.gov.tr/uploads/pages/slug/kilavuz.pdf> (Erişim: 25 Temmuz 2019)

Datta, S.K. and Bonnet, C., 2015, Connect and control things: Integrating lightweight IoT framework into a mobile application, *IEEE 9th International Conference on Next Generation Mobile Applications*, 66-71 pp.

Docker, 2019, Docker yapısının temelleri, <https://docs.docker.com/get-started/> (Erişim: 25 Temmuz 2019)

Emil, B., 2014, “Why NoSQL databases are needed for the Internet of Things”, *Machina Research*.

European Commision, 2008, “Internet of things in 2020 roadmap for the future”, https://docbox.etsi.org/erm/Open/CERP%2020080609-10/Internet-of-Things_in_2020_EC-EPoSS_Workshop_Report_2008_v1-1.pdf, (Erişim: Eylül 2018)

Flip Platformu, <http://myflip.io/> (Erişim: 13 Mayıs 2019)

Guth, J., Breitenbucher, U., Falkenthal, M., Leymann, F. and Reinfurt, L., 2016, Comparison of IoT platform architectures: A field study based on a reference architecture, *IEEE Cloudification of the Internet of Things (CIoT)*, 978-1-5090-4960-8:1-6 pp.

Guth J. A., 2016, Architectural design of an abstraction layer for the integration of heterogeneous cyber-physical systems, MSc Thesis, Institute of Architecture of Application Systems University of Stuttgart.

KAYNAKLAR DİZİNİ (devam)

Google Trends, 2019, “Dünya genelinde m2m ve nesnelerin interneti trendleri”, <https://trends.google.com.tr/trends/explore?date=all&q=m2m,%2Fm%2F02vnd10>, (Erişim:7 Mart 2019)

Heo, Y.J., Oh, S.M., Chin, W.S. and Jang, J.W., 2015. A lightweight platform implementation for internet of things, *IEEE 3rd International Conference on Future Internet of Things and Cloud*, 526-531 pp.

Hidro dizayn, 2019, Atıksu arıtma sistemleri çözümleri, http://www.hidro dizayn.com.tr/atik_su.asp (Erişim: 2 Temmuz 2019)

IBM Watsons, 2019, Watsons Ürün Mimari Modeli, <https://www.ibm.com/support/knowledgecenter/en/SSQP8H/iot/overview/architecture.html> (Erişim: 22 Nisan 2019)

ITU, 2005, “ITU Internet Reports 2005: The Internet of Things”, https://www.itu.int/dms_pub/itu-s/opb/pol/S-POL-IR.IT-2005-SUM-PDF-E.pdf (Erişim tarihi: 5 Mart 2019)

İzsu, 2019, <https://www.izsu.gov.tr/tr/TesisDetay/1/9/2?AspxAutoDetectCookieSupport=1> (Erişim: 3 Temmuz 2019)

José da Cunha, M., Barros de Almeida, M. and Carrijo, R. S., 2016, Proposal for an IoT architecture in industrial processes, *12th IEEE International Conference on Industry Applications (INDUSCON)*.

Krcic, S., Pokric, B. and Carrez, F., 2014, Designing IoT architecture(s): A European perspective. *IEEE World Forum on Internet of Things (WF-IoT)*, 79-84 pp.

L. Richardson, S. Ruby, 2008, “Restful web services”, O’Reilly, 17 pp.

Malche, T. and Maheshwary, P., 2017, Internet of Things (IoT) for building Smart Home System, *International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)*.

MongoDB, 2019, MongoDB ile SQL yapı karşılaştırması, <https://docs.mongodb.com/manual/reference/sql-comparison/> (Erişim: 9 Mayıs 2019)

Mqtt Protokolü, 2019, <http://mqtt.org/>, (Erişim: 2 Şubat 2019)

Node.js, 2019, <https://nodejs.org/en/about/> (Erişim: 7 Kasım 2018)

KAYNAKLAR DİZİNİ (devam)

Ribeiro, F. and Metrolho, J., 2016, An open and lightweight Internet of Things platform to facilitate the development of context-aware applications, *Wseas Transactions on Computer Research*, 31-40 pp.

Sails.js, 2019, <https://sailsjs.com> (Erişim: 10 Kasın 2018)

Shadid N. and Aneja S., 2017, Internet of Things: Vision, Application Areas and Research Challenges. International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud).

Thingworx Case Study, 2019, <https://deniscanty.com/2015/04/21/case-study-iot-technology-platform-thingworx-10/> (Erişim: 10 Haziran 2019)

Waterline, 2019, <https://waterlinejs.org/>, (Erişim: 07 Mart 2019)

Windows Azure, 2019, Azure IoT Referans Mimari Modeli, <https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/iot/index> (Erişim: 13 Mayıs 2019)

Yüksel ve Zaim, 2009, “RFID’nin kablosuz iletişim teknolojileri ile etkileşimi”, *Akademik Bilişim*, 2-5 ss.

Zdravković, M., Trajanović, M., Sarraipa, J., Jardim-Gonçalves, R., Lezoche, L., Aubry, A., Panetto, H., 2016, Survey of Internet-of-Things platforms.

TEŞEKKÜR

Bu çalışma süresince teknik bilgisini ve manevi desteğini esirgemeyen eşim Nazım Şahin'e, her daim yanımda olduklarını hissettiğim aileme ve lisans ile yüksek lisans öğrenimim boyunca desteklerini esirgemeyen kıymetli hocam ve danışmanım Doç. Dr. Rıza Cenk ERDUR'a teşekkürü bir borç bilirim.

03 / 09 / 2019

İmzası

Adı-Soyadı

Gizem ÇELEBİ ŞAHİN

ÖZGEÇMİŞ

Gizem ÇELEBİ ŞAHİN

Adres: Gazikent Mahallesi, 73. Sok. No:15, Gaziemir / İzmir

Tel:+90 507 182 80 28

E-mail:gzmcelebi@gmail.com

Kişisel Bilgiler

Uyruğu: T.C.

Doğum Yeri/Tarihi: İzmir / 30.07.1991

Medeni Durum: Evli

Sürücü Belgesi: B sınıfı

Eğitim Durumu

2015-2019: Ege Üniversitesi – Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Yüksek Lisans

2011-2014: Ege Üniversitesi – Mühendislik Fakültesi Bilgisayar Mühendisliği

2005-2009: İzmir 60. Yıl Anadolu Lisesi

Yabancı Dil

İngilizce:İleri Seviye

Programlama Dilleri, Veritabanları ve Teknolojiler

JavaScript, Node.js,

Thingworx App Development Platform,

MongoDB, PostgreSQL,

Java, Python

Staj

Arçelik Bulaşık Makinası Fabrikası-Ankara

Hugo Boss Yazılım Departmanı – İzmir

İş Deneyimi

Tam Zamanlı : 2014 - Günümüz Ericsson Ar-Ge Yazılım Mühendisi, İzmir

Yarı Zamanlı: 2014 - Ericsson Ar-Ge Yazılım Mühendisi, İzmir

2013 - Vestel Elektronik Yazılım Mühendisi, Manisa