УНИВЕРЗИТЕТ У БАЊОЈ ЛУЦИ
ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ

Жељко Трипић

FOG COMPUTING - ПРОГРАМИРАЊЕ IOT GATEWAY-A

дипломски рад

Бања Лука, новембар 2023. године

Тема: **FOG COMPUTING - ПРОГРАМИРАЊЕ IOT GATEWAY-A**

Кључне ријечи:
IoT
IoT gateway
Fog Computing
Cloud Computing
Интернет

Комисија: **доц. др Милош Љубојевић, предсједник**
проф. др Зоран Ђурић, ментор
Александар Келеч, ма, члан

Кандидат:
Жељко Трипић

УНИВЕРЗИТЕТ У БАЊОЈ ЛУЦИ
ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ
КАТЕДРА ЗА РАЧУНАРСКУ ТЕХНИКУ И ИНФОРМАТИКУ

Предмет: МРЕЖНО И ДИСТРИБУИРАНО ПРОГРАМИРАЊЕ

Тема: FOG COMPUTING - ПРОГРАМИРАЊЕ IOT GATEWAY-A

Задатак: Дати кратак преглед Internet of Things и Cloud Computing технологија. Описати IoT *gateway* уређаје и њихову употребу. Образложити потребу за увођењем додатног *fog* међуслоја. Детаљно изложити концепт Fog Computing-а. Дискутовати о добрим и лошим странама Fog Computing-а. У практичном дијелу рада реализовати апликацију за IoT *gateway* уређај која ће имплементирати принципе Fog Computing-а и омогућити комуникацију између IoT *gateway*-а и *cloud* сервиса. Циљна апликација треба да омогући регистрацију новог и пријављивање постојећег IoT уређаја, као и слање агрегираних података, прикупљених са IoT уређаја индустријске машине (екскаватора), ка REST сервисима на *cloud*-у.

Ментор: проф. др Зоран Ђурић

Кандидат: Жељко Трипић (1147/19)

Бања Лука, новембар 2023. године

Садржај

Уз рад је приложен CD.

1. УВОД

Кроз историју људског друштва константно је присутна тенденција да се олакша извршавање различитих људских активности и повећа њихова ефикасност, а, самим тим, олакша и унаприједи свакодневни живот људи. У модерном друштву ова тенденција је, захваљујући развоју различитих рачунарско-комуникационих технологија, прерасла у тенденцију аутоматизације свих типова процеса и активности. Потреба за рационалнијом и ефикаснијом употребом ресурса, подизањем степена сигурности људи и њихове имовине, брз развој технологије, као и све претходно наведене мотивације створили су све неопходне услове за развој концепта под називом „Интернет ствари“ (енг. *Internet of Things* – IoT).

Интернет ствари подразумијева умрежавање свакодневних уређаја попут сатова, фрижидера, машина за веш, система расвјете, али и намјенских специјализованих уређаја попут авионских мотора, *pacemaker* уређаја и др. Умрежавање ових уређаја омогућава људима да их надзиру и контролишу са удаљене локације путем мобилне или *web* апликације посредством интернета, да прикупљају податке и да аутоматизују процесе и активности.

Велика количина прикупљених података и њихова комплексна анализа захтијевају велику процесорску и складишну моћ. Иако је свакодневним уређајима уграђена способност комуникације и процесирања података, постављени захтјеви се нису могли испунити са ограниченим ресурсима који су доступни типичним IoT уређајима. Зато се дошло на идеју да се прикупљени подаци складиште и обрађују на удаљеним специјализованим системима. За сакупљање и слања података ка удаљеним сервисима користе се IoT *gateway* уређаји који се налазе у оквиру локалне мреже IoT уређаја. Велики и константно растући број IoT уређаја захтијева велику скалабилност. Због својих особина које одговарају потребама IoT екосистема, за потребе обраде и похране података често се користи релативно нова и јако скалабилна технологија Cloud Computing-а.

Појава случајева употребе IoT рјешења код којих је вријеме реакције јако важно (попут аутономних возила и *pacemaker*-а), као и све већи број умрежених уређаја који заузимају готово сав доступан пропусни опсег, довела је до тога да се почне размишљати о локалној обради прикупљених података. Иницијална или у неким случајевима комплетна обрада података у самим IoT уређајима, или на специјализованим уређајима у оквиру локалне мреже крајњих IoT уређаја, омогућава да се мања количина података шаље интернетом ка *cloud* сервисима, као и да се избјегне кашњење изазвано мрежном комуникацијом и тако смањи критично вријеме реакције. Нови модел рачунарства, који омогућава имплементацију наведених принципа, је настао под називом Fog Computing. Основна идеја Fog Computing-а је приближити сервисе, налик поједностављеним *cloud* сервисима, извориштy и корисницима самих података, тј. IoT уређајима.

IoT *gateway* уређаји постају један од критичних дијелова IoT екосистема, јер добијају све већа задужења. Они се данас користе за прикупљање, филтрирање, енкрипцију, агрегацију, структурисање, обраду података и њихово просљеђивање ка удаљеним системима посредством интернета. Из поменутих разлога софтвер за ове уређаје постаје све софистицираније и комплексније.

У другом поглављу дат је увид у IoT област. Након основних информација и кратког историјата развоја ове области, укратко је дат осврт на IoT екосистем, његову архитектуру, те технологије који се користе за остваривање комуникације између IoT уређаја. На крају поглавља изложен је широк спектар предности и примјена IoT рјешења.

У трећем поглављу је у кратким цртама изложена област Cloud Computing-а. Продискутовано је како се технологије Cloud Computing-а и IoT-а надопуњују и зашто је баш ова технологија погодна за обраду прикупљених података. У наставку је изложена виртуелизација ресурса *cloud* платформе, као и подјела Cloud Computing модела.

У четвртом поглављу говори се о концепту Fog Computing-а. У првом дијелу поглавља појашњена је потреба за увођењем Fog Computing-а као међуслоја између IoT уређаја и *cloud* сервиса. Након основних информација о овом моделу, дефинисана је улога *fog* слоја у IoT екосистему. Изложене су сличности и кључне разлике између Fog Computing и Edge Computing, односно Cloud Computing модела.

Пето поглавље говори о IoT *gateway* уређајима. Као увод, дат је кратак преглед *gateway*-а. Након основних информација о IoT *gateway* уређајима, укратко је продискутована могућност проширења њихове функционалности и потреба за њиховим програмирањем.

У шестом поглављу представљен је практични дио рада. У практичном дијелу рада реализован је систем за обраду и складиштење IoT података, чији централни дио заузима апликација за IoT *gateway* уређај. Апликација имплементира принципе Fog Computing концепта, прикупља и агрегира сензорске податке, те омогућава комуникацију између IoT *gateway*-а и удаљених *cloud* сервиса. За сваку од апликација које чине систем, укратко, је појашњена њена архитектура и основни ток активности. Такође, описане су и све екстерне библиотеке које су коришћене приликом имплементације апликација.

На крају рада, дат је закључак који представља кратак осврт на изложене технологије и моделе у рачунарству, те ефикасност реализоване апликације и имплементираних принципа Fog Computing-а.

2. INTERNET OF THINGS

Интернет ствари представља тренутно један од најпопуларнијих концепата у домену рачунарско-комуникационих технологија. Развија се великом брзином и непрестано мијења свакодневни живот људи.

IoT представља мрежу међусобно повезаних физичких уређаја који могу да, посредством мреже, размјењују прикупљене податке у реалном времену и да комуницирају са другим системима. Под физичким уређајима се овдје подразумијевају уређаји који не представљају класичне рачунаре (десктоп рачунари, лаптопи, паметни телефони, сервери, итд.) него уређаји попут камера, кућних апарата, сензора, возила, медицинских уређаја, паметних сатова и др. IoT уређаји могу бити и прости сензори и актуатори који имају одговарајуће електронске компоненте, које им омогућавају умрежавање и слање прикупљених података, односно интерпретирање инструкција пристиглих путем мреже, али и много софистицираније уређаји са бројним функционалностима. Овако умрежени физички уређаји изграђују аутоматизоване системе различитих намјена којима је потребна минимална људска интервенција. Главни циљеви IoT технологије су аутоматизација и оптимизација различитих процеса, те смањење потребе за директном људском интервенцијом [1]. Према тренутно доступним информацијама постоји преко 15 милијарди умрежених IoT уређаја, и очекује се да ће број наставити да расте [2].

Сам назив Интернет ствари је помало збуњујући и не осликава право стање ствари. Иако би се из назива дало закључити да би сви IoT уређаји морали бити повезани на интернет (што и јесте чест случај), IoT уређаји не морају нужно бити повезани на интернет. Довољно је да уређаји буду дио локалне рачунарске мреже која им омогућава комуникацију са другим IoT уређајима, који заједно чине неку цјелину, односно систем. С обзиром на то, можда би „Мрежа ствари“ (енг. *Network of Things*) била прикладнији назив. Ипак, за постизање пуног потенцијала IoT-у је неопходан интернет [1], јер се комуникација између удаљених уређаја и система у великој мјери ослања на јавну мрежу.

IoT уводи интернет у свијет физичких уређаја и тиме их приближава виртуелном свијету [1]. Захваљујући IoT-у, физички уређаји могу да користе сервисе доступне на интернету за складиштење и обраду прикупљених података на удаљеним локацијама, као и да буду контролисани, конфигурисани и праћени са удаљене локације.

Један од циљева IoT-а јесте прикупити што већу количину података из реалног свијета, чијом ће се обрадом добити што бољи увид у процесе реалног свијета, а што ће даље омогућити оптимизацију истих. Постојање великог броја IoT уређаја који прикупљају податке у реалном времену доводи до генерисања енормне количине података свакога дана. Прикупљени подаци се могу искористити у бројне намјене, од којих неке могу бити и малициозне. Огромна количина података, велика брзина њиховог прикупљања, њихова неструктурисаност и потреба за њиховом обрадом (често у реалном времену) доводе до појаве феномена који је у рачунарском свијету познат под називом *Big Data*. Како се у IoT домену све врти око података, каже се да је IoT вођен подацима (енг. *data driven*) [3].

Иако су IoT и вјештачка интелигенција различите технологије и концепти, они се међусобно допуњују. Моћ обраде података, закључивања и учења IoT уређаја је ограничена и зато се за обраду прикупљених података често користе методе вјештачке интелигенције, које омогућавају да се из прикупљених података извуку додатне информације, односно да се добије додатна вриједност.

2.1 Историјат

Појави IoT-а претходио је развој ARPANET-а (*Advanced Research Projects Agency Network*), а касније и интернета. За развој IoT-а био је неопходан развој софистицираних сензора као и различитих технологија, прије свега комуникационих, које су омогућиле умрежавање IoT уређаја. У основи свих ових технологија лежи иста идеја, а то је да се омогући комуникација између рачунара, односно других специјализованих уређаја.

Концепт IoT-а је настао прије појаве IoT кованице, током 80их година прошлог вијека. Као први IoT уређај сматра се апарат за продају газираних пића којег су научници на Carnegie Mellon[1] универзитету 1982. године модификовали тако да се може повезати на интернет, што је власницима омогућило да прате стање температуре и залихе пића са удаљене локације [4]. Творац IoT кованице је британски истраживач на MIT-у (Massachusetts Institute of Technology [2]), Kevin Ashton, који ју је први јавно употријебио како би описао концепт повезивања физичких уређаја на интернет употребом RFID (*Radio-frequency identification*) технологије са циљем њиховог надзирања у реалном времену [1].

Иако је концепт IoT-а настао рано, експанзија почиње тек 2000их година, након развоја других технологија. Појава тих технологија је омогућила да се премости разлика између физичког и виртуелног свијета и тако омогући и потпомогне развој IoT рјешења [1]. Такође, развој стандарда од стране Industrial Internet Consortium[3] и Open Connectivity Foundation[4] организација је омогућио интероперабилност хетерогених IoT уређаја.

Технологије које су омогућиле развој и експанзију IoT-а су: софистицирани сензори, енергетски ефикасне електронске компоненте, бежична комуникација, IP (*Internet Protocol*), Cloud Computing, Big Data, вјештачка интелигенција и уграђени системи.

Бежична комуникација омогућава умрежавање IoT уређаја без потребе за физичким медијумима. Омогућена је развојем технологија попут: бежичне локалне рачунарске мреже (енг. *Wi-Fi*), протокола ZigBee, Bluetooth, Z-Wave, LoRaWAN *(Long Range Wide Area Network*), ћелијске мреже (3G, 4G, 5G) и RFID. RFID је бежична комуникациона технологија која се међу првима користила за имплементацију IoT-а. RFID *tag*-ови су омогућили да се физички уређаји идентификује и приступе мрежи.

IP је интернет протокол који омогућава да IoT уређаји буду идентификовани и доступни путем интернета. IPv4 (*Internet Protocol version 4*) верзија овог протокола није била одржива, јер није обезбиједила довољан број адреса. IPv6 (*Internet Protocol version 6*) је верзија IP протокола која је ријешила проблем недостатка адреса и омогућила умрежавање великог броја уређаја.

Cloud Computing је технологија коју карактерише висока скалабилност и која омогућава да се велика количина прикупљених података обради и ускладишти.

Big Data је технологија која омогућава да се у приближно реалном времену обради велика количина података и из њих извуку додатне информације.

Вјештачка интелигенција омогућава да се из прикупљених података извуку обрасци и закључци, као и да се донесу аутономне одлуке без људске интервенције.

[1] https://www.cmu.edu
[2] https://www.mit.edu/
[3] https://www.iiconsortium.org/
[4] https://openconnectivity.org/

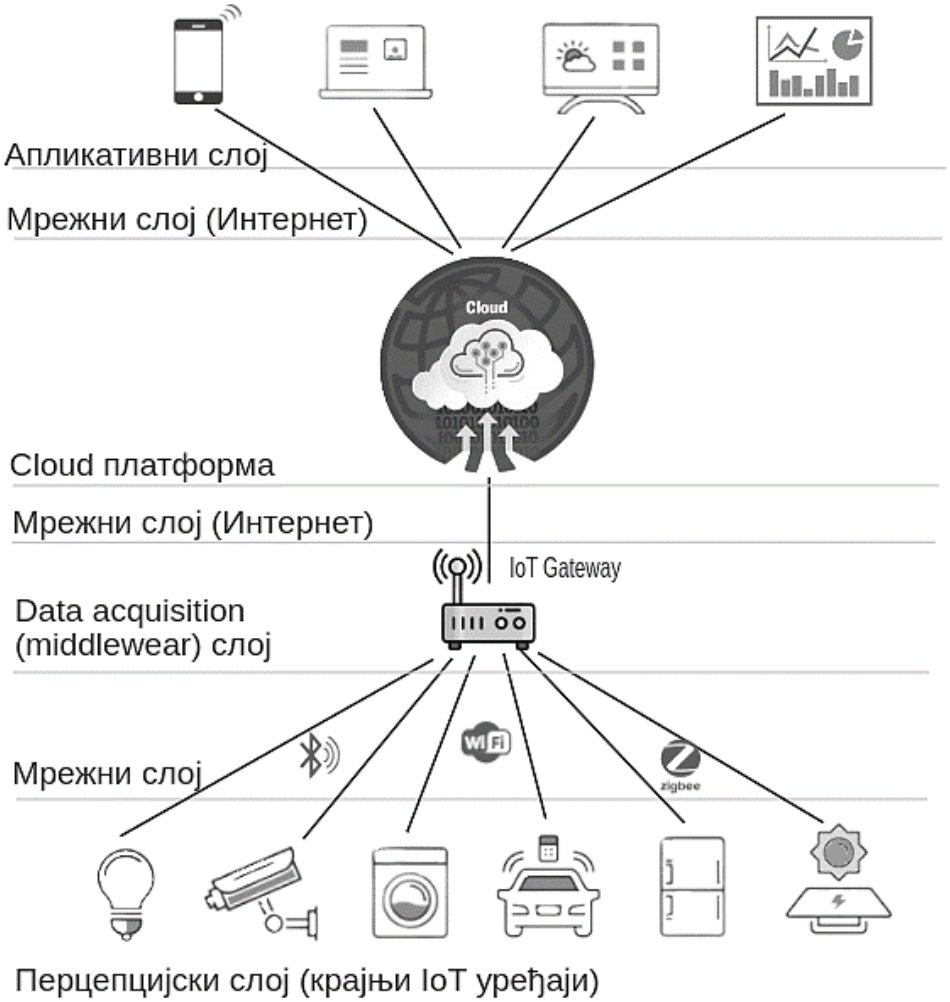
Већина IoT уређаја садржи специјализоване рачунарске системе (уграђене системе) који им омогућавају комуникацију и иницијалну површну обраду прикупљених података.

IoT је једна од технологија која је захваљујући својој распрострањености и готово неограниченим могућностима употребе довела до феномена свеприсутног рачунарства (енг. *ubiquitous computing*) [5]. Свеприсутно рачунарство представља присутност умрежених рачунара у готово свим физичким објектима који нас окружују, чиме се креира умрежена рачунарска околина. Очекује се да у скоријој будућности дође до феномена „Интернета свега" (енг. *Internet of Everything*) [6] који поред повезивања рачунара и физичких уређаја подразумијева умрежавање људи (енг. *Internet of People*), података (енг. *Internet of Data*) и процеса (енг. *Internet of Process*).

Савремена употреба IoT система је довела до појаве нових захтјева које се морају испунити. То се највише односи на употребу IoT рјешења у ситуацијама у којима вријеме реакције и обраде прикупљених података критично. Нове технологије попут Edge Computing-а и Fog Computing-а су управо усмјерене ка рјешавању поменутих проблема.

2.2 IoT екосистем

IoT екосистем је прилично комплексан, јер се састоји од великог броја хетерогених уређаја и технологија које имају широк спектар примјене. Оквирно, екосистем би се могао подијелити на наредних пет слојева [5], приказаних на слици 2.1.



Слика 2.1. – IoT екосистем

2.2.1 Перцепцијски слој

Перцепцијски слој је најнижи али и фундаментални слој чија је основна улога интеракција са физичким свијетом и прикупљање велике количине сирових података. У суштини, то је слој умрежених класичних IoT уређаја (паметни термостати, мрежне камере, паметни сатови, сензори покрета, паметни системи осветљења, итд.). Ови уређаји неизоставно садрже компоненте које им омогућавају комуникацију путем мреже, сензоре и/или актуаторе. Без комуникационих компоненти уређаји ни не могу бити дио IoT екосистема. Сензори омогућавају IoT уређајима да прикупљају податке и имају перцепцију околног физичког свијета, док им актуатори омогућавају да дјелују на физички свијет.

Уређаји који чине овај слој могу бити прости сензори и актуатори којима је придодата могућност мрежне комуникације, али и софистицирани уређаји, попут паметних сатова, мрежних камера и паметних машина за веш, који имају ограничену моћ процесирања и складиштења прикупљених података. Ови уређаји се често називају и крајњим уређајима (енг. *edge devices*). Паметни уређаји (енг. *smart devices*) су подврста крајњих уређаја која ради активно и аутономно, имају процесорске и складишне ресурсе, комуникацијску и напојну јединицу, као и опционо сензоре и/или актуаторе [9].

Сензори су кључан дио IoT екосистема који омогућава прикупљање података из физичког свијета. За њих се каже да су очи и уши IoT система. То су уређаји који имају могућност детекције промјене одговарајућих физичких величина мјерених у физичком свијету. Када сензори детектују промјену вриједности, они генеришу сигнал. У суштини, они врше конверзију вриједности из реалног свијета у сигнале који су разумљиви рачунарима и другим уређајима. Сигнал се даље шаље на обраду и складиштење другим уређајима путем мреже, или се, у случају комплекснијих уређаја, прије слања локално врши иницијална обрада. Постоје различити типови уређаја попут температурних, сензора за дим, покрет, ниво осветљења, влажност, притисак, Lidar сензори, акцелерометри, и др. Сензори су кориштени и прије појаве IoT-a, али су његовом појавом сензорске информације добиле додатну вриједност и постале много употребљивије.

Актуатори су компоненте IoT уређаја и система путем којих се врши дјеловање на физички свијет. Актуатор када добије одговарајућу инструкцију путем мреже он изврши одређену радњу. Ангажовање актуатора се врши на основу одлука добијених обрадом прикупљених података, што је задужење других дијелова IoT уређаја/система. Актуатори обично контролишу физичке уређаје попут мотора, прекидача и екрана путем којих се остварује дејство на околину.

За уређаје овог слоја јако је битна и пожељна способност аутоконфигурације (енг. *plug and play*, *zero-touch deployment*) [3]. Аутоконфигурација је способност аутоматске конфигурације уређаја уз минималну или никакву људску интервенцију. Овакви уређаји већ долазе са иницијалном конфигурацијом која им омогућава да, након спајања на извор електричне енергије и умрежавања, преузму неопходну конфигурацију путем мреже и убрзо буду у потпуности оперативни. Аутоконфигурација нарочито добија на значају када се узме у обзир да је број IoT уређаја јако велики.

2.2.2 Мрежни слој

Мрежа је критичан дио IoT екосистема, јер спаја велики број хетерогених уређаја који размјењују знатне количине података [8]. Мрежа је кључна карика, јер омогућава остваривање IoT концепта. Рачунарска мрежа, у суштини не представља засебан слој, него је она присутна у свим другим слојевима IoT екосистема. Комуникација између засебних слојева се такође врши путем мреже.

Директна конекција свих IoT уређаја на глобалну интернет мрежу није могућа, из разлога што адресирање сваког IoT уређаја са јавном IP адресом није одрживо. Зато се често уређаји умрежавају у локалне мреже формирајући логичке цјелине или комплетне IoT системе. Те локалне мреже обично остварују конекцију са интернетом посредством специјализованих уређаја – IoT gateway-а.

Технологије које се користе за умрежавање уређаја варирају од случаја до случаја употребе. Умрежавање се може остварити посредством физичког медијума или бежично.

Протоколи који се у IoT екосистему користе на физичком слоју референтне OSI (*Open System Interconnection*) мрежне архитектуре су: LoRaWAN, 6LowPan (*IPv6 over Low-power Wireless Personal Area Networks*), NFC (*Near Field Communication*), Bluetooth, WiFi, Ethernet, ZigBee, Sigfox, ћелијске мреже (енг. *Cellular Networks*), RFID и WiSUN (*Wireless Smart Utility Network*).

Неки од протокола на апликативном слоју који користе IoT уређаји су:

- HTTP (*Hypertext Transfer Protocol*) - протокол који се користи за преузимање података са *web* сервера, обично путем REST (*Representational State Transfer*) API-ја (*Application Programming Interface*).
- CoAP (*Constrained Application Protocol*) - лаган и једноставан (енг. *lightweight*) протокол креиран специјално за IoT уређаје. У питању је захтјев-одговор протокол сличан HTTP-у, али намијењен уређајима са ограниченим ресурсима, *low-power* и мрежама са мањим пропусним опсегом. Може се сматрати компресованом верзијом HTTP протокола. У позадини се користи UDP (*User Datagram Protocol*) протокол на транспортном слоју.
- MQTT (*Message Queuing Telemetry Transport*) - једноставан MQ (*Message Queuing*) протокол, који се користи за ефикасну *publish-subscribe* комуникацију између IoT уређаја и *broker*-а. Нарочито је користан у ситуацијама када се подаци шаљу асинхроно заинтересованим странама. Обично се користи за прикупљање података са сензорских уређаја.
- AMQP (*Advanced Message Queuing Protocol*) - добро познат и широко примјењиван MQ протокол који није првенствено намијењен IoT домену због своје комплексности. Пригодан је у ситуацијама када су потребне комплексне интеракције путем порука и када је потребно гарантовати испоруку порука. Много је комплекснији и захтјевнији од MQTT протокола.
- DDS (*Data Distribution Service*) - протокол намијењен за размјену података у реалном времену. Често се користи у ситуацијама у којима је кључна правовремена и поуздана комуникација.
- WebSockets - омогућава успостављање *full-duplex* комуникационе везе путем само једне TCP (*Transmission Control Protocol*) везе. Користи се када је потребно остварити континуалну комуникацију у реалном времену (нпр. надзор IoT уређаја у реалном времену).
- SNMP (*Simple Network Management Protocol*) - користи се за надзор и управљање умреженим уређајима.
- XMPP (*Extensible Messaging and Presence Protocol*) - користи се у случајевима када је потребна комуникација размјењивањем порука у реалном времену између IoT уређаја.

Један од кључних захтјева које IoT мора да испуни јесте могућност скалирања [1]. Могућност скалирања се односи на могућност умрежавања нових уређаја који тада постају дио IoT екосистема без нарушавања постојећих перформанси. Могућност скалирања је захтјев који је првенствено усмјерен ка мрежном слоју.

Хетерогеност је једна од главних особина IoT екосистема [1]. Хетерогеност је посљедица умрежавања уређаја са различитим хардвером и софтвером, употребом различитих технологија повезивања у мреже различитих топологија. IoT уређаји се разликују у величини (велике индустријске машине и мали сензори у оловкама), облику, функцији, протоколима који користе, као и бројним другим аспектима.

За адресирање IoT уређаја не мора се нужно користити IP протокол. Разлог за то је комплексност TCP/IP протокола, која није одговарајућа у свим ситуацијама, нарочито за мале паметне уређаје.

2.2.3 Data acquisition слој

Средњи слој се састоји од IoT gateway и сличних уређаја. Основна улога овог слоја је агрегација података са перцепцијског слоја и њихово даље прослеђивање, обично *cloud* сервисима, на обраду и складиштење. Ово је међуслој између перцепцијског слоја, који прикупља податке, и *cloud* слоја, који их обрађује. Може се рећи и да је ово слој путем које IoT уређаји излазе на интернет.

У зависности од литературе, негдје се ово не издваја као посебан слој, али како IoT gateway уређаји имају суштински другачију функцију од осталих IoT уређаја, згодно их је смјестити у засебан слој. На средишњем слоју се обично врши конверзија комуникационих протокола, а понекад и агрегација, филтрирање, форматирање и структурисање података.

Негдје се зове и слој за прикупљање података (енг. *Data Acquisition Layer*).

2.2.4 Cloud сервиси

Ресурси класичних IoT уређаја су ограничени, па је, самим тим, ограничена и њихова моћ обраде и складиштења података. За потребе складиштења и обраде прикупљених података користе се удаљени сервиси. Иако би се ти сервиси могли извршавати у намјенским рачунарским центрима, то се обично не ради тако, него се подаци шаљу сервисима који се извршавају на *cloud*-у. Након прикупљања података у перцепцијском слоју, подаци се путем интернета и посредством IoT gateway уређаја шаљу ка *cloud*-у.

Одабир Cloud Computing технологије за ове намјене је првенствено из разлога њене изузетне скалабилности. Више информација о *cloud* слоју и Cloud Computing технологији је дато у наредном поглављу.

2.2.5 Апликативни слој

Апликативни слој представља интерфејс ка крајњим корисницима и подразумијева екстерне апликације (енг. *third party applications*) и API-је које оне користе. Ове апликације користе обрађене и ускладиштене податке и *cloud* сервисе за визуелизацију прикупљених података, приказ добијених закључака и информација, напредну обраду података, филтрирање, агрегацију и трансформацију података као и њихову статистичку анализу. Корисници путем корисничког интерфејса којег обезбјеђују ове апликације могу да контролишу и надгледају IoT уређаје, приступају подацима у реалном времену, као и да на основу увида у трендове и статистичке податке у виду контролних табли (енг. *dashboard*), табела и графова, доносе даље одлуке.

2.3 Области примјене

Постоји јако велики број области људске дјелатности у којима се IoT рјешења већ користе. IoT се у тим областима користи из различитих мотивација, попут аутоматизације, уштеде ресурса, удаљене контроле, оптимизације процеса, олакшаног праћења процеса и објеката, повећања степена сигурности и др. Непрестано се појављују нови иновативни примјери употребе IoT рјешења и за сада се не виде границе могућности IoT технологије. Очигледно је да је IoT постао дио свакодневног живота људи и да на њега има јако велики утицај. За очекивати је да ће се људска свакодневница константно мијењати упоредо са појавом и интеграцијом нових IoT уређаја.

Неки од примјера употребе IoT рјешења су дати у наставку.

2.3.1 Примјена IoT рјешења у индустрији

Интернет ствари који се користи у индустријском сектору се назива „Индустријски интернет ствари“ (енг. *Industrial Internet of Things* – IIOT). Интеграција IIoT рјешења у индустријска постројења је уско повезана са четвртом индустријском револуцијом (енг. *Industry 4.0*) [6] која подразумијева дигитализацију и аутоматизацију индустрије и производних процеса. IoT рјешења се у индустрији користе са циљем постизања “паметних” постројења. “Паметна” постројења подразумијевају висок степен аутоматизације, могућност удаљеног праћење ресурса и опреме, оптимизацију ланаца снабдијевања, подизање ефикасности производних процеса, смањење или укидање времена застоја постројења, предвиђање потенцијалних кварова и неопходне поправке опреме, удаљено управљање постројењем, као и што рационалнију употребу ресурса.

2.3.2 Примјена IoT рјешења у саобраћају, логистици и транспорту

Подаци прикупљени са IoT уређаја се могу искористити за проналажење најоптималнијих рута, детекцију уских грла која стварају гужве у саобраћају, смањење загађења који потиче од издувних гасова возила, праћење робе у реалном времену и надзор услова у складиштима робе. Уз помоћ IoT рјешења може се пратити стање возила и услова на путу, надзирати кључни дијелови саобраћајне инфраструктуре, предвиђати потенцијални кварови возила и креирати мрежа аутономних возила која ће смањити број саобраћајних незгода и допринијети сигурности путника.

2.3.3 Примјена IoT рјешења у паметним градовима и паметним кућама

Уз помоћ IoT технологија могуће је креирати “паметне” градове. “Паметни” градови подразумијевају постојање IoT система који служе за оптимизацију употребе енергената, управљање отпадом, видео надзор одређених јавних површина, повећање ниво безбиједности грађана и њихове имовине, контролу животних услова и смањење загађења, праћење возила и оптимизацију јавног превоза, управљање паркинзима (“паметни” паркинг) и градском расвјетом (“паметна” расвјета).

IoT уређаји омогућавају људима да имају контролу и надзор над својом кућом са удаљене локације. То подразумијева увид у различите параметре, попут температуре и влажности, контролу над видео надзором и системом за освјетљење. Све популарнији су и сигурносни системи за куће (енг. *Home Security System*) који представљају мрежу видео камера, сензора покрета, аларма, сефова, детектора дима, и других сличних уређаја којима се може удаљено управљати.

2.3.4 Примјена IoT рјешења у метеорологији

Подаци прикупљени путем IoT уређаја се могу искористити за предвиђање временских услова као и за упозерења на потенцијалне опасности (попут радијације, земљотреса, ерупције вулкана, итд.), и то све без опасности по људске животе.

2.3.5 Примјена IoT рјешења у пољопривреди

IoT је нашао своју примјену и у пољопровреди. IoT рјешења омогућавају прикупљање података о стању и параметрима усјева, на основу којих се доноси одлука о даљем дјеловању на усјеве, посредством других IoT система (системи за наводњавање, противградни системи, итд.), са удаљене локације .

2.3.6 Примјена IoT рјешења у медицини

IoT уређаји се у медицини користе за прикупљање података о стању пацијената, уочавање аномалија у функционисању људског организма и правовремено дјеловање на људски организам.

2.4 Предности и изазови IoT технологије

Бројне предности које са собом доноси примјена IoT технологије су један од главних разлога за тако брзу интеграцију IoT рјешења у свим сферама људског живота. Иако је позитиван утицај IoT-а присутан и видљив у свим областима примјене, постоје и бројни изазови са којима се развој и интеграција IoT рјешења сусрећу. Вртоглаво брз раст умрежених IoT уређаја са собом доноси нове проблеме и изазове.

Употреба IoT технологије доноси разне предности, као што су: већи степен аутоматизације процеса, већи ниво сигурности, повећана ефикаснот и продуктивност процеса, отимизација процеса, уштеда енергената и ресурса, смањење отпада (што доводи до смањења трошкова и загађења околине), надзор, праћење и конфигурација објеката и процеса са удаљене локације, предвиђање потенцијалних кварова опреме и правовремено одржавање, боље разумијевање животне средине и процеса који се одвијају на основу прикупљених података и боље корисничко искуство.

Неки од проблема и изазова са којим се сусреће приликом развоја и употребе IoT технологија су:

- Сигурност - слаба тачка већине IoT уређаја је [5]. Имплементација сигурносних мјера и механизама у IoT домену заостаје за осталим областима рачунарства. Неки од разлога за тако стање су велики број хетерогених IoT уређаја који потичу од великог броја произвођача, ограничени ресурси уређаја, као и недовољно тестирање. Низак ниво сигурности IoT система представља потенцијалну мету за нападаче који могу да упадом у систем дођу до осјетљивих података или чак креирају мрежу ботова за извође дистрибуираних DOS (енг. *Denial Of Service*) напада.
- Приватност података - IoT уређаји прикупљају велику количину података, од којих неки могу бити приватни или сигурносно осјетљиви. Пренос и складиштење таквих података захтијева одређене сигурносне мјере, као придржавање одговарајућих законских прописа. Неауторизовано праћење људи и њихове имовине представља нарочит проблем.

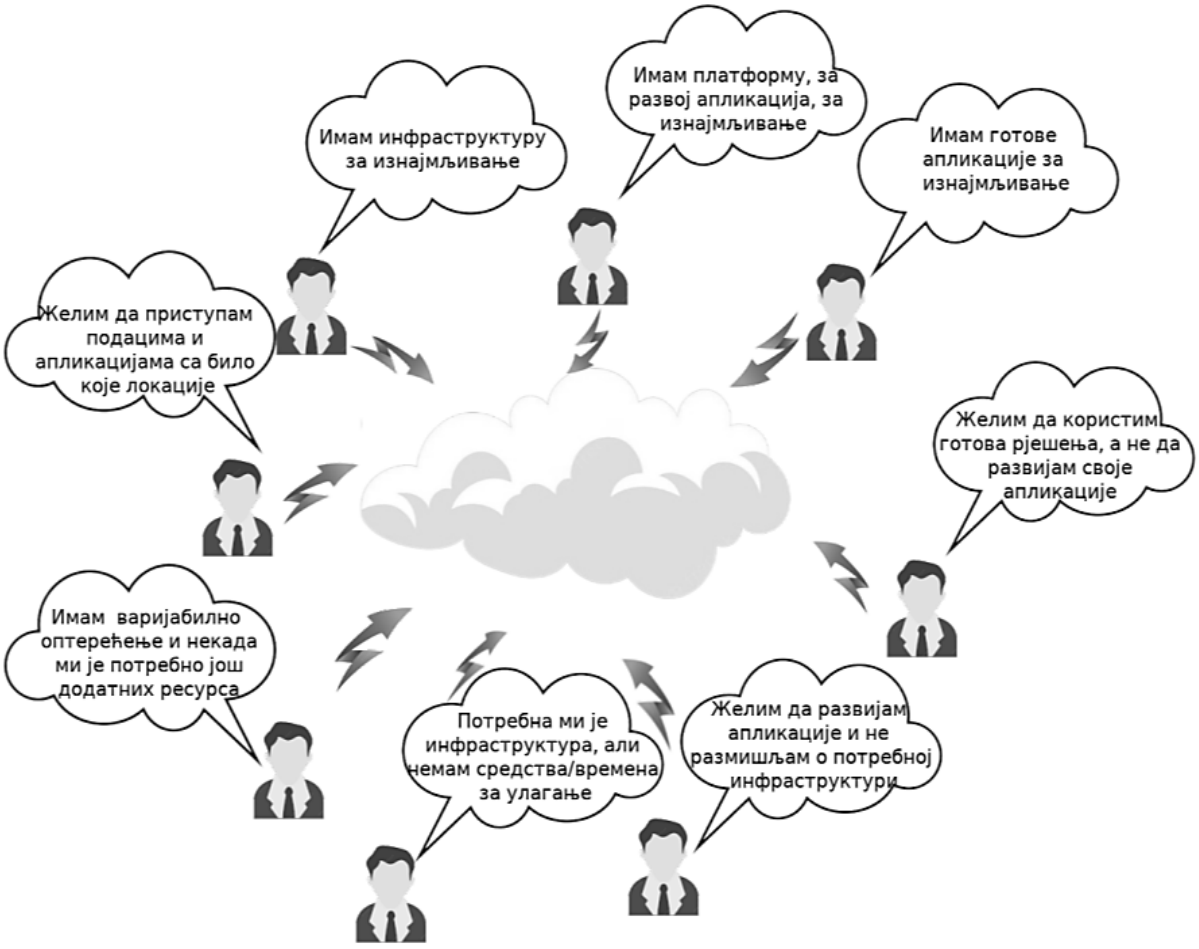
- Интероперабилност - како би концепт IoT-а уопште и функционисао неопходно је да различити IoT уређаји могу заједно раде и међусобно комуницирају. Сама интероперабилност између IoT уређаја је проблематична из три разлога: хетерогености уређаја (разлике у хардверу и софтверу), великог броја произвођача IoT опреме и недостатка адекватних стандарда.
- Употреба енергије и утицај на околину - IoT уређаји користе електричну енергију за рад. Иако се води рачуна о потрошњи електричне енергије и стално се ради на оптимизацији, све већи број умрежених уређаја троши јако велике количине енергије. Нарочито су проблематични уређаји који раде на батерије.
- Комплексност IoT мреже - велики број умрежених уређаја, као и различите мрежне топологије и технологије које се користе за комуникацију, константно подижу ниво комплексности рачунарске мреже која повезује IoT уређаје. Проблематично је и обезбјеђивање довољно великог пропусног опсега за све умрежене уређаје [5].
- Генерисање велике количине података - IoT свакога дана прикупља огромне количине података које је изазовно обрадити у реалном времену. Прикупљени подаци захтијевају велику процесорску моћ за обраду и складишта података великог капацитета.
- Власништво над прикупљеним подацима - одређивање власништва над подацима у случајевима када у прикупљању, процесирању и складиштењеу учествује више страна, представља посебан проблем. Поријекло података је, такође, некада проблематично [1].

3. CLOUD COMPUTING

Рачунарство у облаку (енг. *Cloud Computing*) је широк концепт у рачунарству који је брзо попримио глобалне размјере и постао јако популаран. Овај нови модел рачунарства али уједно и нови модел пословања (због новог модела плаћања ресурса/сервиса) већ има очигледан утицај на бројне аспекте људског друштва [10].

Ријеч *cloud* у називу је метафора за интернет, тј. глобалну рачунарску мрежу. Употреба баш овог термина је посљедица честе праксе приказа интернет инфраструктуре у виду облака на различитим дијаграмима од стране рачунарских инжењера [10].

Основни концепт на којем је Cloud Computing базиран јесте изнајмљивање ресурса путем интернета, а не њихова класична трајна продаја, као и услуживање што већег броја корисника са што мање ресурса. Неки од мотива употребе *cloud* платформи су приказани на слици 3.1.



Слика 3.1 – Cloud Computing мотивација

Постоје бројне дефиниције Cloud Computing-а. Према NIST-у[5] (National Institute of Standards and Technology), Cloud Computing је „плати колико потрошиш“ модел рачунарства код којег се конфигурабилним рачунарским ресурсима приступа на захтјев, у кратком року, путем мреже, и то све уз минималне административне напоре и минималну комуникацију са даваоцем услуга. Према IBM-у (International Business Machines[6]) ово је модел базиран на испоруци сервиса, софтвера и хардверских ресурса путем јавних и приватних мрежа.

[5] https://www.nist.gov/
[6] https://www.ibm.com/us-en

Cloud Computing је, у суштини, модел код којег се различитим динамичким, скалабилним, виртуелизованим рачунарским ресурсима, који су у виду сервиса, приступа путем интернета [11]. Рачунарски ресурси се у Cloud Computing-у, због начина испоруке, називају још и сервисима. Под ресурсима се подразумијевају физички рачунарски ресурси, попут складишта података, процесорских јединица, мрежне опреме, али исто тако и различити софтверски пакети. Ови ресурси су физички смјештени у великим рачунарским центрима провајдера Cloud Computing услуга. Како су рачунарски центри често опремљени са савременом технологијом, корисници обично добијају и добре перформансе. *Cloud* платформе из корисничке перспективе излгедају као бесконачни извори рачунарских сервиса.

Идеја, која је преузета из концепта Grid Computing-а, јесте да се рачунарски ресурси користе и наплаћују баш као што се ради са струјом и водом. То значи да су доступни одмах на захтјев, путем одговарајуће мреже, и да се плаћају према потрошњи.

Концепт Cloud Computing-а помијера мјесто процесирања и складиштења података са локалних рачунара на удаљене рачунаре негдје у мрежној инфраструктури, тј. „негдје на интернету“ [10], што је приказано на слици 3.2. Ово није нов приступ. Кроз историју рачунарства често су се смјењивали трендови обраде података на удаљеним машинама и обраде на локалној машини. Сличан концепт обраде података на једној или више специјализованих удаљених машина, којима се приступа путем терминала (који су у неким случајевима јако једноставни уређаји) је присутан у Client-Server Computing-у, Grid Computing-у, Mainframe Computing-у и дистрибуираном рачунарству.

Слика 3.2 – Cloud Computing платформа

За Cloud Computing се каже да има готово тренутан одзив, јер за разлику од традиционалног рачунарствена, нема чекања на испоруку, конфигурацију и инсталацију неопходне инфрструктуре, тј. хардверских и софтверских ресурса.

Некада је тешко повући црту између локалног и Cloud Computing-а само из разлога што су класични *cloud* сервиси већ доступни и јако популарни на нашим рачунарима, а уз то пружају и корисничко искуство блиско класичним платформски специфичним (енг. *native*) апликацијама. Примјери таквих сервиса су: Google Maps[7], Office 365[8] и Gmail[9].

Уколико је креиран рачунарски систем којем се може приступити путем мреже, који има репозиториј ресурса, које може да подијели путем мреже, и који омогућава брзо скалирање услуга, онда је вјероватно креирана једноставна *cloud* платформа.

Приликом пројектовања неког сервиса потребно је испитати да ли сервис истовремено задовољава сљедећа три услова [10], који упућују на то да је *cloud* одговарајуће окружење за дати сервис:

- Да ли сервис треба да буде доступан у било које вријеме, са било којег мјеста, путем било ког уређаја који има приступ интернету, без потребе за инсталацијом.
- Да ли сервис мора бити стално доступан и поуздан.
- Сервис има довољно велики број корисника.

Класичан примјер употребе Cloud Computing концепта јесу Chromebook лаптопи. Ови лаптопи имају тек толико ресурса за извршавање ChromeOS оперативног система, док се сва обрада и складиштење података обавља путем Google-ових *cloud* сервиса, што их чини уређајима усмјереним ка *cloud* платформама (енг. *cloud-centric device*).

Најпознатији провајдери Cloud Computing услуга су: Google Cloud [10], Amazon Web Services[11], Microsoft Azure[12], IBM Cloud[13] и Alibaba Cloud[14].

3.1 Карактеристике Cloud Computing-а

Важна особина овог концепта је та да приступ ресурсима у *cloud*-у није ограничен удаљеношћу, нити физичком локацијом корисника и ресурса, све док постоји стабилна веза ка интернету. Приступ интернету и било какав уређај који има подршку за *web* читаче су довољни предуслови за приступ *cloud* сервисима. Иако не постоје физичка ограничења по питању удаљености, мања удаљеност клијента од ресурса обично (у зависности од брзине линка) значи мање кашњење и, самим тим, боље корисничко искуство. Из овог разлога (али и других, попут сигурносних) провајдери обично дистрибуирају ресурсе, тј. своје рачунарске центре на удаљене физичке локације.

Иако дијели доста сличности са другим концептима у рачунарству, постоје и многобројне карактеристике које концепт чине јединственим. Главне карактеристике Cloud Computing-а ће бити дате у наставку.

[7] https://www.google.com/maps
[8] https://www.office.com/
[9] https://www.google.com/gmail/about/
[10] https://cloud.google.com/
[11] https://aws.amazon.com/
[12] https://azure.microsoft.com/en-us
[13] https://www.ibm.com/cloud
[14] https://sg.alibabacloud.com/en

3.1.1 Ресурси су доступни према потреби, на захтјев (самоуслуживање)

Клијент може да приступи ресурсима у *cloud*-у путем *web* читача, упућивањем захтјева на одговарајуће адресе, и у већини случајева није потребна интервенција провајдера услуга. Захтјев се аутоматски одобрава, уколико су захтијевани ресурси доступни и уколико је клијент спреман да плати свој утрошак. Клијент, тј. потраживач услуга и ресурса може бити било ко, ко жели да плати за коришћење ресурса.

Карактеристике приступа ресурсима: клијент приступа ресурсима онда када су му потребни, клијент користи (и плаћа) ресурсе само док су му потребни, клијент користи оне ресурсе који су му потребни и клијент користи онолико ресурса колико му је потребно.

3.1.2 Зависност од мреже (интернета)

Концепт Cloud Computing-а је у потпуности зависан он интернета, јер се приступ свим ресурсима, одвија управо путем мреже. Рачунарска мрежа је кључан, али уједно и критичан дио Cloud Computing екосистема и често представља уско грло. Квалитет услуге (енг. *Quality of Service* - QoS) и корисничко искуство су увелико зависни од треће стране – интернет сервис провајдера. За остваривање пуног потенцијала Cloud Computing концепта јако је битна добра мрежна инфраструктура која ће обезбиједити довољан пропусни опсег и минимално кашњење [10].

3.1.3 Фондови ресурса

Ресурси се логички организују у фондове ресурса (енг. *resource pool*). Додјела ресурса клијентима се врши динамички, тако што се они на захтјев повлаче из фонда и дају клијентима на коришћење. Након што се ресурс повуче из фонда, он тамо више није доступан, а након што клијент заврши са коришћењем ресурса, ресурс се динамички враћа у фонд и постаје поново доступан на захтјев другим клијентима.

Само груписање сродних ресурса (процесори, складишта података, софтвер, итд.) у логичке цјелине не представља концепт фонда ресурса, већ њихову класификацију. За креирање фонда ресурса потребно је ресурсе подијелити на мање логичке цјелине како би се у корисничке потребе могле задовољити употребом минималне количине ресурса [10].

На примјер, посматрајмо ситуацију у којој клијент захтијева на коришћење 10GB складишта података. Нека се у рачунарском центру провајдера користе дискови величине 1TB. Давалац услуга мора да задовољи клијентске потребе и омогући им коришћење минимално 10GB складишног простора. У овој ситуацији постоје двије опције. Прва је да одбије клијента, јер нема ресурсе који одговарају клијентским потребама, чиме се губе клијенти и зарада. Друга је да клијенту уступи цијели диск величине 1TB на коришћење. У овом случају смо задржали и задовољили клијента, али смо му дали више ресурса него што му је потребно. Ово је примјер неефикасне употребе рачунарских ресурса. Вишак ресурсе се може искористи за услуживање других клијената и остваривање додатног профита. Из овог разлога ресурси се дијеле на мање логичке цјелине. На примјер дискови се издијеле на партиције од 10GB. Надаље се ове партиције користе као основне јединице ресурса. Па тако, уколико пристигне захтјев за 100GB складишног простора, клијенту ће бити дато десет логичких јединица ресурса на коришћење. У случају процесорских јединица подјела се може извршити на логичка (виртуелна) језгра.

Ове логичке јединице ресурса су у потпуности транспарентне за клијенте, тј. они ни не знају са колико логичких јединица су задовољене њихове потребе. Поред овакве подјеле на логичке цјелине у Cloud Computing-у се врши и виртуелизација ресурса, о којој

ће касније бити више говора. Управо је виртуелизација кључна технологија на којој је заснован Cloud Computing.

Ова карактеристика је предуслов за самоуслуживање ресурсима на захтјев.

3.1.4 Скалабилност, еластичност и флексибилност ресурса

Клијенти користе онолико ресурса колико им је потребно, све док постоји потреба. Када потреба за ресурсима нестане, ресурси се могу једноставно вратити под контролу провајдера чиме престаје мјерење утрошка. Оваква природа ресурса их чини еластичним и флексибилним.

Ресурси које клијент користи се, исто тако, могу у јако кратком року скалирати на основу клијентских потреба. Скалирање може бити на захтјев или аутоматско на основу унапријед утврђене стратегије (нпр. количина саобраћаја). Скалабилност омогућава да се количина заузетих ресурса прилагођава тренутним потребама клијента.

Комбинација ових особина рјешава чест проблем варијабилног оптерећења ресурса. Уколико се ресурси обезбјеђују и набављају према минималном оптерећењу, у тренуцима повећаног оптерећења доћи ће до губитка клијената и немогућности обраде свих захтјева. Набавка ресурса према највећем оптерећењу доводи до слабе искоришћености ресурса, а набавка базирана на просјечном оптерећењу умјерено доводи и до једног и до другог проблема [10].

Уколико клијент у одређеним временским интервалима или данима очекује веће оптерећење корисника, једноставно и аутоматизовано се може извршити скалирање ресурса. На примјер, за потребе интернет пословања неке компаније, додатни ресурси се могу додијелити за вријеме празника или распродаја.

Постоји могућност и приоритетизације сервиса на основу дефинисаних политика која омогућава динамичку прерасподјелу ресурсима и повећану ефикасност [10].

3.1.5 Плати колико потрошиш

Наплата коришћених сервиса се врши на основу измјереног утрошка. Утрошак ресурса може бити изражен у виду заузетог складишног простора, броја упућених захтјева, времена које је ресурс био доступан клијенту и др. У већини случајева нема потребе за плаћањем унапријед, иако постоје пакети базирани на претплати.

Овакав модел пословања укида потребу за куповином и одржавањем, односно, уопште посједовањем потребног намјенског хардвера и/или софтвера. Фирме уопште не морају да брину о потребној хардверској инфраструктури нити о инсталацији неопходног софтвера. Хардверски и софтверски ресурси се изнајмљују од провајдера према личним потребама. Предуслов за овај модел плаћања јесте да се ресурси испоручују у виду сервиса.

3.1.6 Сервиси нису брига клијента

О сервисима (било да је у питању хардверска инфраструктура, оперативни системи, извршна окружења или готова софтверска рјешења) које користе клијенти брине провајдер услуга. Провајдер је задужен да одржава и ажурира ресурсе, да обезбиједи њихову сигурност, поузданост и константну доступност.

3.2 Историјат

Развој и популаризација интернета су оптеретили постојеће складишне и процесорске капацитете. Долази до употребе умрежених рачунара ограничених ресурса за обраду и складиштење података. Развијају се технологије које су омогућиле еластичан и флексибилан рад са хардверским ресурсима, на првом мјесту се мисли на виртуелизацију.

Употребу термина Cloud Computing за опис новог модела у рачунарству, је јавно подстакао Eric Schmidt, директор у Google-у, током свог излагања на Search Engine Strategies конференцији 2006. године. Зачеци Cloud Computing концепта сежу до 60их и 70их година прошлог вијека када долази до развоја виртуелизације, *time-sharing* технологија које су омогућиле истовремену употребу рачунара и ресурса од стране више корисника, појаве приватних виртуелних мрежа (енг. *Virtual Private Networks* - VPN), као и услуге *hosting-a* [10]. Идеје и концепти који се користе у Cloud Computing-у су одређеној мјери били присутни и у другим моделима рачунарства који су му претходили.

Још од свог оснивања, 1999. године, фирма Salesforce[15] је почела са испоруком апликација и сервиса путем своје једноставне *web* странице, због чега се и сматрају пионирима SaaS *cloud* модела. Фирма Amazon[16] је 2002. године представила своју Cloud Computing платформу под називом Amazon *web* Services. Овај догађај се узима за почетак доба модерног *cloud* рачунарства. Иста компанија 2006. године представља двије нове *cloud* платформе, Elastic Compute Cloud (EC2) и Simple Storage Service (S3). Током 2008. године Google је развија нову *cloud* платформу Google App Engine, која одговара PaaS моделу омогућава развој и *hosting* апликација. Microsoft[17] 2009. година представља своју *cloud* платформу под називом Windows Azure (данас познату као Microsoft Azure).

За разлику од других провајдера Cloud Computing услуга, Microsoft има нешто другачију политику и залаже се за хибридни приступ, комбиновањем *cloud* и локалног рачунарства [10]. Разлог за то је тај што би потпуним преласком рачунарства на Cloud Computing концепт, дошло до изумирања рачунарства на локалним машинама, а на овом пољу Microsoft већ има јако велики утицај. Microsoft се залаже за то да се дио обраде и складиштења података ради на крајњим уређајима, тј. клијентима. Овакав приступ има чврсте аргументе, а то је да су локални ресурси за уобичајене потребе јефтинији и доступнији и да постоје ограничења по питању пропусног опсега (који расте, али расте и величина садржаја који се шаље путем мреже).

Предуслови који су омогућили развој овог концепта у рачунарству: развој интернета (добра мрежна инфраструктура, велики пропусни опсег уз минимално кашњење), напредак рачунарских центара, развој Big Data концепта [10] и виртуелизације.

Постоје бројни концепти и приступи у рачунарству који су претходили Cloud Computing моделу. Са готово свима њима Cloud Computing дијелим мање или више карактеристика, и зато се може рећи да је то резултат интеграције и еволуције различитих приступа у рачунарства [10].

[15] https://www.salesforce.com/
[16] https://www.amazon.com/
[17] https://www.microsoft.com/bs-ba/

Неки од концепата рачунарства (енг. *computing model*) који су претходили Cloud Computing-у су:

- Mainframe Computing - централизовани рачунарски модел код којег су ресурси сконцентрисани у моћним *mainframe* рачунарима, којима се приступа путем једноставних терминала. Процесирање и складиштење података се не одвија локално, него искључиво на *mainframe* рачунарима. За овај концепт се у литератури може пронаћи тврдња да представља и први Cloud Computing концепт [10].
- Utility Computing - основна идеја јесте плаћати рачунарске ресурсе онолико колико су кориштени. Код овог концепта не постоји потреба за посједовањем ресурса. За разлику од Cloud Computing-а, Utility Computing је само специфичан начин наплате рачунарских ресурса и не спецификује бројне друге аспекте, попут ресурсима (сасвим у реду је ресурсима приступати на мјесту гдје се они физички налазе). За Cloud Computing се може рећи да имплементира концепт Utility Computing-а.
- Client-Server - иако се чини да је Cloud Computing модел идентичан клијент-сервер моделу, он је, у ствари, много више од тога и представља унапређење поменутог модела, код којег се сви рачунарски ресурси испоручују у виду сервиса. Приступ ресурсима у Cloud Computing-у је базиран на овом принципу, с тим да *cloud* (серверска страна) представља за клијенте привидно бесконачан фонд ресурса. Клијенти не морају само да користе ресурсе који су доступни на серверу, него могу и да подигну своје (нпр. *hosting* клијентских апликација).
- Cluster Computing - основна идеја јесте да се обрада података врши у серверским кластерима који представљају јако спрегнуте рачунарске ресурсе који су намијењени за постизање јединственог циља. У Cloud Computing-у је такође користе кластери рачунарских ресурса, с тим да је много мања спрега, а већа аутономност ресурса.
- Service Computing - идеја је да се апликације испоручују клијентима у виду сервиса без потребе за куповином унапријед. Апликације, тј. сервиси треба да буду слабо спрегнути и независни. Овај концепт је за разлику од Cloud Computing-а ограничен на софтверски (апликативни) ниво. Доста сличности дијели са Software as a Service моделом.
- Distributed Computing - у суштини, ради се о извршавању послова на већем броју умрежених, дистрибуираних, независних машина. Мрежа тих машина је у потпуности транспарентна и клијенту је представљена као јединствен ентитет. Карактеристике овог концепта су скалабилност, конкурентност, хетерогеност, континуална доступност и отпорност на отказе. Циљ је био ефикасна употреба ресурса и већа моћ процесирања без употребе моћних рачунара.
- Grid Computing - овај концепт је намијењен за извршавање комплексних научних израчунавања. Дистрибуирани чворови формирају виртуелне супер рачунаре. Сваки посао се дијели на мање цјелине које се прослеђују чворовима на паралелну обраду. Након завршетка обраде, сваки чвор враћа резултата обраде централном *master* чвору. Овај концепт је првенствено намијењен за ситуације у којима је потребна велика моћ обраде података, као на примјер комплексна научна израчунавања и симулације. Сличност између ова два концепта је у томе што се посао извршава на већем броју дистрибуираних чворова. За разлику од дистрибуираног рачунарства, хардверски чворови су се налазили на удаљеним географским локацијама (нпр. припадали су различитим организацијама).

За развој Cloud Computing-а се може рећи да је природна појава. Кроз историју развоја рачунарства смјењивале су се фазе централизације и децентрализације, односно дистрибуције рачунарских ресурса. У почетку, усљед техничких услова и финансијских фактора централизација је била једина могућност. Напретком рачунарске технологије, развојем персоналних рачунара и смањењем цијене дошло је до дистрибуирања и децентрализације рачунарских ресурса. Сада је поново, усљед повећане мобилности корисника, потребе за сталном доступношћу ресурса и развојем мрежне инфраструктуре, популаран вид централизација ресурса који је дошао са развојем Cloud Computing-а [10]. Иако је помало парадоксално, паралелно је присутна дистрибуција обраде и складиштења података.

Етапе развоја Cloud Computing технологије [10]:

1. Could Computing 1.0 - примјена виртуелизације рачунарских ресурса у рачунарским центрима је омогућила да се апликације и извршна окружења логички одвоје од хардверске инфраструктуре. Апстракција хардвера је омогућила да више апликација и извршних окружења истовремено дијеле исте хардверске ресурсе. Употреба софтвера за распоређивање послова и ресурса, као и виртуелизација ресурса је омогућила да се подигне ниво ефикасности и искоришћености ресурса.
2. Cloud Computing 2.0 - испорука ресурса у виду сервиса и аутоматизовано управљање инфраструктуром. Овакав приступ је увелико допринио брзој испоруци ресурса као и еластичности услуга које се пружају.
3. Cloud Computing 3.0 - прелаз са класичне слојевите организације софтвера на развој и примјену сервисно орјентисане архитектуре софтвера. Сервисно орјентисана архитектура и Cloud Computing дијеле битне карактеристике попут слабе спреге и дистрибуираности компонената. Примјена новог архитектуралног стила је омогућила и нове моделе Cloud Computing-а попут FaaS-а.

3.3 IoT и Cloud Computing

Интеграција IoT и Cloud Computing технологија се понегдје у литератури назива „облак ствари“ (енг. *Cloud of Things* – CoT).

Употреба Cloud Computing технологија у постојећем IoT екосистему је била логичан корак усљед све већих потреба за: поузданим складиштењем све веће количине прикупљених података, великом процесорском моћи која ће обрадити податке, једноставним скалирањем инфраструктуре (усљед раста броја умрежених уређаја), једноставним приступом подацима са различитих локација, управљањем и праћењем све већег броја уређаја.

Провајдери *cloud* сервиса обезбјеђују IoT *cloud* платформе које омогућавају једноставно повезивање IoT уређаја, складиштење, агрегацију, обраду и визуелизацију података, удаљено праћење и управљање уређајима, праћење аналитике у реалном времену, као и употребу других *cloud* сервиса од стране уређаја. Неке од IoT *cloud* платформи су: AWS IoT, Microsoft Azure IoT, Google Cloud IoT, IBM Watson IoR, Particle IoT, ThingSpeak.

Интеграција ових технологија је омогућила да велика количина прикупљених сензорских података добије смисао и да се на основу њих извуку закључци и донесу

одлуке. Захваљујући Cloud Computing-у IoT подаци се могу ускладиштити, обрадити, анализирати и визуелизовати, те им се може лако приступити. Примјена Cloud Computing технологије је надомјестила недостатке IoT платформе које се односе интероперабилност, поузданост, скалабилност и флексибилност [9]. Често *cloud* платформе служе и као посредници у комуникацији између IoT уређаја. IoT системи се могу посматрати као извори података, а *cloud* сервиси као њихова одредишта.

Интеграција Cloud Computing-а са IoT системима омогућава: већу флексибилност и скалабилност платформе, увид у сакупљене податке готово у реалном времену, предиктивну анализу која може да детектује аномалије у функционисању IoT система, доношење одлука на основу процесираних података (подизање нивоа аутоматизације), визуелизацију података, агрегацију и размјену података између удаљених IoT система, поуздано удаљено складиштење прикупљених података, детаљну и комплексну обраду података, примјену техника вјештачке интелигенције при обради података, удаљено праћење, контролу, конфигурацију и одржавање IoT уређаја.

3.4 Виртуелизација ресурса Cloud платформе

Виртуелизација је кључна технологија у Cloud Computing моделу. Омогућава униформно управљање и једноставно скалирање хетерогених физичких ресурса [10], те доприноси ефикаснијој употреби постојећих ресурса *cloud* платформе. Виртуелизација је у потпуности транспарентна за клијенте. Клијент не мора и не треба ништа да зна о инфраструктури, имплементацији сервиса и технологијама које омогућавају испуњавање његових захтјева и потреба. Развој виртуелизације је омогућио и развој Cloud Computing-а. Иако је виртуелизација кључна технологија, Cloud Computing користи бројне друге технологије (попут оних које омогућавају коришћење контејнера, оркестрацију, рутирање, сигурност, као и софтвера за управљање и аутоматизацију) и представља много више саме виртуелизације ресурса.

Основни циљ виртуелизације јесте апстракција хардверских ресурса која омогућава да се превазиђу различита ограничења која намеће физичка природа хардвера. Виртуелизација омогућава креирање виртуелних логичких репрезентација хардверских ресурса попут процесора, меморије, складишта података, мрежних уређаја итд. Захваљујући виртуелизацији, на истој хардверској платформи, тј. физичкој машини, може се наћи већи број оперативних система и извршних окружења, у оквиру којих се могу извршавати различите апликације, при чему сав тај софтвер дијели исте хардверске ресурсе.

Рачунарски системи су обично подијељени на слојеве, почевши од хардвера, преко оперативног система до апликативног софтвера. Виртуелизација омогућава апстракцију и логичко раздвајање тих слојева и ресурса, при чиме се креирају нови виртуелни слојеви. Врши се апстракција ресурса на нижим нивоима које користе виши слојеви и тако смањује степен спреге и постиже се транспарентност нижих слојева.

Основна јединица која изграђује *cloud* платформу, обрађује податке и извршава задатке на *cloud* платформама је виртуелни сервер [10]. Виртуелни сервер је логички сервер који се „извршава“ на физичком серверу захваљујући техници виртуелизације. Виртуелни сервери су у потпуности транспарентни за кориснике, тј. њима изгледају као потпуно функционалне физичке машине које имају своје сопствене хардверске ресурсе. На истом физичком серверу се може извршавати већи број виртуелних сервера, чиме се постиже боља искоришћеност хардверских ресурса. Провајдери обично омогућавају

креирање „слике“ (енг. *image*) виртуелног сервера којим опслужују клијенте [10]. Свака слика има дефинисан број CPU-ова (*Central Processing Unit*) и језгара, брзину рада, количину оперативне меморије, оперативни систем итд. Креирање виртуелног сервера на основу слике је радња која се одвија аутоматски на захтјев, и у суштини, представља алоцирање ресурса стварних сервера и хардверске опреме за потребе креирања виртуелне инстанце сервера. Виртуелни сервери који дијеле ресурсе истог физичког сервера су у потпуности изоловани и независни.

Опслуживање клијената са виртуелним, а не стварним физичким серверима доноси бројне предности. Овакав приступ омогућава провајдерима да за опслуживање користе хардверске ресурсе који појединачно не одговарају клијентским захтјевима (пружају премало или превише ресурса) и то све на начин који је транспарентан за клијенте [10]. Виртуелизација омогућава дијељење хардверских ресурса између више корисника, што доводи до боље искоришћености ресурса, као и једноставније балансирање оптерећења хардверске инфраструктуре. Уколико се утврди оптерећеност неког хардверског чвора у *cloud* инфраструктури, лако се за потребе извршавања задатака могу додијелити додатни виртуелни ресурси истог типа који се извршавају на другим хардверским чворовима.

Виртуелизација ресурса *cloud* платформе се не врши у ситуацијама у којима су перформансе кључне како би се избјегло додатно оптерећење које доноси виртуелизација.

3.5 Употреба контејнера

Иако употреба виртуелизације у *cloud* окружењима доноси бројне предности, ипак виртуелне машине и даље не рјешавају у потпуности два проблема – скалабилност и миграцију сервиса [10]. Виртуелизација је омогућила једноставније и брже скалирање ресурса, али вријеме које је потребно за креирање и покретање нових виртуелних машина и даље није занемариво. Различите виртуелне машине представљају различита извршна окружења што онемогућава да се свака апликација без додатне конфигурације поуздано и конзистентно извршава на свим виртуелним машинама. Управо ови проблеми су ријешени употребом контејнера.

У области Cloud Computing-а, као и многим другим областима рачунарства, све је популарнија и прихваћенија употреба контејнера, као *lightweight* виртуелних машина. Раст популарности овог приступа је дјеломично изазван и развојем сервисно орјентисане, а нарочито микросервисне архитектуре софтвера.

Контејнери омогућавају креирање конзистентних окружења за развој, извршавање и испоруку апликација и њихових зависности. Управо креирање конзистентних окружења са свим зависностима рјешава проблем портабилности и омогућава једноставну миграцију апликација.

Често се *cloud* платформе које омогућавају управљање, оркестрацију и *hosting* контејнера у којима се извршавају апликације сврставају у нови Cloud Computing модел под називом “контејнери као сервиси” (енг. *Container as a Service* – CaaS), који се налази негдје између постојећих IaaS и PaaS модела (нешто ближе IaaS моделу јер се врши виртуелизација на нивоу оперативног система) [12].

Предности које доноси употреба контејнера [12]:

- Портабилност - апликација спакована у контејнер се може извршавати на било којој платформи, у било којем окружењу, јер контејнер већ садржи све што је потребно тој апликацији

- Скалабилност - једноставно се врши хоризонтално скалирање подизањем нових идентичних контејнера
- Ефикасност – захтијевају мање ресурса, не захтијевају сопствени *kernel*
- Сигурност – контејнери су изоловани између себе
- Брзина – подизање и уклањање контејнера је брза операција (рјешава проблем скалабилности)

3.6 Подјела Cloud Computing-а према власништву *cloud* платформе

Подјела се може извршити на основу власништва над *cloud* платформом и типу сервиса који се испоручују клијентима [10]. Подјела на основу власништва *cloud* платформе је дата у наставку.

3.6.1 Приватни (намјенски) cloud

Приватне *cloud* платформе су у власништву компанија или организација које их користе искључиво за своје потребе. Приступ овим платформама се обично остварује путем приватне мреже и, у складу са тим, скуп корисника је ограничен.

3.6.2 Јавни cloud

Јавне *cloud* платформе су у власништву компанија које се баве испоруком *cloud* сервиса (енг. *cloud provider*). Доступне су путем интернета свим корисницима који су вољни да плате услуге које користе. Корисници не знају са ким дијеле ресурсе нити како су они имплементирани. За разлику од приватних *cloud* платформи, овдје инфраструктуру одржавају провајдери услуга. Провајдери обезбјеђују сигурност и поузданост сервиса које пружају.

3.6.3 Хибридни cloud

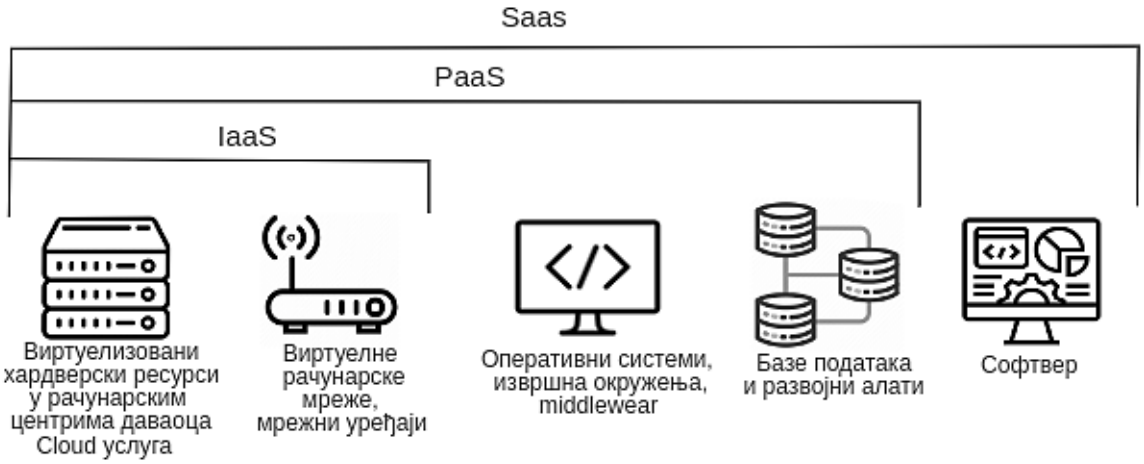
Хибридне *cloud* платформе су комбинација приватних и јавних платформи. У суштини један дио инфраструктуре *cloud* платформе је под контролом компаније/организације, а други дио чине сервиси који се изнајмљују од провајдера *cloud* услуга. Постоје два основна мотива за увођење овог типа платформе, неповјерење у сигурносне аспекте јавних платформи по питању складиштења тајних и сигурносно осјетљивих података и варијаблино оптерећење и потреба за ресурсима.

Сигурносно осјетљиви подаци се обично чувају на приватној платформи. У случају варијабилног оптерећења, дијелови система који имају варијабилно оптерећење могу се пребацити на јавни дио платформе што ће омогућити ефикаснију употребу ресурса. Изнајмљивање ресурса доводи до уштеде по питању куповине и одржавања опреме која ће се само повремено користити. Сервиси који имају константно оптерећење се обично налазе на приватном дијелу платформе.

У неким литература се помињу још и индустријске и заједничке (енг. *community*) *cloud* платформе [10].

3.7 Подјела Cloud Computing-а према типу сервиса

Клијенти, у зависности од типа сервиса који користе, остварују и одређени ниво контроле над одговарајућим аспектима *cloud* платформе. На слици 3.3 приказани су аспекти *cloud* платформе које клијент добија на коришћење код појединих модела испоруке сервиса.



Слика 3.3 – Модели испоруке сервиса

3.7.1 Инфраструктура као сервис (енг. Infrastructure as a Service – IaaS)

Код овог модела, провајдери *cloud* услуга клијентима на коришћење дају своју инфраструктуру. Под инфраструктуром се подразумијевају ресурси попут складишта података, виртуелних машина, *firewall*-а, *router*-а, *mail* сервера и др. Клијентима се обично изнајмљују виртуелизоване инстанце ових ресурса. Amazon[18] се сматра пиониром овог модела.

Клијент може да конфигурише инфраструктуру онако како му одговара и да је користи за шта год му треба. Суштина овог модела јесте *hosting* виртуелних рачунарских центара корисника у стварним физичким рачунарским центрима провајдера.

Клијент путем овог модела може да изнајми виртуелне машине (и контејнере), виртуелне мреже (енг. *Virtual Local Area Network* - VLAN), системе за похрану података (фајл, блок, објекат), слике оперативних система, *firewall*-е*, load balancer*-е, мрежну опрему итд.

Као илустративан примјер, може се узети проблем обезбјеђивања мјеста за становање. У том случају, IaaS модел одговара изнајмљивању земљишта на ће се уз помоћ сопствене опреме, алата и материјала саградити кућа.

3.7.2 Платформа као сервис (енг. Platform as a Service – PaaS)

PaaS је модел код којег се клијентима испоручују унапријед инсталирана и конфигурисана окружења која укључују оперативни систем, сервер базе података, *web* сервер, API за аутоматизацију рада, као и алат, подршку за радне оквире и програмске језике и извршно окружење, који омогућавају развој, тестирање, извршавање и *hosting* клијентских апликација. Изнајмљени ресурси се могу искористити за спровођење процеса континуалне испоруке и континуалне интеграције.

Овај тип услуге намијењен је тимовима и појединцима који раде на развоју софтвера. Велика предност PaaS-а је што омогућава развојним тимовима да се

[18] https://aws.amazon.com/

концентришу на развој и да не брину о потребној инфраструктури и опреми. PaaS обезбјеђује виши ниво апстракције од IaaS-а. Корисницима се заједно са развојним окружењем транспарентно испоручује и сва неопходна инфраструктура (хардверски ресурси, базе података и др.) која у овом случају није под њиховом контролом.

Софтвер се развија на удаљеној инфраструктури којој се приступа путем мреже и нема потребе за инсталацијом алата и потребног софтвера. Скалирање нижих нивоа се обавља транспарентно према потреби.

Коришћење PaaS модела одговара изнајмљивању плаца и неопходне опреме за самосталну градњу куће.

3.7.3 Софтвер као сервис (енг. Software as a Service – SaaS)

SaaS је модел код којег се клијентима испоручују готова софтверска рјешења. Овај тип услуге је намијењен клијентима који не желе да развијају своје апликације, него им је у реду да користе постојеће системе које обезбјеђују провајдери. SaaS се налази на врху хијерархије модела услуга и обезбјеђује највиши ниво апстракције.

Клијенти приступају готовим софтверским рјешењима путем *web* читачима и нема потребе за инсталацијом и конфигурацијом софтвера, нити имају икакве обавезе везане за одржавање сервиса и потребне инфраструктуре, исправку проблема у софтверу и сл. Провајдери су задужени за одржавање софтвера и свих ресурса који су потреби за његово неометано функционисање.

Системи који се нуде као услуга могу бити опште намјене попут Google Calendar[19] и Gmail, OneDrive[20], Microsoft 365, али могу бити и намјенска рјешења за компаније попут система за колаборацију, система за исплату доприноса и управљање људским ресурсима, система за односе са купцима (Salesforce.com и Sugar CRM[21]).

Употреба SaaS модела одговара изнајмљивању сграђене и опремљене куће.

3.7.4 Функција као сервис (енг. Function as a Service – FaaS) [13]

Још један назив за овај модел јесте Serverless (временом је термин *serverless* попримио шире значење и означава скуп архитектурних образаца). У питању је испорука функција, тј. малих функционалних компонената апликације које су прилично самосталне и изоловане од осталих. Назив *serverless* потиче од чињенице да се ови једноставни сервиси извршавају само по потреби и то на машинама које нису резервисане искључиво за њих. Функције су доступне у виду API-ја.

Ово је модел без стања (енг. *stateless*), базиран на догађај-вођеној архитектури (енг. *event-driven*). Апликације, односно функције се активирају појавом одређених дешавања, попут корисничких акција или системских догађаја. Када се појави захтјев, динамички се алоцирају сви неопходни ресурси и аутоматски се подиже функција која ће га обрадити. Уколико истовремено стигне више захтјева, платформа ће инстанцирати довољан број функција како би се обрадили сви захтјеви. FaaS концепт је алтернатива класичном приступу код којег се апликације у виду процеса стално извршавају на намјенским серверима. Једноставност функција, као и непостојање намјенске инфраструктуре за извршавање, диктира stateless природу модела. Апликације се обично извршавају у

[19] https://calendar.google.com/calendar
[20] https://onedrive.live.com/
[21] https://www.sugarcrm.com/

привременим контејнерима који живе све док пристижу захтјеви које одговарајућа функција треба да обради.

У суштини, то су функције које обично имају ограничено вријеме извршавања и које се извршавају у Linux контејнерима који су под контролом платформе провајдера.

FaaS је јако скалабилан јер се ресурси алоцирају тек по потреби. Погодан је за активности које су заказане, или се дешавају по неком правилу (нпр. генерисање извјештаја).

Ријетко се у пракси цијели системи могу имплементирати као скупови краткоживућих функција. Обично се системи састоје комбинације функција и дугоживућих микросервиса.

Примјери FaaS сервиса: IBM Cloud Functions[22], Amazon's AWS Lambda[23], Google Cloud Functions[24], Microsoft Azure Functions[25], OpenFaaS[26].

Неки од проблема на које се може наићи приликом употребе FaaS модела су: потребно више времена за активацију функција након периода неактивности (хладан старт), не постоји стандардизација која омогућава миграцију FaaS сервиса између провајдера (што повлачи зависност клијента од специфичне инфраструктуре и API-ја провајдера) и није одговарајући за дуготрајне послове и задатке,тј. за комплексну обраду.

Пратећи аналогију са мјестом становања, FaaS би представљао изнајмљивање било које куће у само одређеним тренуцима када је то потребно (нпр. вријеме ручка).

У називу наведених модела може се примијетити фраза “као сервисе”, која означава дио сервиса који клијент користи, а о којем се брине провајдер услуга.

3.8 Предности и мане

Cloud Computing је брзо постао јако прихваћен и популаран модел рачунарства управо због својих бројних предности.

Неке од предности које доноси употреба овог моделу су: скалабилност и еластичност (прилагођавање ангажованих ресурса тренутним потребама корисника), већа искоришћеност хардверских ресурса, смањење трошкова који се односе на куповину и одржавање намјенске опреме (хардвер и софтвер), омогућава међународно пословање јер су ресурси свугдје доступни [13], ширење интернет пословања без страха и улагања унапријед (омогућено једноставним скалирањем потребних ресурса и наплатом на основу коришћења), омогућава већу мобилност корисника, поузданост сервиса обезбијеђена од стране провајдера услуга и нема потребе за обезбјеђивањем додатног простора за потребну инфраструктуру - инфраструктура је на удаљеној локацији (енг. *off-site infrastructure*).

Употреба Cloud Computing модела доноси и одређене проблеме и мане: приватност и сигурност података, недоступност сервиса, зависност од интернет конекције, ограничена контрола инфраструктуре и ресурса који се изнајмљују, акумулација трошкова, миграција сервиса, варијабилне перформансе (посљедица ограничења мрежне инфраструктуре и

[22] https://cloud.ibm.com/functions/
[23] https://aws.amazon.com/lambda/
[24] https://cloud.google.com/functions
[25] https://azure.microsoft.com/en-us/products/functions
[26] https://www.openfaas.com/

чињенице да се ресурси дијеле између већег броја корисника) и немогућност довољног прилагођавања сервиса потребама корисника.

Складиштење и обрада приватних и сигурносно осјетљивих података на туђој инфраструктури, којој се још приступа путем јавне мреже, је проблематична. Провајдери услуга би се требали побринути за сигурност података и сервиса али увијек постоји доза сумње.

Сервиси могу бити недоступни због проблема који нису изазвани од стране клијента и над којим они немају контролу (нпр. нестанак струје на локацији рачунарског центра провајдера, природне катастрофе, проблеми у мрежној инфраструктури, одражавање инфраструктуре).

Иако се плаћање врши на основу употребе сервиса, трошкови се могу нагомилати и премашити средства која су потребна за куповину и одржавање намјенске инфраструктуре.

Нема потребне стандардизације између различитих *cloud* провајдера, што често доводи до тога да клијенти постају везани за одређеног специфичног провајдера услуга и не могу извршити миграцију својих сервиса.

4. FOG COMPUTING

Недостатак ресурса за обраду и складиштење прикупљених података, класичних IoT уређаја, је подстакао употребу *cloud* сервиса. Постало је уобичајено да IoT системи имају архитектуру која се ослања на *cloud* сервисе (енг. *Cloud-centric Internet of Things* - CIoT) [14].

Иако *cloud* сервиси константно добијају на популарности и доносе бројне предности, изнесене у претходном поглављу, њихова природа не одговара у потпуности потребама географски дистрибуираних IoT система. Као један од проблема који се нашао пред интеграцијом IoT и *cloud* технологија, тј. пред CIoT архитектуром, а постаје све актуелнији константним и убрзаним повећавањем броја умрежених IoT уређаја, јесте недовољан пропусни опсег мрежне инфраструктуре за пренос свих прикупљених података од IoT уређаја до *cloud* платформи. Ово повлачи закључак да ће ускоро пренос свих прикупљених података у сировом формату, заједно са њиховом комплексности, убрзо превазићи могућности мрежне инфраструктуре [15], те ће овакав приступ постати неодржив и немогућ. Поред тога континуалан пренос велике количине прикупљених података путем интернет инфраструктуре изискује знатна новчана средства. *Cloud* сервиси, због своје физичке локације и начина приступа који уноси велико кашњење, нису оптимални ни за обраду захтјева пристиглих од апликација које раде у реалном времену и код којих је вријеме реакције критично. Због свега наведеног појавила се потреба за географском дистрибуцијом рачунарских ресурса, који су до тада били централизовани (*cloud* платформе), по непосредној околини IoT уређаја. Циљ је био дистрибуираном, парцијалном и локалном обрадом прикупљених података, која ће представљати иницијалну обраду, која претходи слању података на *cloud* сервисе, надомјестити споменуте недостатке *cloud* платформи [16]. Као одговор на ове проблеме, дошло је до развоја Fog Computing модела који је требао да задржи предности Cloud Computing-а, а истовремено надомјестити недостатке, буде оптималан за потребе IoT екосистема и попуни технолошки јаз [17].

Уколико посматрамо "паметни" сензор температуре уочићемо да он врши мјерење отприлике сваке секунде и да те податке шаље на даљу обраду ка удаљеним *cloud* сервисима. Често се временски блиска мјерења неће много разликовати, те неће ни бити од великог интересе, већ ће само загушивати саобраћај. Много елегантније и ефикасније би било вршити локалну обраду резултата мјерења и само након детектовања значајних флуктуација температуре слати податке на детаљнију обраду ка удаљеним *cloud* сервисима. Управо овај приступ преставља модел Fog Computing-а.

Cisco[27] је 2012. године представио термин Fog Computing како би описао "алтернативу" Cloud Computing моделу [18].

Универзитет Princeton[28] у сарадњи са лидерима у индустрији (Cisco, ARM[29], Dell[30], Intell[31], Microsoft) 2015. године формирао организацију под називом OpenFog Consortium[32] који ће радити на развоју и промовисању Fog Computing модела.

[27] [27]https://www.cisco.com
[28]https://www.princeton.edu
[29]https://www.arm.com
[30]https://www.dell.com
[31]https://www.intel.com
[32]https://opcfoundation.org/

OpenFog Consortium је предложио референту архитектуру Fog Computing модела рачунарстава и дао јој назив OpenFog Architecture.

Термини *fogging* и *fog networking* су синоними за Fog Computing. Често се за опис овог модела погрешно користе други термини који представљају различите, али донекле сличне технологије – Cloudlets, Edge Computing, Mist Computing [19].

Fog Computing модел рачунарства је јако флексибилан и нема јасно дефинисане границе, па се зато некада другачије дефинише у различитим литературама. Према некима то је децентрализован модел рачунарства који проширује могућности Cloud Computing модела, приближавајући рачунарске ресурсе и сервисе границама мреже (крајњим корисницима који су овом случају IoT уређаји), како би боље пристајале потребама IoT екосистема [20]. Према OpenFog-у, Fog Computing је хоризонтална архитектура на системском нивоу која дистрибуира рачунарске ресурсе ближе крајњим корисницима у облак-ствар спрези (енг. *Cloud-to-Things continuum* – C2T) [19].

Симболика термина магла у називу овог модела, лежи у чињеници да је магла у ствари облак близак земљи, а Fog Computing је једноставна реплика *cloud* сервиса физички блиско лоцирана крајњим IoT уређаја.

Идеја Fog Computing-а јесте приближити моћ интелигенције, процесирања и складиштења података корисницима и мјесту гдје су потребни - граници саме мреже, тј. уређајима који прикупљају податке, са циљем да се IoT уређајима обезбиједи брз приступ локалним рачунарским ресурсима и сервисима. Флексибилни рачунарски ресурси су у Fog Computing моделу измјештени из „језгра“ интернета (*cloud* рачунарских центара) и приближени извору самих података, тако да се налазе негдје између *cloud* сервиса и IoT уређаја. Fog Computing треба да попуни празнину која постоји у C2T моделу [14].

Под границом мреже могу се подразумијевати мрежни уређаји попут *router*-а, *switch*-ева, *edge* сервера али и процесори у оквиру IoT уређаја. У суштини граница мреже је било који простор физички близак уређајима (за разлику од удаљених сервера и *cloud* сервиса) [15].

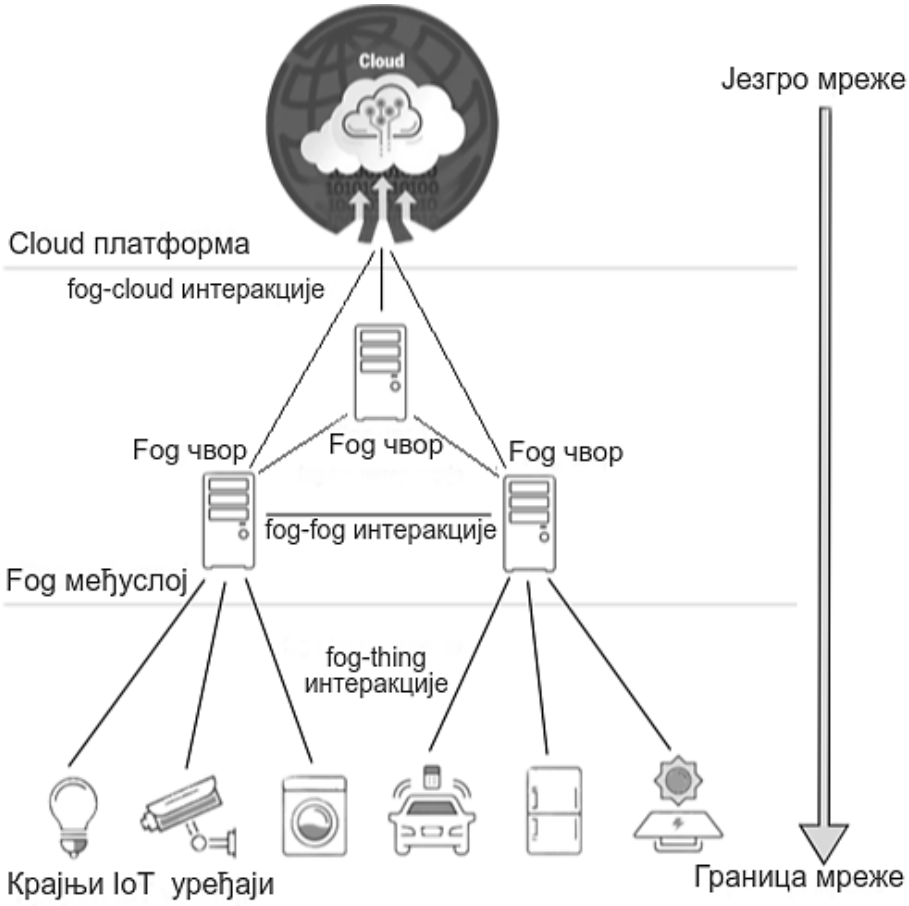
У суштини *fog* слој обезбјеђује рачунарске ресурсе и сервисе (ограничених могућности), доступне на *cloud*-у, локално, много ближе IoT уређајима [16], који ће вршити иницијалну и парцијалну обраду података и преузети дио оптерећења *cloud* сервиса.

Fog чворови су централни дио Fog Computing архитектуре. То су физички (намјенски сервери, gateway-и и др.) или виртуелни уређаји који су у спрези са крајњим IoT уређајима и који им нуде своје ресурсе и сервисе. У суштини сваки уређај који има способност складиштења и обраде података, те повезивања на мрежу може бити *fog* чвор. Иако су *fog* чворови инфериорни по питању ресурса у односу на виртуелне сервера на *cloud* платформама, њихова дистрибуираност и сарадња надомјештају тај недостатак. Како су они физички релативно близу изворима података, одлуке донесене на основу обраде прикупљених података много брже стижу на одредиште (ка одговарајућим IoT уређајима) [17]. Неке од основних функција *fog* чворова су: конверзија протокола, прикупљање података, иницијална обрада, привремено складиштење, филтрирање, компресија, нормализација, агрегација, кеширање [14] и прослеђивање података ка *cloud* сервисима.

Обрада података и извршавање задатака ближе крајњим уређајима, који су лоцирани блиско локалној рачунарској мреже, омогућава да се постигне минимално кашњење и минимално вријеме одзива система. Ово надаље омогућава имплементацију апликација за рад у реалном времену које имају јако ниску толеранцију на кашњење и код којих је вријеме одзива критично. Доношење закључака и одлука изведених из

прикупљених података, у што краћем року, омогућава да се правовремено ангажују одговарајући актуатори и обезбиједити реакцију система на измјену стања у околини.

Fog слој се може се посматрати као локална мала *cloud* платформа која има своје сопствене ресурсе, податке прикупља и добија од крајњих IoT уређаја и има доступно мноштво ресурса у стварном *cloud*-у. Fog међуслој у оквиру IoT екосистема је приказан на слици 4.1.



Слика 4.1 - Fog Computing модел у IoT екосистему

Иницијална обрада, агрегација, компресија и филтрирање релевантних прикупљених података смањује количину података који се преносе путем мреже, што повлачи рационалније коришћење доступног пропусног опсега.

Локална обрада и складиштење података смањују зависност рада IoT система од интернет инфраструктуре и омогућавају одржавање основних функционалности и за вријеме прекида приступа интернету.

Обрада прикупљених података на намјенским *fog* чворовима, омогућава да се дио оптерећења преузме са крајњих IoT уређаја и тако смањи њихова потрошња енергије.

Локална обрада сигурносно осјетљивих података, омогућава потпуну контролу на процесом обраде и складиштења података, а непостојање потребе за њиховим преносом ка удаљеним сервисима повећава сигурност података.

Локална *fog* инфраструктура омогућава да се имплементирају додатне сигурносне мјере како би се подигао степен сигурности читавог система.

Према неким изворима Fog Computing је блиско повезан са технологијом Cloudlet-a. Cloudlet-и су апликације које се извршавају на граничним уређајима у мрежи, са циљем

смањења кашњења у M2M (енг. *machine to machine*) комуникацији. Cloudlet-и имају способност конекције на интернет и доступни су за употребу од стране локалних уређаја, те се могу посматрати као *fog* чворови [19].

Уколико посматрамо приступне тачке које опслужују IoT уређаје у својој околини, њихова улога је да прослиједе пристигле пакете ка сервисима на интернету. Могуће је обезбиједити већи број приступних тачака како би се обезбиједила стабилнија и поузданија конекција. Уколико би се тим уређајима придодале функционалности обраде и складиштења података, те омогућила међусобна интеракција, они би формирали један *fog* слој састављен од мреже паметних IoT gateway-а [21].

4.1 Fog Computing у IoT екосистему

Fog Computing се имплементира у виду међуслоја који је посредник између крајњих IoT уређаја и *cloud* сервиса. Основна функција овог међуслоја је прикупљање, парцијална обрада и филтрирање IoT података. Релевантни процесирани подаци се прослеђују на даљу обраду и трајно складиштење ка *cloud* сервисима. OpenFog референтна архитектура прописује *fog*-*cloud*, *fog*-*fog* и *fog*-*thing* интеракције на овом слоју.

Комуникација између *fog* чворова и IoT уређаја са једне стране омогућава прикупљање података, а, са друге стране, пружање сервиса IoT уређајима.

Fog Computing уводи нови слој апстракције у IoT екосистем који је транспарентан за кориснике и крајње уређаје. Штавише, корисници/уређаји не морају бити уопште свјесни гдје се подаци шаљу, тј. разлика између *fog* и *cloud* сервиса је готово невидљива [16].

Увођење новог слоја који обезбјеђује сервисе у локалу, доводи до дилеме везане за расподјелу задатака и између *fog* слоја и *cloud* платформе. Битно је нагласити да не постоји стриктно правило према које се врши расподјела задужења, али неко оквирно правило гласи да *fog* слој врши привремено складиштење, првобитну, парцијалну обраду и филтрирање података, док је *cloud* платформа задужена за дубљу анализу релевантних података, извлачење закључака и трајно складиштење података. Неке функционалности су по својој природи боље одговарају cloud платформи, а друге *fog* чворовима. Сегментација задатака између *fog* и *cloud* слоја може бити статичка (унапријед дефинисана) али и динамичка, код које расподјела задатака зависи од пропусног опсега, оптерећења *cloud* платформе и *fog* чворова, отказа, сигурносних пријетњи и планираних оперативних трошкова [22].

Сам *fog* слој може имати дистрибуирану и централизовану архитектуру. Код централизоване архитектуре комуникација се остварује кроз централни чвор који уједно надзире и управља осталим чворовима. Код дистрибуиране архитектуре не постоји централни чвор. Сви *fog* чворови су равноправни и комуницирају по принципу P2P (*point to point*) како би сарађивали на извршавању послова, договорили око дистрибуције послова, размјене података, креирали резервне копије података, итд [16]. Овај принцип комуникације се назива *fog*-to-*fog* комуникација и омогућава да се превазиђу ограничености чворова по питању ресурса [16]. У односу на *cloud*, *fog* чворови располажу са немјерљиво мање ресурса.

Архитектура *fog* слоја може бити и слојевита, тако да се на сваком слоју налазе чворови који имају специфичне задатке везане за обраду података. Број слојева зависи од случаја употребе.

Као примјер се може узети трослојна архитектура *fog* слоја. Чворови у најнижем слоју обично имају задатак прикупљања података од IoT уређаја (*fog-thing* интеракције), њихову нормализацију и конверзију у други формат, по потреби. На сљедећем слоју се одвија филтрирање и процесирање података у реалном времену, као и доношење одлука. На највишем слоју врши се извлачење неких дубљих закључака и знања из података, њихова агрегација и слање ка *cloud* сервисима (*fog-cloud* интеракције) [22]. Највиши слој ни не мора да ради у реалном времену, тј. постоји већа толеранција за кашњење овог слоја.

Традиционални мрежни уређаји попут *router*-а, *switch*-ева, приступних тачака и др. имплементацијом Fog Computing модела добијају нову улогу и постају *fog* чворови. Поред стандардне функције ови уређаји су проширени ресурсима и функционалностима које омогућавају прикупљање, обраду и складиштење релевантних података [16]. У суштини додавањем поменутих функционалности ови уређаји постају "паметни". Поред ових уређаја постоје и намјенски *fog* уређаји (чворови) попут нано сервера, микро рачунарских центара и IoT gateway-а. *Fog* чворови формирају слој сервиса, сличан *cloud* платформи, али лоциран много ближе крајњим уређајима и корисницима [14], познат под називом *fog* међуслој. Према неким литературама постоје три типа *fog* чворова – *fog* уређаји, *fog* сервери и *fog gateway*-и [18].

Fog чворови могу да имају различита задужења, попут: кеширања, привременог складиштења, филтрирања, сакупљања, агрегације, нормализације, обраде, конверзије, енкрипције, компресије IoT података и комуникационих протокола.

Посебно су интересантни IoT *gateway* уређаји, чија је примарна улога прикупљање и просљеђивање IoT података ка *cloud* сервисима. Њихова функционалност се може проширити на разне начине како би попримили улогу *fog* чворова и, уједно, обрађивали прикупљене податке.

4.2 Fog Computing и Cloud Computing

Једна од идеја IoT концепта јесте аутоматизација процеса вођена одлукама донијетим на основу обраде и детекције дешавања у физичком свијету, у реалном времену. Управо ова идеја губи смисао у случају да читав процес траје предуго, тј. када систем уноси превелико кашњење [16].

Нови случајеви употребе IoT система, попут оних са критичним временом реакције и минималном толеранцијом на кашњење, излазе ван оквира могућности *cloud* сервиса. Потреба за што краћим временом између тренутка прикупљања података и реаговања на основу одлука заснованих на резултатима њихове обраде (вријеме одзива), није могла бити задовољена употребом *cloud* сервиса. Поред тога велика количина података које генеришу IoT уређаји сваке секунде доводи до загушења мрежног саобраћаја и посљедично до пада нивоа квалитета услуга. Разлог ових проблема није недостатак моћи процесирања *cloud* платформи, него њихова локација и начин приступ који је условљен ограниченим могућностима постојеће мрежне инфраструктуре и комуникационих технологија. Зато је идеја да се иницијална обрада и филтрирање података врши у оквиру локалне мреже IoT уређаја на специјализованим уређајима са већом процесорском моћи од класичних IoT уређаја, прије њиховог слања на даљу обраду *cloud* сервисима.

Недостаци употребе *cloud* сервиса за обраду IoT података су: непоуздан приступ сервисима путем интернета, непредвидиво и значајно кашњење, недостатак свјесности локације ресурса, загушење интернет саобраћаја великом количином сирових података, обрада потенцијално сигурносно осјетљивих података на удаљеној локацији,

незанемариви трошкови изнајмљивања платформе (нарочито током интензивне и континуалне употребе), велика везаност за провајдере *cloud* услуга, немогућност рада у ситуацијама у којима из неког разлога не постоји конекција ка интернету и постојање трошкова преноса велике количине података путем интернета.

Cloud сервиси јесу добро рјешење за обраду велике количине комплексних података (енг. *Big Data*), али не и за ситуације у којима је потребна обрада података у реалном времену.

Важно је напоменути да употреба Fog Computing приступа не представља замјену за постојећу употребу Cloud Computing концепта у IoT екосистему, већ његов комплемент, надопуну и адаптацију. Комбинација ова два приступа омогућава да се у потпуности задовоље потребе IoT екосистема. Ово значи да се и даље обрада и складиштење IoT података врши путем *cloud* сервиса, само се сада не шаљу сирови, нефилтрирани и неагрегирани подаци, него се иницијална обрада обави у *fog* слоју, тј. на дистрибуираним *fog* чворовима, а онда се само релевантни подаци прослиједе на детаљнију и дубљу анализу и складиштење. Складиштење података у *fog* слоју може бити само привремено због ограничених ресурса *fog* чворова.

Cloud Computing и даље има бројне карактеристике које су потребне IoT екосистему, попут: велике флексибилности, једноставног скалирања, поузданог складиштења и велике моћи обраде података (омогућава добијање дубљих увида и стицање нових знања на основу прикупљених података). Само се, примјеном и Fog Computing принципа, мијења улога *cloud* сервиса, који су сада задужени за трајно складиштење и обраду релевантних IoT података на вишем нивоу.

Сличности ова два модела се огледају у идеји да *fog* слој треба да обезбиједи поједностављене сервисе налик *cloud* сервисима, који задржавају све предности Cloud Computing модела, само се сада налазе локално, у непосредној близини IoT уређаја. Fog Computing је флексибилне и прилагодљиве природе која одговара природи *cloud* сервиса, па би требало да подржава виртуелизацију, *pool*-ове ресурса и употребу контејнера [22].

Разлика између ова два модела јесте у физичкој локацији рачунарских ресурса и сервиса. Код Fog Computing модела ресурси и сервиси су смјештени у близини или чак унутар саме локалне мреже IoT уређаја, док су код Cloud Computing модела смјештени „негдје на интернету“, тј. у неком удаљеном рачунарском центру. Разлика постоји и у самој инфраструктури. Потребна инфраструктура *fog* слоја је обично под контролом власника и самих IoT уређаја и система, док је *cloud* инфраструктура обично под контролом провајдера услуга. Иако се приступ ресурсима код оба модела врши путем мреже, много је стабилнији и поузданији приступ ресурсима *fog* слоја јер се до њих долази посредством локалне рачунарске мреже, те се тако избјегава комуникација путем интернета која подразумијева велики број посредника и међутачака у комуникацији. Важно је напоменути и да су *fog* чворови инфериорни по питању рачунарских ресурса у односу на *cloud* платформе и њихове виртуалне сервери, који имају доступно готово бесконачно ресурса.

Главна разлика између Fog Computing-а и Cloud Computing-а је у пружању сервиса у реалном времену и сервисима који могу да обраде податке, али са значајним нивоом кашњење [16].

4.3 Fog Computing и Edge Computing

Идеја приближавања рачунарских ресурса и дистрибуције обраде прикупљених података ближе извору података није нешто ново што се појавило са Fog Computing моделом. Овакав приступ обради података представља основну идеју и у нешто старијем концепту под називом Edge Computing, који се појавио 2004. године [14].

Често се термини Fog Computing и Edge Computing мијешају и користе како би се описала локална обрада IoT података. Иако је то на неком високом нивоу апстракције тачно, ипак између модела које описују ови појмови, постоји суштинска разлика која ће бити изнијета у наставку.

Сличност ова два концепта је очигледна, и огледа се у помијерању обраде прикупљених података у блиску околину крајњих IoT уређаја. Локална обрада прикупљених података на намјенској инфраструктури је циљ код оба модела.

Разлика између Fog Computing и Edge Computing модела није у идеји, нити у циљу, него о томе гдје се помијерају рачунарски ресурси. Код Fog Computing-а рачунарски ресурси за обраду података су сконцентрисани у дистрибуираним и умреженим *fog* чворовима који се налазе унутар локалне рачунарске мреже, гдје су смјештени и IoT уређаји који користе њихове сервисе. Код Edge Computinga врши се екстремна миграција рачунарских ресурса на саме крајње уређаје или уграђене PAC чипове (енг. *Programmable Automation Controllers*) [19]. Таква екстремна миграција рачунарских ресурса резултује тиме да се подаци обраде у оквиру самих уређаја који су их и генерисали/прикупили. Један од примјера имплементације Edge Computing принципа јесте камера којој је уграђена способност детекције објеката и која након детектовања објекта огласи неки сигнал.

Edge Computing је првенствено усмјерен на локацију ресурса и сервиса, а Fog Computing ставља нагласак и на њихову дистрибуцију у близини IoT уређаја [19]. Такође, Fog Computing предлаже слојевиту архитектуру и интензивну интеракцију између *fog* чворова који су дио исте рачунарске мреже (*fog-fog* интеракције), тј. неког IoT система, и у којој формирају *fog* међуслој. За разлику од тога, Edge Computing уопште не узима у обзир комуникацију између својих чворова који раде засебно и изоловано, те само уколико је потребно комуникацију остварују посредством неког екстерног систем, нпр. путем *cloud* сервиса.

Edge Computing не инсистира, или чак и не узима у обзир спрегу и сарадњу са *cloud* сервисима, док је Fog Computing намјенски креиран за сарадњу са *cloud* слојем надомјештањем његових недостатака.

Према неким литературама, није могуће успоставити и управљати *fog* слојем без интеграције Edge Computing технологија, па се стога некада користи термин који описује интеграцију ове двије технологије – FEC (енг. *Fog and Edge Computing*) [14].

У литератури се помиње и концепт Mist Computing-а, који се често мијеша са сродним концептима Fog и Edge Computing-а, а, у ствари, је подскуп Fog Computing модела који је усмјерен ка екстремном измјештању рачунарских ресурса ка крајњим уређајима (слично уграђеним сервисима) попут сензора и актуатора, све са циљем максималног смањења кашњења у комуникацији [14].

У суштини циљ Fog, Edge и Mist Computing-а је исти, а то је рјешавање проблема кашњења, непоузданог приступа сервисима на интернету, недостатка пропусног опсега и ниског нивоа QoS-а.

4.4 Примјери употребе

У наставку ће бити дати неки примјери употребе IoT рјешења код којих је употреба *fog* међуслоја доноси бројне предности у односу на директно слање података на обраду *cloud* сервисима.

4.4.1 Аутономна возила

Концепт Fog Computing-а, због блискости уређаја уграђених у аутомобиле и путну инфраструктуру, омогућава имплементацију идеалне архитектуре IoT система за аутономну вожњу, који подразумијева интензивну обраду велике количине прикупљених података, доношење одлука изведених из прикупљених података (енг. *data-diven*) и комуникацију између аутономних возила у реалном времену. Fog слој чини интеракцију у реалном времену између аутомобила, саобраћајне сигнализације и приступних тачака могућом, и истовремено сигурнијом и ефикаснијом. Вријеме реакције је у овој ситуацији кључно, а кашњење од само дјелића секунда при брзина већим од 80 km/h може бити фатално [19].

Аутономна возила се на високом нивоу апстракције, могу посматрати као покретни *fog* чворови који међусобно комуницирају [22], иако се у суштини састоје из већег број *fog* и других IoT уређаја.

Обрада података прикупљених за вријеме аутономне вожње ће постати неодржив, јер се предвиђа се да ће аутономна возила генерисати око 1GB података сваке секунде [23].

4.4.2 Системи за управљање саобраћајном сигнализацијом

Употребом *fog* слоја могуће је правовремено детектовати и реаговати на ротациона свјетла и сирену, те унапријед прилагодити саобраћајну сигнализацију, како би се омогућио несметан и што бржи пролазак возила под ротацијом.

4.4.3 Детекција лица/објеката на видео снимцима

Камере које имају способност кориштења локалних рачунарских ресурса за иницијалну детекцију лица или објеката на видео снимцима, или издвајање дијелова видео записа на којима су детектоване одређене флуктуације које би могле сигнализирати појаву објеката на слици, увелико смањују количину видео снимака (који су и иначе комплексне структуре и заузимају много простора) који се преносе путем интернета ка *cloud* сервисима на детаљну обраду.

Према неким истраживањима, могуће је постићи смањење времена детекције са 900ms на 196ms употребом *fog* слоја умјесто *cloud* сервиса [23].

4.4.4 Медицински уређаји

Употреба *fog* слоја може увелико допринијети смањењу критичног времена реаговања на промјену стања виталних параметара пацијента, као и повећању нивоа сигурности прикупљених података о пацијентима.

4.5 Предности и мане

Предности употребе *fog* међуслоја у IoT екосистему су:

- Смањење кашњења између захтјева и одговора на захтјев - вријеме које протекне од тренутка прикупљања података до тренутка генерисања резултата њихове обраде је драстично смањено, јер су рачунарски ресурси многе ближе изворима података. Подаци не морају ни да излазе на јавну мрежу која има комплексну архитектуру и велики број чворова посредника. Пренос података унутар локалне мреже је добар за перформансе, јер је могуће одабрати одговарајућу технологију умрежавања која ће обезбиједити постизање жељених перформанси. Смањење удаљености које подаци морају да пређу је нарочито осјетно у ситуацијама када уређаји очекују повратну информацију на основу које ће реаговати.
- Смањење количине података који се преносе путем интернета - филтрирање, агрегација, кеширање, компресија и слање само релевантних података на *fog* слоју може довести до знатног смањења саобраћаја ка *cloud* сервисима, што посљедично доводи до смањења загушења саобраћаја на интернету и рационалнијег кориштења пропусног опсега.
- Већа контрола над подацима и инфраструктуром - подаци се обрађују у локалу, на намјенској инфраструктури која је под контролом компаније која посједује и IoT уређаје.
- Подизање нивоа доступности - приступ *fog* сервисима се остварује посредством локалне мреже, која је под контролом власника IoT система, па је мања вјероватноћа да ће доћи до квара који ће онемогућити приступ ресурсима и који не може бити отклоњен у разумном року.
- Подизање нивоа сигурности - у оквиру *fog* слоја могу се имплементирати различити сигурносни механизми који ће допринијети сигурности IoT уређаја и података. Имплементација криптографских техника на овом нивоу може омогућити сигуран пренос података ка удаљеним *cloud* сервисима.
- Мања зависност од интернета и подизање нивоа поузданости - IoT системи увелико зависе од *cloud* сервиса, а самим ти и од конекције ка интернету. Нестабилна интернет конекција и недоступност удаљених сервиса у великој мјери ограничава ефикасност или чак у потпуности онемогућава рад IoT уређаја. Увођење *fog* међуслоја омогућава да IoT уређаји имају доступне барем основне сервисе и функционалности, које ће на себе преузети *fog* чворови у ситуацијама у којима је приступ интернету онемогућен. Такође, конфигурациони и подаци од значаја се могу чувати на *fog* чворовима, што представља вид резервне копије података која онемогућава да дође до губитка прикупљених података за вријеме недоступности удаљених сервиса.
- Смањење трошкова - мања количина података који се шаљу посредством интернета ка *cloud* сервисима смањује оперативне трошкове IoT система.
- Подизање нивоа квалитета услуга - смањење загушења интернет саобраћаја и кашњења повлачи и већи квалитет услуга. Рачунарски ресурси нису ближи само IoT уређајима него и крајњим корисницима IoT система па се побољшава и корисничко искуство [16].
- Повећана свјесност локације у односу на *cloud* сервисе

- Могућност рада без конекције ка интернету (енг. *offline*) - ако би се омогућио *offline* начина рада, *fog* чворови привремено чувају релевантне податке, процесиране информације, конфигурациона подешавања, алгоритме и логику за доношење одлука локално. У току *offline* начина рада IoT уређаји се опслужују искључиво резултатима добијеним радом локалних *fog* сервиса, а подаци се привремено складиште локално како би се накнадно могла урадити синхронизација са *cloud* сервисима и спријечио губитак података [24]. Код *offline* рада обично се врши приоритетизација задатака како би се обезбиједило да се критични задаци поуздано изврше. Ова особина *fog* слоја повећава и ниво отпорности читавог система на отказе.

Мане употребе Fog Computing модела су:

- Потенцијална гужва у мрежном саобраћају од IoT уређаја према *fog* чворовима
- Мања скалабилност у односу на *cloud*
- Повећана комплексност мреже, те управљање и одржавање *fog* чворова због дистрибуиране природе *fog* слоја и њихове хетерогености - нове технологије попут софтверски вођеног умрежавања, виртуелизације мрежних функција и *network slicing*-а омогућавају креирање флексибилних мрежних окружења која су једноставна за одржавање.
- Ограничени ресурси у односу на *cloud* платформе
- Неконзистентност података између *fog* чворова, као и између *fog* слоја и *cloud* платформе [24]
- Потребна намјенска инфраструктура која се мора одржавати, што повлачи одређене трошкови улагања
- Ограничен приступ ресурсима - могућ само из локалне мреже IoT уређаја
- Сигурност – потребно се бринути о сигурности великог броја *fog* уређаја
- Умрежавање – одржавање велике мреже хетерогених уређаја на рубу мреже IoT уређаја.

5. IOT GATEWAY

5.1 Network gateway уређаји

Мрежне капије (енг. *network gateway*) су намјенски мрежни уређаји или софтверске компоненте које служе као посредници у комуникацији између рачунарских мрежа, омогућавајући им размјену података. Један од њихових основних задатака јесте конверзија протокола.

Унутар рачунарских мрежа се за потребе интерне комуникације може користити један пакет мрежних протокола, док се за повезивање са другом мрежом може користити други пакет протокола. Како би се омогућила комуникација између разнородних технологија, потребно је вршити конверзију протокола података који се размјењују на граници између мрежа. Тако на примјер, мрежне капије омогућавају уређајима из једне локалне мреже, који имају приватне IP адресе, да приступе интернету и да њиховим посредством буду ограничено доступни путем интернета.

Мрежне капије се могу посматрати као тачке путем којих локална мрежа приступа интернету или комуницира са другом локалном мрежом.

Функционалност мрежних капија се некада проширује способностима попут превођења адреса, рутирања, дистрибуције оптерећења, енкрипције, проксирања, инспекције саобраћаја, логовања, контроле приступа, кеширања и др, чиме се добијају специјализоване мрежне капије (*Application Firewall*, *Application Gateway*, *VPN Gateway*, *VoIP Gateway*, *IoT Gateway* и др).

5.2 IoT gateway уређаји

На самом почетку развоја рачунарских мрежа, мрежне капије су биле намјенски уређаји чија је главна намјена била конверзија комуникационих протокола. Развојем и појавом *router*-а и *switch*-ева, мрежне капије нестају као самостални уређаји и њихова функционалност се уграђује у ове уређаје. До поновне употребе мрежних капија као самосталних уређаја дошло је усљед развоја IoT система и појаве проблема изазваних великим бројем умрежених и хетерогених уређаја, као и огромне количине интернет саобраћаја који генеришу [25].

IoT је мрежа великог броја хетерогених уређаја. У теорији, директно повезивање великог броја уређаја на *cloud* сервисе је могуће, али и јако непрактично. Поред непрактичности оваквог приступа, често IoT уређаји ни немају довољно ресурса и способности да самостално остваре такав вид комуникације. Недостатак ресурса је посљедица њихове бројности и трошкова производње, али и ограничености по питању потрошње електричне енергије (често се IoT уређаји напајају из сопствених батерија). Због ограничених ресурса, IoT уређаји су приморани на употребу једноставних комуникационих технологија које захтијевају мање ресурса, али и остварују ограничен степен сигурности података који се преносе, нарочито на веће удаљености путем јавне мреже.

Са друге стране, хетерогеност уређаја који се умрежавају, велики број протокола који се користе на апликативном и физичком слоју референтне OSI архитектуре, и недостатак стандарда готово онемогућавају директно повезивање крајњих уређаја на *cloud* сервисе посредством интернета и постојеће мрежне инфраструктуре.

Употреба намјенских IoT *gateway*-а који су лоцирани блиско крајњим уређајима, који имају довољно ресурса за комуникацију са удаљеним сервисима, који се обично

напајају директно из електричне мреже, који имају велики број различитих интерфејса и подржавају различите технологије за повезивање крајњих IoT уређаја, рјешавају наведене проблеме. Због свега наведеног дошло је до употребе IoT *gateway* уређаја као посредника у комуникацији између IoT уређаја и *cloud* сервиса. IoT *gateway*-и још више добијају на значају са порастом употребе *mash* мрежне топологије у IoT домену.

IoT *gateway* је хардверска или софтверска компонента која омогућава прикупљање података са крајњих IoT уређаја путем различитих комуникационих протокола и њихово прослеђивање путем интернета [26]. Подаци се даље прослеђују локалним рачунарским центрима или удаљеним *cloud* платформама. У IoT екосистему ови уређаји припадају *data-acquisition* слоју и представљају тачке акумулације и преноса свих прикупљених сензорских података.

У суштини, IoT *gateway* је паметни централни *hub* који омогућава успостављање комуникације између IoT уређаја (енг. *device-to-device*) као и њихове комуникације са *cloud* платформама путем интернета (енг. *device-to-cloud*), при чему се врши конверзија протокола и формата података [27]. Овакав начина рада је сличан раду *broker*-а.

IoT *gateway*-и морају бити способни да се носе са константим повећањем броја умрежених IoT уређаја [26] и да при томе очувају ниво квалитета услуга. Зато IoT *gateway*-и омогућавају једноставно скалирање како би се задовољиле потребе свих умрежених уређаја.

IoT *gateway* мора да садржи окружење које омогућава извршавање унапријед дефинисаних или намјенских корисничких софтвера за манипулацију подацима [27] и да подржава велики број комуникационих технологија, како би се задовољиле потребе IoT екосистема и омогућило повезивање различитих IoT уређаја [25] (ово их између осталог разликује од рутера). Зато је карактеристично да IoT *gateway* има мноштво различитих интерфејса. За повезивање са крајњим IoT уређајима се често користе технологије попут Bluetooth LE, Zigbee, Z-wave, LoRa, LTE, LTE-M и WiFi, док се за приступ интернету користи WiFi, 4G, 5G, Fiber Optics WAN (енг. *Wide Area Network*) и HDLC/PPP (High-level Data Link Control/Point to Point Protocol). Поред наведених технологија IoT *gateway*-и често морају да разумију и протоколе апликативног нивоа попут MQTT, CoAP, AMQP, DDS и др. На основу технологија које се користе за повезивање крајњих уређаја може се закључити да је доминантна употреба бежичних технологија.

Модерни IoT *gateway*-и омогућавају бидирекциону комуникацију између IoT уређаја и *cloud* сервиса. Бидирекциона комуникација омогућава слање прикупљених података ка удаљеним сервисима, али и слање инструкција/команди од удаљених сервиса ка крајњим IoT уређајима [25].

Стандардан циклус активности ових уређаја је: прикупљање података од крајњих уређаја попут сензора, конверзија протокола и прослеђивање података ка одговарајућим сервисима доступним путем интернета.

IoT *gateway*-и се могу се посматрати као „мостови“ између локалне мреже IoT уређаја и интернета, који врше апстракцију удаљених сервиса. Као једина приступна тачка ка умреженим IoT уређајима, ограничавају им приступ са јавне мреже и врше њихову изолацију, формирајући додатни сигурносни слој [28].

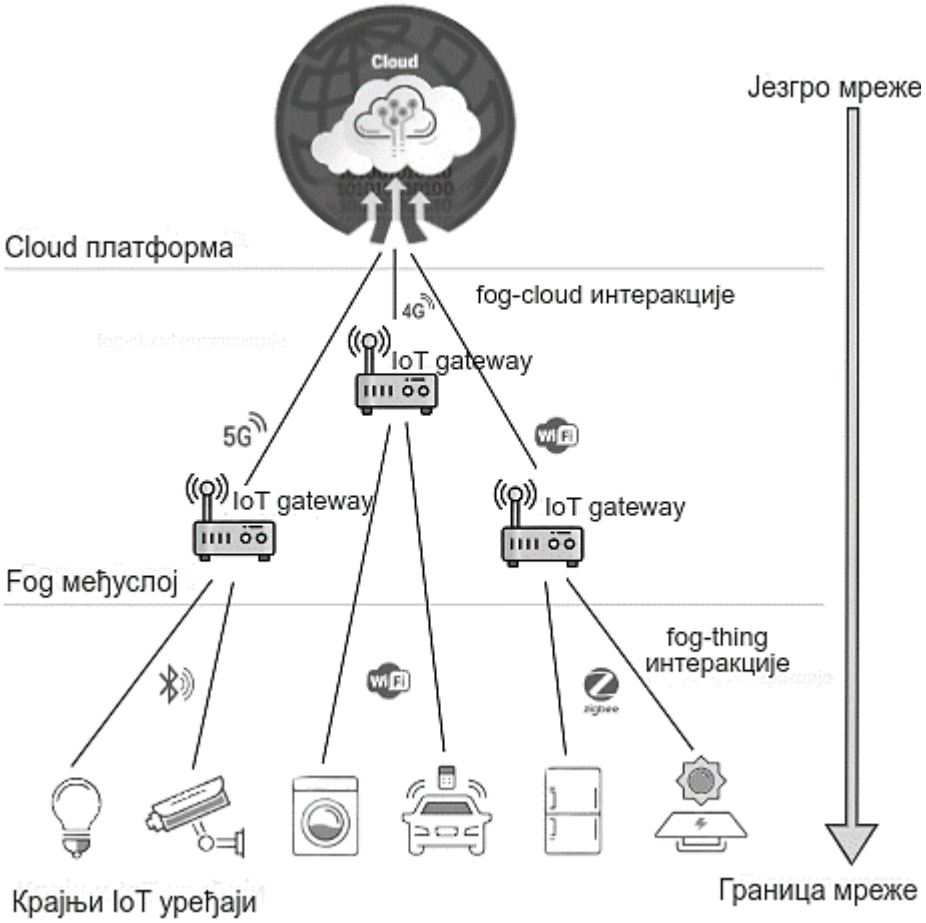
Иако, традиционално IoT *gateway*-и омогућавају да се подаци прикупљају и обезбјеђују да се разумију на њиховом одредишту, а *router*-и омогућавају да подаци стигну на одредиште, све је заступљенија интеграција функционалности ових уређаја, чиме настају IoT *router*-и [29].

У литератури се некада користи назив индустријски IoT *gateway* како би се означио IoT *gateway* прилагођен тежим условима рада, који садржи и индустријске типове интерфејса.

5.3 IoT gateway уређаји и Fog Computing

Као што је већ речено у претходном поглављу, *fog* слој је посреднички слој између крајњих IoT уређаја и интернета који врши локалну обраду прикупљених података. Иако је овај слој могуће, без икаквих проблема, успоставити без коришћења IoT *gateway* уређаја, употребом намјенских *fog* чворова, ови уређаји представљају врло погодно мјесто за имплементацију Fog Computing принципа. Разлог за то је што су IoT *gateway* уређаји већ постојећи уређаји у оквиру IoT екосистема који прикупљају сензорске податке које је уједно потребно и обрадити.

Традиционални IoT *gateway*-и су били ограничени по питању ресурса – имали су минимум ресурса који им омогућава прикупљање података и конверзију протокола. Обично нису имали могућности складиштења, и слабу или никакву моћ процесирања података [26]. Додавањем складишних и процесорских ресурса овим уређајима, те инсталацијом одговарајућег софтвера стварају се сви потребни услови за формирање *fog* чвора. IoT *gateway* уређаји у улози *fog* чворова су приказани на слици 5.1.



Слика 5.1 - IoT gateway-и у улози fog чворова

У већини случајева, није потребно слати све прикупљене податке на обраду ка удаљеним сервисима. Поред тога, много је случајева употребе код којих се слање прикупљених података путем интернета ка удаљеним сервисима и враћање инструкција крајњим IoT уређајима уноси превелико кашњење [26].

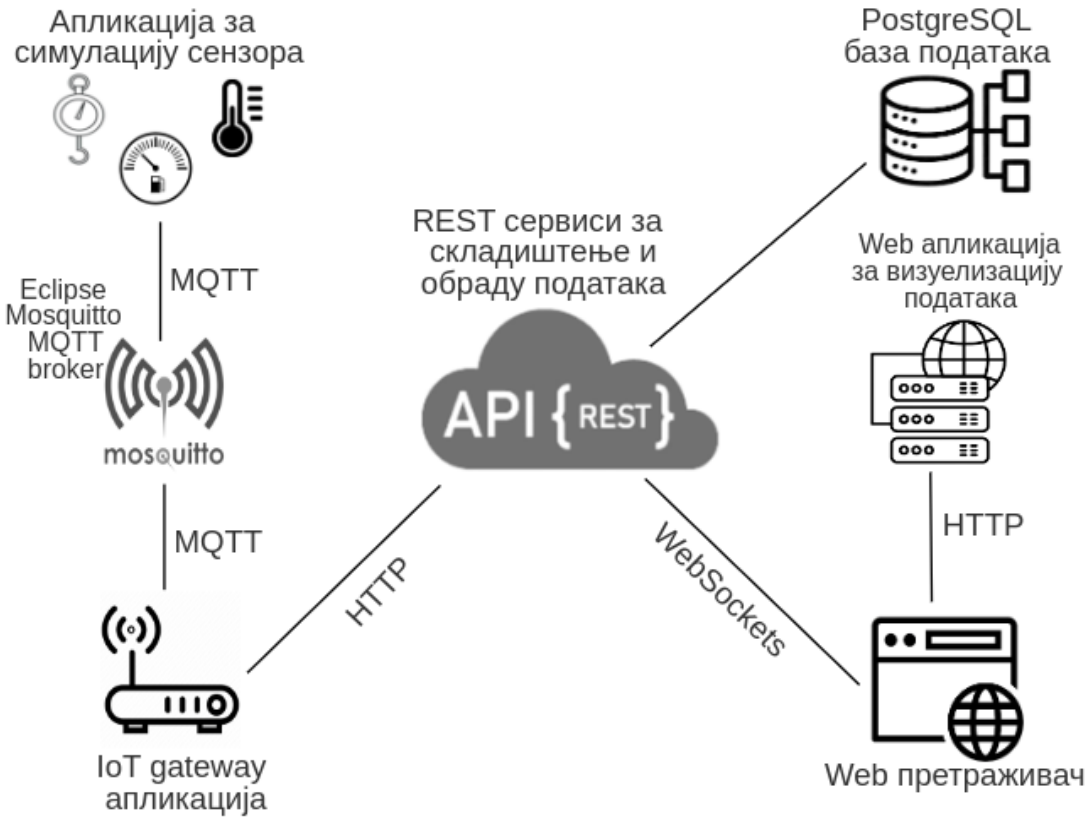
Имплементацијом принципа Fog Computing-а, IoT *gateway*-и добијају нове функционалности, као што су: подизање нивоа сигурности крајњих уређаја имплементацијом различитих сигурносних механизма, агрегација података, иницијална обрада података, филтрирање података, локално привремено складиштење података, кеширање и баферовање података.

Нове функционалости IoT *gateway* уређаја, који поред постојеће добијају и улогу *fog* чворова, стављају нагласак на софтвер који се извршава на овим уређајима. Специфични и намјенски случајеви употребе онемогућавају коришћење генеричких софтвера и стварају потребу за имплементацијом прилагођених софтверских пакета, чија комплексност расте са порастом функционалности *fog* чворова, које се желе имплементирати. Поред наведених, као још један од аргумената за развој намјенских софтвера за IoT *gateway* уређаје може се узети и потреба компанија да остваре што већу контролу над прикупљеним, потенцијално сигурносно осјетљивим подацима.

6. ПРАКТИЧНИ РАД

Задатак практичног дијела рада је реализација софтверског система који омогућава складиштење, обраду и визуелизацију прикупљених IoT података са радне машине, екскаватора. Централни дио система заузима апликација за IoT gateway уређај која омогућава прикупљање и агрегацију сензорских података, као и просљеђивање агрегираних података ка удаљеним сервисима.

Архитектура цјелокупног система је приказана на слици 6.1.



Слика 6.1 – Архитектура система

Систем се састоји од сљедећих шест основних цјелина: апликација за симулацију рада сензора, MQTT *broker*, IoT *gateway* апликација, REST сервиси за складиштење и финалну обраду података, PostrgreSQL база података и *web* апликација за визуелизацију прикупљених података.

6.1 Апликација за симулацију рада сензора

Основна намјена ове апликације јесте да симулира очитавања три врсте сензора. Симулира се рад сензора за температуру мотора, сензора за ниво горива у резервоару и сензора масе терета на оперативној руци машине.

Апликација је реализована као једноставна Python скрипта. Конфигурациони параметри апликације се чувају у JSON (*JavaScript Object Notation*) формату, у конфигурационом фајлу. Конфигурациони параметри спецификују: интервал очитавања сензора, опсег генерисаних вриједности, адресу MQTT *broker*-а и приступне податке. Имплементиран је и механизам *log*-овања грешака насталих у раду апликације. Спецификација *log* фајлова, стратегије *log*-овања, максималне величине *log* фајлова и времена њиховог чувања се налази у logging.conf фајлу.

Скрипта користи стандардне Python модуле: time за рад са временом, math за математичке операције, json за рад са JSON форматом и multiprocessing за рад са процесима. Поред стандардних модула користе се и paho.mqtt.client и numpy пакети који нису доступни у Python окружењу, већ их је потребно накнадно инсталирати.

Paho.mqtt.client[33] је python пакет који обезбјеђује клијента за MQTT протокол. Овај модул омогућава да се успостави конекција са MQTT *broker*-ом, да се пошаље порука и да се клијент претплати на спецификовани *topic*. У реализованом систему овај протокол је искоришћен за остваривање комуникације између сензора и IoT *gateway* апликације.

Numpy[34] пакет је намијењен за нумеричку математику и рачунарство у научне сврхе. Овај модул омогућава елегантан рад са низовима, матрицама и вишедимензионим низовима. Поред тога, омогућава генерисање псеудослучајних вриједности које прате одређену расподјелу случајних вриједности. У апликацији је искоришћен за добијање низа псеудослучајних вриједности које прате униформну расподјелу и које се користе за генерисање вриједности очитавања сензора.

Класа Process из multiprocessing модула омогућава креирање и покретање засебних процеса у оперативном систему који се могу извршавати истовремено.

6.1.1 Ток апликације

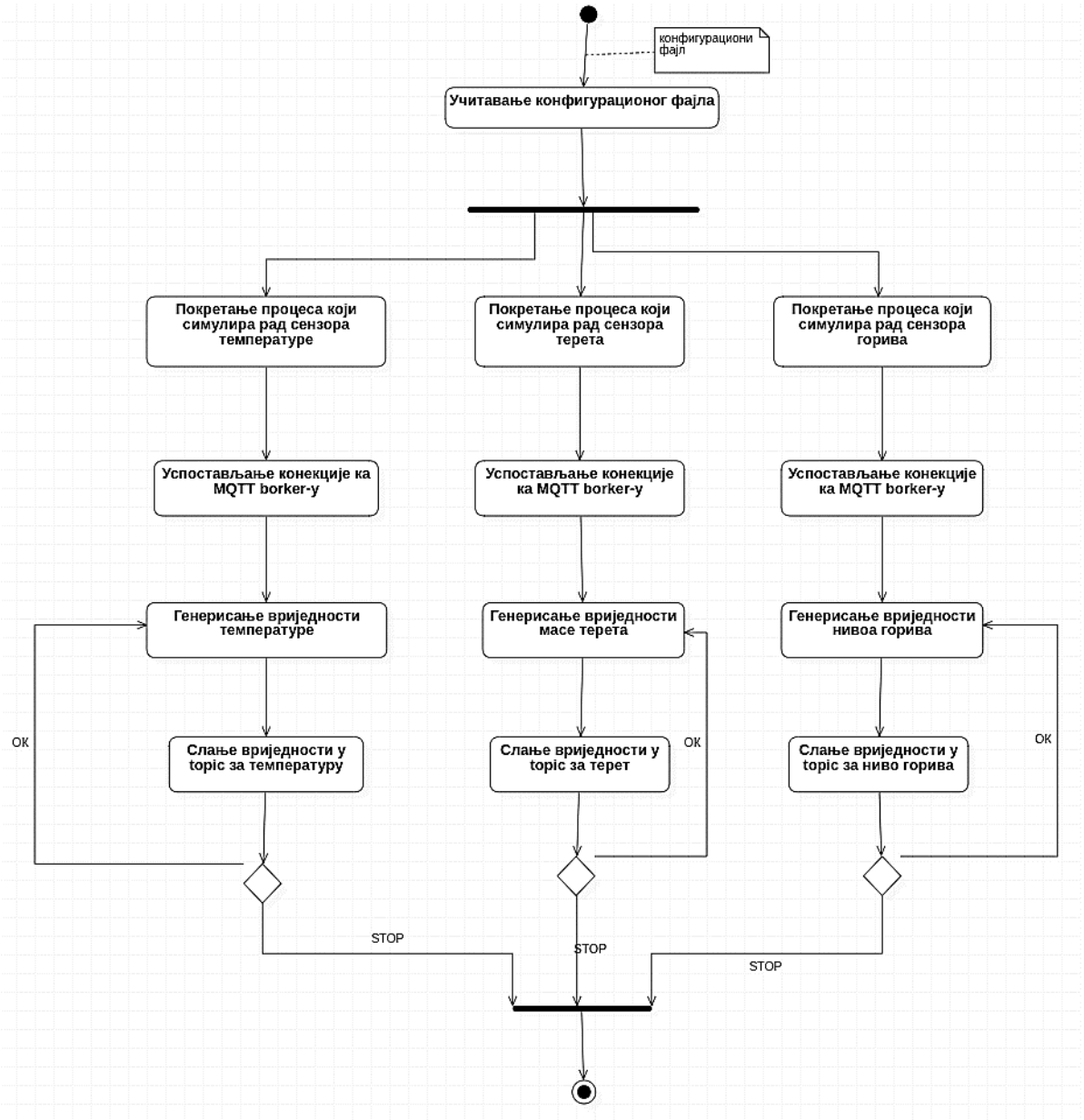
Након покретања апликације прво се учитава конфигурација из конфигурационог фајла, а затим се покреће по један процес за сваки тип сензора. Процеси прво успостављају везу са MQTT *broker*-ом на основу адресе и приступних података специфікованих у конфигурационом фајлу, а затим почиње периодично генерисање вриједности.

Процес, који симулира рад сензора за температуру, периодично генерише вриједност температуре мотора. Логика овог процеса је имплементирана тако да на основу случајних вриједности и просјечне температуре мотора, прво симулира постепено загријавање мотора, а, након тога, период флуктуације температуре око спецификоване просјечне вриједности, чиме се симулира радна температура мотора машине. На слици 6.2 приказан је основни ток апликације у форми дијаграма активности.

Симулација сензора оперативне руке машине је имплементирана тако да се са варијабилним периодом генерише псеудослучајна вриједност која представља масу терета на оперативној руци машине.

[33]https://pypi.org/project/paho-mqtt
[34]https://numpy.org

Ниво горива у резервоару машине се периодично генерише на основу потрошње горива по радном сату, вјероватноће допуњавања резервоара и ефикасности мотора.



Слика 6.2 – Ток активности сензорске апликације

Након што се генеришу вриједности сензорских очитавања, подаци се шаљу у одговарајући *topic* путем MQTT протокола.

6.2 MQTT broker

MQTT је једноставан протокол за размјену порука. Због своје једноставности, не доноси велико оптерећење уређајима који га користе, па из тог разлога представља идеално рјешење за успостављање комуникације између IoT уређаја. MQTT је протокол апликативног нивоа који у позадини користи TCP/IP протокол.

MQTT *broker* је софтверска компонента која служи као посредник у комуникацији употребом MQTT протокола. *Broker* рутира поруке од пошиљаоца ка свим заинтересованим примаоцима на основу *topic*-а и претплата.

Локално је подигнут MQTT *broker*. Кориштена је постојећа имплементација у Јава програмском језику – Eclipse Mosquitto[35]. *Broker* је конфигурисан тако да захтијева аутентикацију корисника и да обезбиједи да порука тачно једном стигне до примаоца.

Кориштење оваквог система за размјену порука омогућава да се смањи спрега између страна у комуникацији и да се омогући њихов асинхрони рад.

6.3 IoT gateway апликација

Централни дио реализованог система представља IoT *gateway* апликација. У питању је монолитна апликација имплементирана у програмском језику Python. Апликација, у суштини, представља клијента за REST базиране *web* сервисе на *cloud*-у.

Основни функционални захтјеви апликације су да омогући регистрацију нових умрежених IoT уређаја, аутентикацију постојећих, прикупљање сензорских података, агрегацију и периодично слање података прикупљених путем IoT уређаја у склопу екскаватора.

Један од циљева је био имплементирати принципе Fog Computing модела и самим тим учинити IoT *gateway* уређај уједно и *fog* чвором. Из тог разлога имплементирана је агрегација, филтрирање и сумаризација прикупљених података. Како би се измјерио утицај локалне обраде прикупљених података, имплементиран је механизам праћења статистике рада IoT *gateway* уређаја. Механизам омогућава да се за вријеме рада уређаја, биљежи количина прикупљених података, количина генерисаног мрежног саобраћаја и број упућених захтијева ка удаљеним сервисима.

Како би се омогућило праћење рада апликације конфигурисан је механизам за *log*-овање грешака у раду апликације. Конфигурација *log*-ера и стратегије *log*-овања се налази у logging.conf фајлу.

6.3.1 Кориштени модули

За имплементацију ове апликације кориштени су requests[36] и paho.mqtt.client екстерни Python пакети.

Paho.mqtt.client пакет је кориштен и у сензорској апликације. Као што је већ речено, овај модул представља клијента за MQTT протокол.

Пакет requests представља HTTP клијента. Овај модул је искориштен за упућивање HTTP захтјева ка удаљеним сервисима.

[35]https://mosquitto.org
[36]https://pypi.org/projects/requests

Употребом уграђеног logging.config модула извршена је конфигурација и прилагођавање *log*-ера.

Json пакет је кориштен за конверзије у/из JSON формата.

Помоћу multiprocessing модула креирани су засебни подпроцеси који ће прикупљати и обрађивати различите сензорске податке.

6.3.2 Структура апликације

Апликација се састоји од четири модула: app, auth, data_service, stats_service.

У auth модулу имплементиране су login и register функције. Ове функције омогућавају да се креденцијали уређаја пошаљу, употребом HTTP клијента из requests модула, на одговарајући REST сервис за пријаву, тј. регистрацију уређаја.

У stats_service модулу дефинисане су Stats и OverallStats класе. Stats класа представља статистичке податке везане за један тип сензорских података, а OverallStats класа представља статистичке податке рада IoT *gateway* уређаја у неком временском интервалу. У оквиру овог модула имплементирана је и функционалност слања статистике рада уређаја ка удаљеним сервисима. Приликом слања статистике, у заглавље HTTP захтјева укључи се и JWT (*JSON Web Token*) токен, добијен након успјешне пријаве/регистрације уређаја.

Data_service модул садржи логику за агрегацију, филтрирање и просљеђивање прикупљених података. Имплементиране су три функције, по једна за сваки тип сензора, са различитим стратегијама агрегације и филтрирања података. Подаци се ка удаљеним сервисима шаљу у JSON формату. Све грешке везане са локалну обраду података или проблеме са удаљеним сервисима се логују путем конфигурисаних *log*-ера. И овдје се приликом слања HTTP захтјева укључује JWT токен ради ауторизације и аутентикације на серверској страни.

App модул представља полазну тачку рада ове апликације. Овај модул користи auth модул за пријављивање уређаја на удаљене сервисе, stats_service модул за праћење статистике и data_service модул за рад са сензорским подацима. У оквиру имплементиране логике app модула, креирају се три *worker*-а који прикупљају и обрађују сензорске податке.

6.3.3 Основни ток активности апликације

На почетку рада апликације прво се учитава конфигурација из фајла. У конфигурационом фајлу су садржани сљедећи параметри: креденцијали уређаја, адреса сервиса, адреса MQTT *broker*-а, формат времена и интервали слања агрегираних података.

Након успјешног учитавања конфигурације, приступа се пријави уређаја на REST сервисе. Уколико пријава буде неуспјешна, периодично се захтијева регистрација уређаја.

Након што се успјешно обави пријава/регистрација уређаја и добије JWT, приступа се креирању три *worker*-а који ће прикупљати, агрегирати и филтрирати сензорске податке. *Worker*-и су засебни подпроцеси који се извршавају истовремено и имају дефинисан ток акција. Након покретања, сва три *worker*-а ће приступити успостављању конекције ка MQTT *broker*-u (за шта су потребни посебни креденцијали) и пријави на *topic* од интереса.

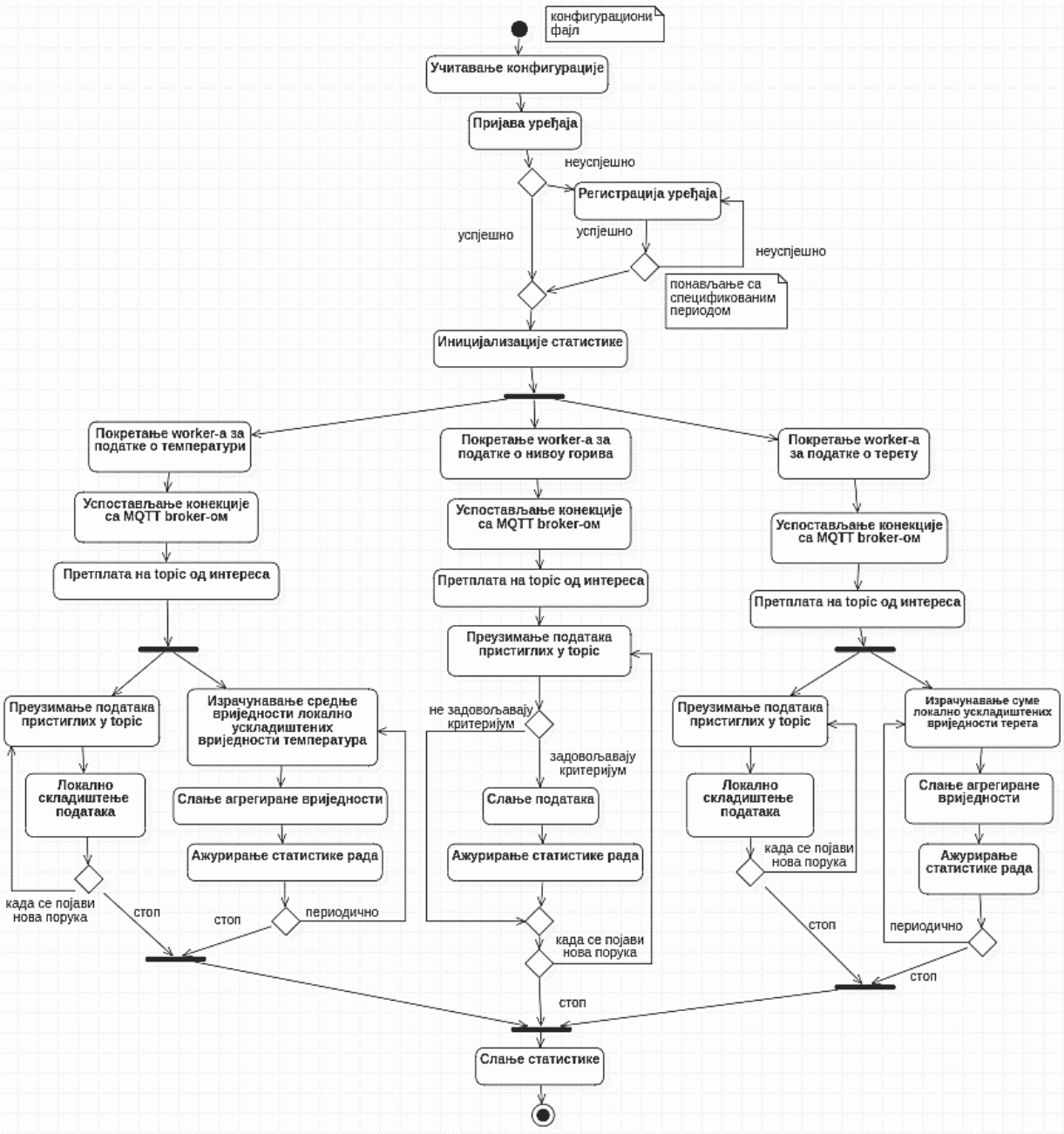
У наставку је дат дио кода којим се врши агрегација и обрада сензорских података.

```
def collect_temperature_data(interval,url,jwt,time,mqtt,port,user,pass,flag,queue):
  new_data = [] ; old_data = []
  def on_message_handler(client, userdata, message):
    if not flag.is_set():
      new_data.append(str(message.payload.decode("utf-8")))
      customLogger.info("Data: "+str(message.payload.decode("utf-8")))
  stats = stats_service.Stats()
  client = mqtt.Client(client_id="temp", transport=tcp, protocol=mqtt.MQTTv5)
  client.username_pw_set(username=user, password=pass)
  client.on_connect = on_connect_temp_handler ; client.on_message=on_message_handler
  while not client.is_connected():
    try:
      client.connect(mqtt,port=port, keepalive=abs(round(interval)) * 3)
      client.loop_start()
    except:
      errorLogger.error("Failed to establish connection with MQTT")
    time.sleep(0.2)
  while not flag.is_set():
    data=new_data.copy() ; new_data.clear()
    for i in old_data:
      data.append(i)
    old_data.clear()
    if len(data) > 0:
      code = data_service.handle_temperature_data(data, url, jwt, time)
      if code != http_ok:
        old_data = data.copy()
      else:
        stats.update_data(len(data) * 4, 4, 1)
      if code == http_unauthorized:
        customLogger.error("JWT has expired!") ; break
    time.sleep(interval)
  queue.put(stats) ; client.loop_stop() ; client.disconnect()
  customLogger.debug("Temperature data handler shutdown!")

def handle_temperature_data(data, url, jwt, time_format):
  data_sum = 0.0 ; unit=unknown
  for item in data:
    try:
      tokens=item.split(" ") ; data_sum += float(tokens[1].split("=")[1])
    except:
      customLogger.error("Invalid temperature data format! - "+item)
  time_value = time.strftime(time_format, time.localtime()) ; unit="unknown"
  try:
    unit = data[0].split(" ")[7].split("=")[1]
  except:
    errorLogger.error("Invalid temperature data format! - "+data[0])
  payload = {"value": round(data_sum / len(data),2),"time": time_value,"unit": unit}
  customLogger.warning("Forwarding temperature data: " + str(payload))
  try:
       auth="Bearer " + jwt
    post_req = requests.post(url,json=payload,headers={"Authorization": auth})
    if post_req.status_code != http_ok:
      customLogger.error("Service response:"+ str(post_req.status_code))
    return post_req.status_code
  except:
        customLogger.critical("Service unavailable!") ; return http_not_found
```

Worker-и локално складиште прикупљене податке и периодично врше њихову агрегацију и прослеђивање ка удаљеним сервисима. За разлику од преостала два *worker*-а, *worker* који обрађује податке везане за ниво горива у резервоару, не врши локално складиштење, већ филтрирање података. Само онда када се појави вриједност нивоа горива од интереса, подаци ће бити прослијеђени. *Worker*-и биљеже статистику свога рада.

На слици 6.3 приказан је основни ток апликације у форми дијаграма активности.



Слика 6.3 – Ток активности

Непосредно прије гашења уређаја врши се слање прикупљених статистичких података.

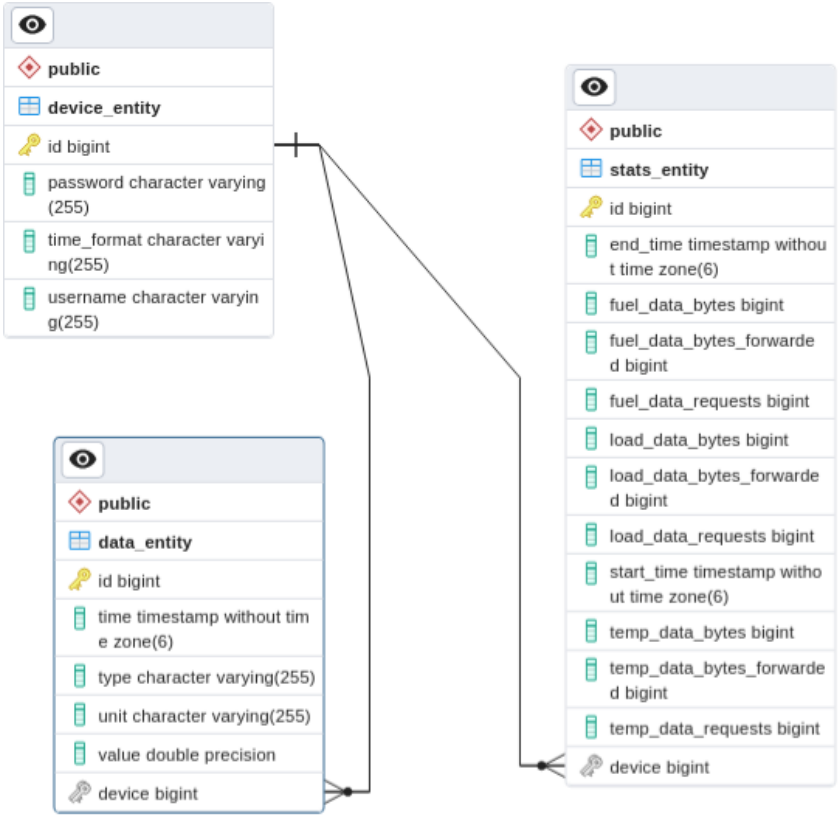
Уколико се у неком тренутку детектује да приступ REST сервисима није могућ због истеклог JWT-а, извршиће се поновна аутентикација уређаја.

6.4 PostgreSQL база података

За потребе трајног складиштења података подигнут је PostgreSQL сервер базе података.

Сервер је конфигурисан тако да се бази података, у којој се чувају прикупљени подаци, може приступити само са одговарајућим кредендцијалима, тј. креиран је посебан кориснички налог са ограниченим привилегијама за потребе приступа подацима од стране REST сервиса.

Шема базе података, са слике 6.4, је креирана на основу JPA ентитетских класа.



Слика 6.4 – Шема базе података

6.5 REST сервиси

За потребе складиштења, обраде и приступа подацима, креирана је серверска апликација у Java програмском језику, употребом Spring развојног оквира. Апликација излаже REST API који користе IoT *gateway* и *web* апликација.

Имплементирани REST сервиси омогућавају пријављивање уређаја, регистрацију новог уређаја, пријем агрегираних података, пријем статистике рада уређаја и добављање података генерисаних од стране специфікованог уређаја.

Spring развојни оквир се састоји од више међусобно независних модула. Основа Spring развојног оквира је "убризгавање зависности" (енг. *dependency injection)*. Зависности за неку класу представљају објекте других класа који су потребни за рад основне класе. Убризгавање зависности је пројектни образац који омогућава постизање инверзије контроле (енг. *inversion of control*). Инверзија контроле је техника код које објекти нису задужени за креирање својих зависности, него су им оне обезбијеђене из екстерног извора. Spring омогућава аутоматску иницијализацију зависности и контролисање њиховог

животног вијека. Такође, омогућено је креирање компоненти, тј. класа чији животни вијек контролише развојни оквир. Оне се могу користити као зависности у другим компонентама, и није их потребно експлицитно иницијализовати.

Поред имплементираних REST сервиса, омогућена је и комуникација путем WebSocket протокола. За те потребе искоришћен је Spring WebSocket модул. Овај модул омогућава једноставну спецификацију WebSocket API-ја, подизање MQ *broker*-а, и спецификацију базних *topic*-а. Креирани WebSocket API искоришћен је од стране *web* апликације за претплаћивање на *topic* одговарајућег уређаја. Путем *topic*-а *web* апликација у реалном времену добија актуелне IoT податке, одмах након њиховог пријема на серверској страни.

6.5.1 Коришћени Spring модули

Модули који су искоришћени приликом развоја сервиса су: Spring Data JPA, Spring Boot, Spring Security, Spring Web Socket и JSON Web Token.

Spring Boot омогућава креирање самосталних Spring апликација које су спремне за продукцију. Употребом Spring Boot аутоконфигурације избјегава се напредно конфигурисање апликације путем XML (*Extensible Markup Language*) фајлова и захтијева се минимална интервенција програмера. Spring Boot омогућава паковање апликације у JAR (*Java Archive*) заједно са уграђеним апликативним сервером као што је Tomcat[37]. Овакав начин паковања омогућава извршавање запаковане апликације у свим окружењима која имају подешену Java виртуелну машину, без потребе за инсталацијом и конфигурисањем посебног апликативног сервера.

Spring Data JPA модул омогућава једноставан и ефикасан рад са релационим базама података. JPA је спецификација програмског интерфејса намијењеног за управљање подацима унутар релационих база података путем Java програмског језика. Spring Data JPA омогућава креирање JPA репозиторијума – Spring компоненти које представљају апстракцију слоја за приступ подацима. Помоћу репозиторијума Java програм, употребом синтаксе на високом нивоу апстракције, може да приступа подацима у релационој бази података, не бринући се за имплементацију и повезивање на базу података.

Spring Security модул омогућава конфигурацију механизама аутентикације и ауторизације, као и других сигурносних аспеката апликације. Уз помоћ овог модула могуће је ограничити приступ изложеном API-ју.

JSON Web Token модул омогућава генерисање и дигитално потписивање JWT ауторизационих *token*-ена. У token се могу укључити произвољне информације попут корисничких имена, права приступа, и др.

6.5.2 Структура апликације

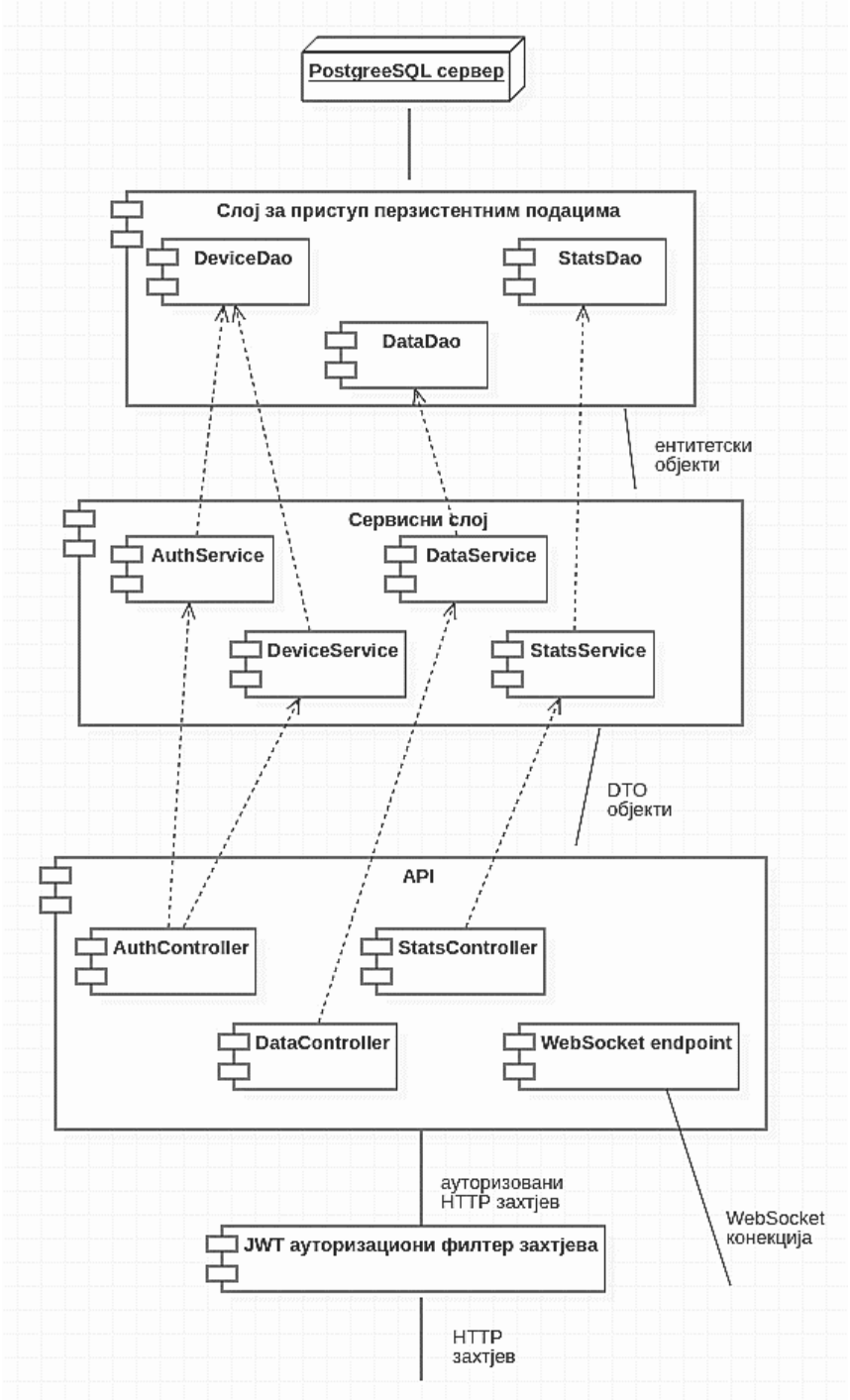
Серверска апликација има слојевиту архитектуру која се састоји од три основна слоја: API-ја, сервисног слоја и слоја за приступ подацима. Слој за приступ подацима користи екстерну PostgreSQL базу података за потребе перзистенције.

API серверске апликације је дефинисан у Controller класама controllers пакета. Controller класе су задужене за обраду пристиглих HTTP захтјева и враћање одговора. Обрада захтјева подразумијева преузимање релевантних података из захтјева и покретање пословне логике. API ове апликације чине REST сервиси који очекују HTTP захтјеве са

[37] https://tomcat.apache.org/

подацима у JSON формату. Спецификовани REST сервиси омогућавају пријаву, регистрацију уређаја, пријем и преузимање сензорских и статистичких података.

Изворни код апликације је подијељен у осам основних пакета: controllers, services, dao, model, dto, config, util и security. Архитектура апликације је приказана на слици 6.5.



Слика 6.5 – Архитектура серверске апликације

Поред REST сервиса, отворен је и један *endpoint* који ради са WebSocket протоколом. Конфигурација овог *endpoint*-а налази се у WebSocketConfig класи config пакета.

У наставку је дат дио кода који представља API за аутентикацију корисника.

```
@RestController
@RequestMapping("/auth")
public class AuthController {
  private final Base64.Decoder decoder = Base64.getDecoder();
  private final AuthService authService;
  @Value("${api.key}")
  private String apiKey;

  public AuthController(AuthService authService) {
    this.authService = authService;
  }

  @GetMapping("/login")
  public ResponseEntity<String> login(@RequestHeader(HttpHeaders.AUTHORIZATION)
      String auth){
    String[] tokens = auth.split(" ");
    byte[] data = tokens[1].getBytes();
    byte[] decodedData = decoder.decode(data);
    String credentials = new String(decodedData);
    tokens = credentials.split(":");
    String username = tokens[0];
    String password = tokens[1];
    String jwt = authService.login(username, password);
    if (jwt != null)
      return new ResponseEntity<>(jwt, HttpStatus.OK);
    else
      return new ResponseEntity<>(HttpStatus.UNAUTHORIZED);
  }

  @PostMapping("/signup")
  public ResponseEntity<String> register(@RequestHeader(HttpHeaders.AUTHORIZATION)
      String key, @RequestParam String username, @RequestParam  String password,
@RequestParam String time_format) {
    if (apiKey.equals(key)) {
      String jwt = authService.register(username, password, time_format);
      if (jwt != null)
        return new ResponseEntity<>(jwt, HttpStatus.OK);
    }
    return new ResponseEntity<>(HttpStatus.UNAUTHORIZED);
  }
}
```

Пословна логика серверске апликације смјештена је у сервисном слоју. Овом слоју припадају класе дефинисане у services пакету. Класе сервисног слоја имплементирају обраду пристиглих података. Контролери као своје зависности имају, управо, објекте одговарајућих класа овог слоја. Контролер када преузме податке из пристиглог захтјева позива логику сервисног слоја и враћа резултата њеног извршавања у HTTP одговору.

Dao (енг. *data access object*) пакет се састоји од класа које чине слој приступа подацима. Слој за приступ подацима омогућава манипулацију подацима у базама података. Како је кориштен JPA, није било потребе имплементирати овај слој, већ само декларисати по једну Repository класу за сваку ентитетску класу. Када је потребно да се у склопу пословне логике приступа перзистентним подацима, сервисне класе позивају логику својих зависности, тј. Repository класа.

У model пакету дефинисане су ентитетске класе апликације. Објекти ових класа се чувају у бази података и са њима манипулишу Repository класе.

DTO (*Data Transition Object*) класе су смјештене у dto пакет. Објекти ових класа пристижу у тијелу HTTP захтијева и користе се од стране сервисног слоја. Такође, као резултат обраде HTTP захтјева враћају се објекти ових класа.

У оквиру util пакета смјештене су класе које служе за конфигурацију *log*-ера и дефинисање стратегије *log*-овања грешака и пристиглих захтјева.

У наставку је дат дио кода којим се подешавају сигурносни механизми апликације.

```
@Override
protected void doFilterInternal(HttpServletRequest request, HttpServletResponse
	response, FilterChain filterChain) throws ServletException, IOException {
	String tokenHeader = request.getHeader("Authorization");
	if (tokenHeader != null && tokenHeader.startsWith("Bearer ")) {
		String token = tokenHeader.substring(7);
		try {
			String username = jwtUtil.getUsernameFromToken(token);
			if (username!=null &&
			SecurityContextHolder.getContext().getAuthentication()
			== null) {
				UserDetails userDetails =
			deviceService.loadUserByUsername(username);
				if (jwtUtil.validateToken(token, userDetails)) {
					UsernamePasswordAuthenticationToken
authenticationToken                                    = new
UsernamePasswordAuthenticationToken(
					userDetails, null, userDetails.getAuthorities());
					authenticationToken.setDetails(new
					WebAuthenticationDetailsSource()
					.buildDetails(request));
					SecurityContextHolder.getContext()
					.setAuthentication(authenticationToken);
				}
			}
		} catch(Exception e){
			e.printStackTrace();
		}
	}
	else
		System.out.println("Request does not contain JWT token!");
	filterChain.doFilter(request, response);
}

@Bean
protected SecurityFilterChain securityFilterChain(HttpSecurity http) throws
Exception {
	return http.addFilter(corsFilter()).csrf(csrf -> csrf.disable())
	.exceptionHandling(eh ->
	eh.authenticationEntryPoint(authenticationEntryPoint))
	.authorizeHttpRequests(ar ->
	ar.requestMatchers(HttpMethod.GET,"/auth/login").permitAll()
	.requestMatchers(HttpMethod.POST,"/auth/signup").permitAll()
	.requestMatchers(HttpMethod.GET,"/ws/**").permitAll()
	.anyRequest().authenticated())
	.sessionManagement(sess ->
	sess.sessionCreationPolicy(SessionCreationPolicy.STATELESS))
	.addFilterBefore(jwtAuthorizationFilter,
	UsernamePasswordAuthenticationFilter.class).build();

}
```

Сигурносни механизми апликације су конфигурисани у security пакету. Креиран је ауторизациони филтер помоћу JwtAuthorizationFilter класе, који пресреће пристигле захтјеве и провјерава да ли у *Authorization* заглављу садрже валидан JWT токен за приступ API-ју. У оквиру WebSecurityConfiguration класе смјештена је конфигурација сигурносних аспеката апликације. Додат је креирани филтер, спецификована је CORS (*Cross-Origin Resource Sharing*) политика и *endpoint*-и који захтијевају ауторизацију употребом JWT-а. JwtUtil класа имплементира методе за креирање и провјеру JWT token-а.

6.6 Web апликација за визуелизацију података

За потребе визуелизације прикупљених података имплементирана SPA (*Single Page Application*) *web* апликација са изгледом контролне табле.

За имплементацију одабран је ReactJS развојни оквир, заснован на JavaScript програмском језику. Овај развојни оквир је одабран из разлога што омогућава ефикасан развој SPA апликација.

Поред стандарних библиотека које долазе у оквиру ReactJS развојног оквира, коришћене су и додатне екстерне библиотеке.

За потребе комуникације путем HTTP апликативног протокола са REST сервисима, искоришћена је Axios[38] библиотека. Axios омогућава креирање и упућивање HTTP захтијева, за шта се у позадини користи XMLHttpRequest. Велика предност ове библиотеке је што омогућава једноставно асинхроно слање захтјева и обраду одговора. Асинхрона обрада пристиглог одговора на захтјев, спречава да дође до замрзавања корисничког интерфејса и самим тим побољшава корисничко искуство.

React библиотека подразумијевано нема могућност навигације. За те потребе искоришћена је React Router[39] библиотека која садржи компоненте помоћу којих се може омогућити навигација. Три најпопуларније навигационе компоненте које се користе су BrowserRouter, HashRouter и MemoryRouter. У апликацији је искоришћен BrowserRouter, који за чување историје навигације користи HTML5 History API.

StompJS[40] је библиотека која обезбјеђује STOMP (*Simple Text Oriented Messaging Protocol*) клијента. STOMP је једноставни протокол апликативног слоја за размјену порука. У суштини то је алтернатива другим протоколима за размјену порука, попут AMQP и MQTT. У имплементираном систему STOMP се користи за комуникацију између *web* апликације и REST сервиса у комбинацији са WebSocket протоколом.

React подразумијевано нуди само компоненте које представљају основне HTML (*Hyper Text Markup Language*) елементе без додатног стилизовања. Постоји велики број библиотека које нуде широк спектар готових компоненти које је могуће искористити за изградњу комплексних интерфејса. Једна од таквих библиотека је и MUI[41] (*Material User Interface*). MUI библиотека је искоришћена за изградњу корисничког интерфејса ове *web* апликације.

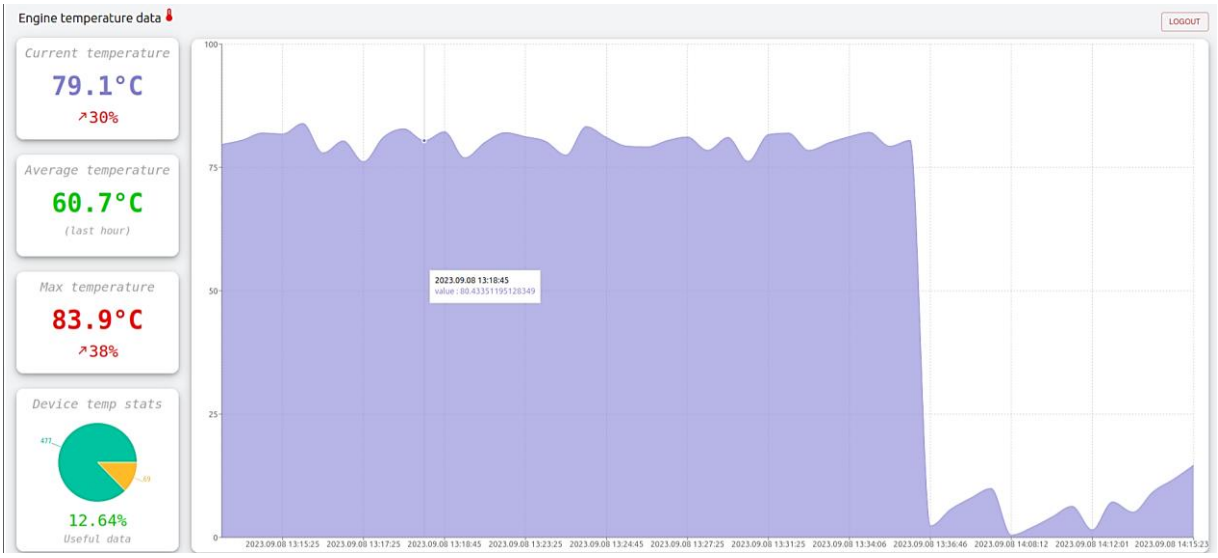
[38]https://axios-http.com
[39]https://reactrouter.com
[40]https://github.com/stomp-js
[41]https://mui.com

Иако MUI библиотека нуди разне графичке компоненте, није погодна за приказ статистичких података и графикона. За приказ дијаграма и графикона са прикупљеним подацима искоришћена је Recharts[42] библиотека.

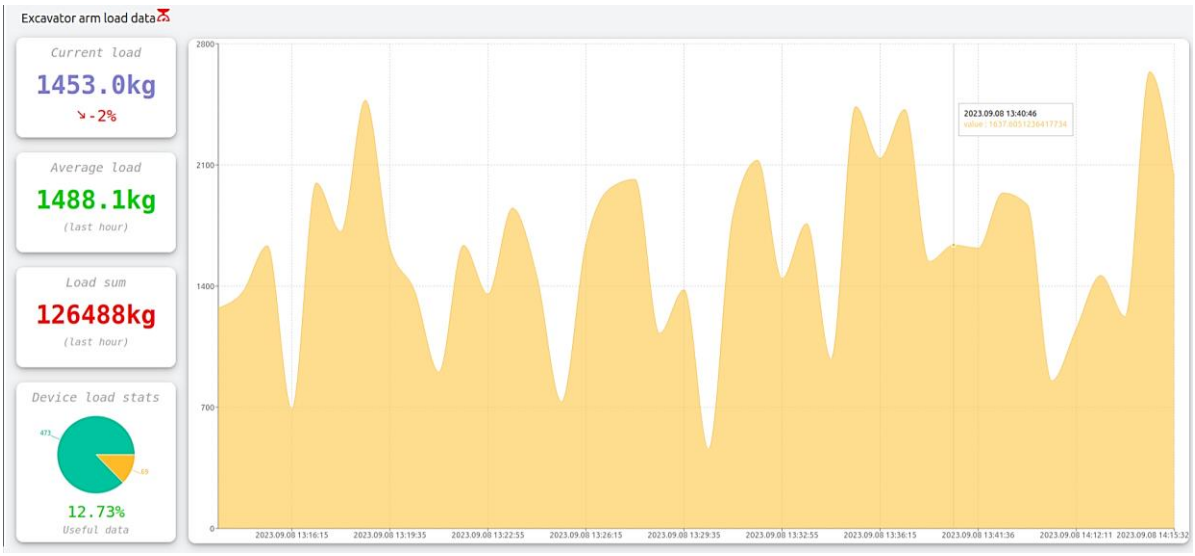
6.6.1 Кориснички интерфејс

Приликом приступа апликацији кориснику се приказује форма за унос креденцијала. Након успјешне аутентикације корисника, омогућава му се приступ подацима одговарајућег уређаја.

Приказ података је подијељен у три цјелине, по једна за сваки тип сензорских података, што је приказано на сликама 6.6, 6.7 и 6.8.



Слика 6.6 – Приказ података о температури мотора

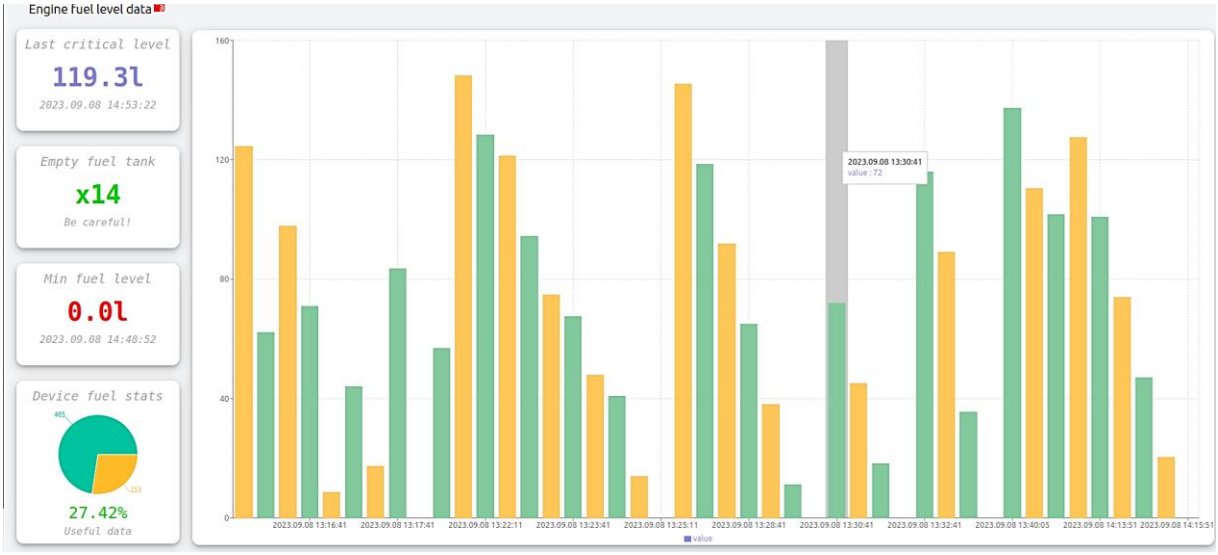


Слика 6.7 – Приказ података о маси терета

Како би се корисницима омогућио приказ актуелних података у реалном времену, комуникација између *web* апликације удаљених сервиса је имплементирана путем WebSocket протокола.

[42]https://recharts.org

Кориснику су приказани сљедећи подаци: тренутна температура мотора, највиша забиљежена температура мотора, просјечна температура мотора, график зависности температуре од времена, маса терета, просјечна маса терета, сума масе терета, график зависности масе терета од времена, задња измјерена, критична количина горива у резервоару и статистички подаци везани за уштеду у преносу прикупљених података путем интернета.



Слика 6.8 – Приказ података о нивоу горива у резервоару

6.7 Deployment система

Све имплементиране апликације, MQTT *broker* и PostgreSQL сервер базе података се извршавају у оквиру засебних Linux контејнера. За креирање и контролисање контејнера искоришћен је Docker[43]алат. Linux контејнери су поједностављене виртуалне машине које врше виртуелизацију доступних хардверских ресурса и које не захтијевају сопствени *kernel*.

За све имплементиране апликације на основу Dockerfile-ова креиране су слике контејнера, које представљају шаблон на основу којих се могу покренути стварни контејнери. За MQTT *broker* и PostgreSQL сервер базе података преузете су готове слике са Dockerhub-а[44].

Како би се поједноставило покретање читавог система искоришћен је Docker Compose алат. Овај алат олакшава рад са системима који се састоје од већег броја контејнера. У docker-compose.yaml фајлу спецификовани су сви потребни контејнери, њихове слике, фајлови са варијаблама окружења, зависности између контејнера, именовани *volume*-и и *bind mount*-и. Именовани *volume*-и омогућавају да генерисани подаци надживе своје контејнере, а *bind mount*-и да се апликацијама, које се извршавају у контејнерима, приликом покретања контејнера обезбиједе конфигурациони фајлови.

Овако подигнут систем, у суштини, представља мрежу Linux контејнера који међусобно сарађују и комуницирају.

Да би се систем могао покренути и извршавати довољно је имати инсталиран Docker алат на произвољној платформи.

[43]https://www.docker.com
[44]https://hub.docker.com

6.8 Потенцијал за продукцију система

Како би се реализовани ситем могао користити у продукцији, потребно је првенствено замијенити скрипту која симулира рад сензора са стварним сензорима. Скрипта је у овом случају кориштена само усљед недостатка стварних сензора. Сензоре би требало прилагодити тако да користе MQTT протокол за слање својих очитавања.

IoT *gateway* апликација би се могла већ у постојећој верзији користити у продукцији, али би било реалније имплементирати кеширање, комплекснију обраду и агрегацију прикупљених података. Могли би се додати механизми који ће подићи аларме у случају детектованих аномалија и недозвољених вриједности у прикупљеним подацима. Функционалност би се, такође, могла проширити увођењем актуаторских компоненти које би се након локалне обраде података, могле правовремено ангажовати ради остваривања адекватног утицаја на физички свијет. На примјер, у случају детекције јако високе температуре мотора, могао би се ангажовати актуатор који ће искључити машину како не би дошло до оштећења, без да се користе удаљени сервиси и тако уносе велико кашњење у реакцију система.

REST сервисе би требало прилагодити извршавању на *cloud* платформи, која је највјероватније и најчешће мјесто извршавања оваквих система. Прилагођавање се највише односи на архитектуру, коју би са слојевите требало трансформисати у сервисно орјентисану, и на комуникацију са екстерним сервисима, попут сервера базе података. Како је све популарнија употреба CaaS платоформи, апликација би се и у продукцији могла извршавати у оквиру Linux контејнера.

Web апликација је сасвим функционална и спремна за продукцијско окружење. Њена функционалност би се могла проширити паралелним приказом података са већег броја уређаја, могућношћу да корисник подешава параметре приказа и контролну таблу прилагођава својим потребама.

Једна од идеја за проширење могућности система је функционалност удаљене контроле крајњих IoT уређаја или IoT *gateway*-а путем имплементиране контролне табле (на примјер гашење уређаја или измјена интервала слања података).

7. ЗАКЉУЧАК

Настанак и развој интернета представља велику прекретницу у историји људског друштва. На самом почетку, није се могао ни наслутити утицај који ће интернет остварити на цјелокупно људско друштво. Иако је као и његов претходних ARPANET, настао из академских потреба, убрзо се појавио широк спектар примјена који је превазишао све планиране оквире. Веома брз развој и ширење глобалне рачунарске мреже довео је до тога да се интернет користи на дневном нивоу и да постане неизоставан дио већине свакодневних људских активности. Интернет је промијенио начин на који се комуницира, начин на који се дијеле информације, начин на који компаније послују (интернет пословање, енг. *e-business*), начин на који се ради (рад са удаљене локације, енг. *remote work*), и отворио нове могућности за учење и образовање. Такође, појавом интернета створили су се неопходно услови за развој бројних других технологија.

Једноставан и практичан начин комуникације, као и доступност мноштва информација и садржаја на дохват руке нису били довољни. Незаустављив развој технологија и константна жеља за напретком, и генерално, за олакшавањем живота, довела је до појаве концепта умрежавања ствари и свакодневних уређаја, тј. IoT-а.

Умрежавање великог броја уређаја и учестала сензорска очитавања довели су до велике потребе за рачунарским ресурсима који ће обрадити и ускладиштити прикупљене податке. Потребе су брзо превазишле могућности IoT уређаја, па се дошло на идеју преноса података путем интернета и њихове обраде на удаљеним системима употребом Cloud Computing модела. Иако се примјена Cloud Computing-а чинила као рјешење свих проблема, употреба ове технологије није се показала оптималном за IoT системе који раде у реалном времену и код којих је вријеме реакције критично. Са једне стране било је потребно смањити вријеме кашњења, које у систем уноси комуникација путем интернета са удаљеним сервисма, а са друге стране, смањити количину саобраћаја који се преноси путем глобалне мреже.

За сада, Fog Computing, се као адаптација Cloud Computing модела за IoT екосистем, показао као сасвим добро рјешења за наведене проблеме. Локална обрада и филтрирање података омогућавају да се правовремено ангажују одговарајући актуатори како би се реаговало на дешавања у физичком свијету, те да се слањем само релевантих података рационалније користи доступни пропусни опсег интернета. Предност овог модела се огледа и у чињеници да се смањује зависност IoT система од интернета, што је јако погодно за IoT системе који имају отежане услове рада и нестабилну конекцију.

Иако су IoT *gateway* уређаји били дио IoT екосистема прије појаве Fog Computing-а, показали су се као готово неизоставни дио овог модела. Првобитна намјена им није била да буду *fog* чворови, него да омогуће комуникацију између хетерогених IoT уређаја и буду мјесто прикупљања података, али због своје природе, показали су се као идеално мјесто за имплементацију *fog* слоја. Како они већ представљају мјесто прикупљања сензорских података, сасвим је природно искористити их за првобитну обраду и агрегацију истих. Близина IoT *gateway*-а крајњим IoT уређајима и комуникација путем локалне рачунарске мреже, омогућили су да се знатно смањи кашњење, које, иначе, у систем уноси комунукација путем интернета.

Како се IoT системи користе у различите сврхе, разликује се и начин обраде прикупљених података. У складу са тим, варирају и захтјеви везани за софтвер који ће вршити њихову обраду, те се ствара потреба за развојем намјенских и софистицираних софтвера за IoT *gateway* уређаје.

У практичном дијелу рада реализован је систем који прикупља, обрађује, складишти и визуелизује податаке о температури мотора, нивоу горива и оптерећењу оперативне руке грађевинске машине, екскаватора. Централни дио овог система је апликација за IoT *gateway* уређај која прикупља, филтрира, агрегира, и просљеђује релевантне сензорске податке. Даљу обраду и трајно складиштење података преузимају удаљени имплементирани REST сервиси. Удаљено надзирање параметара машине омогућено је развојем *web* апликације која релевантне податке визуелизује у виду контролне табле.

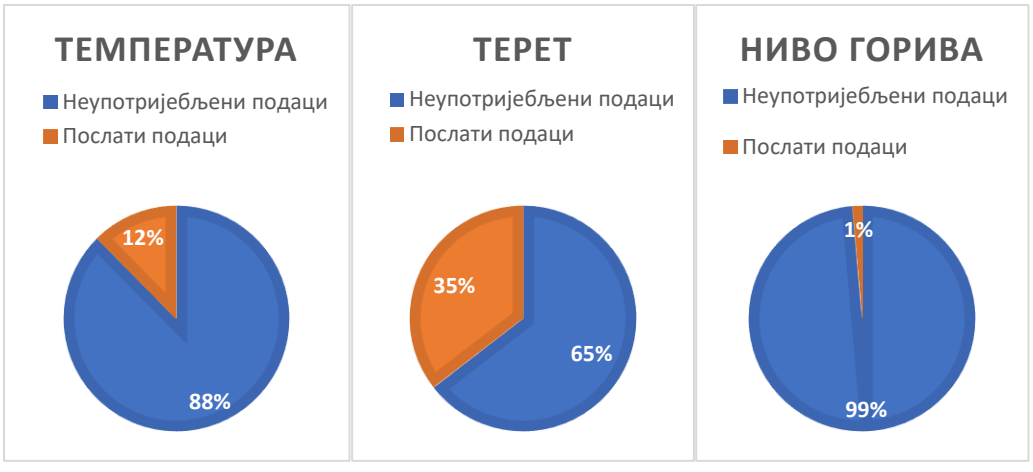
IoT системи који су инсталирани на грађевинским машинама, због природе њихове употребе, не би требали имати висок степен зависности од удаљених сервиса, јер се, у већини случајева, не може обезбиједити стабилан приступ интернету. Управо зато, ови системи представљају идеалан случај за имплементацију локалне обраде прикупљених података. Локална обрада прикупљених података је омогућила да се смањи количина генерисаног саобраћаја, а да се истовремено омогући правовремена реакција на прикупљене параметре. Тако би, у случају прегријавања мотора, било могуће то локално детектовати и ангажовати одговарајуће актуаторе, без бојазни да ће инструкције од удаљених система каснити, или, чак, бити недоступне.

Реализована апликација за IoT *gateway*, који је у функцији *fog* чвора, омогућила је да се у просјеку оствари уштеда у преносу података од 83,57%, на узорку од осам радних часова грађевинске машине. Детаљнији увид у остварене резултате је приказан у табели 7.1 и графицима 7.1.

Табела 7.1 – Резултати практичног дијела

Сензор	*Интервал очитавања сензора*	*Интервал слања података*	*Количина прикупљених података*	*Стратегија обраде података*	*Количина прослијеђених података*	*Проценат уштеде*
Температура	5s	40s	23020B	Просјек	2856B	87,59%
Терет	Стохастично	60s	5 364B	Сума	1904B	64,50%
Ниво горива	5s	- (≈ 365s)	23016B	Филтрирање	316B	98,63%

График 7.1 – Уштеда података



Добијени резултати указују на ефикасност имплементираних принципа Fog Computing-а на рационалнију употребу доступног пропусног опсега и смањење загушења интернет саобраћаја. Иако је имплементирана агрегација података смањила ниво детаља увида у параметре машине, код оваквих и сличних случајева употребе IoT система, то је прихватљиво и толерантно.

ЛИТЕРАТУРА

[1] Mattern Friedemann, Floerkemeier Friedemann, „Vom Internet der Computer zum Internet der Dinge”, Informatik-Spektrum 33(2), стр. 107–121, фебруар 2010

[2] Statista, “Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2023, with forecasts from 2022 to 2030”, https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/, посјећено 09.10.2023. године.

[3] GeeksforGeeks, “Characteristics of Internet of Things”, https://www.geeksforgeeks.org/characteristics-of-internet-of-things/, посјећено 26.07.2023. године

[4] Alexander S. Gillis,TechTarget, “What is the internet of things (IoT)?”, https://www.techtarget.com/iotagenda/definition/Internet-of-Things-IoT, посјећено 29.07.2023. године

[5] Kumar S., Tiwari P., Zymbler M., “Internet of Things is a revolutionary approach for future technology enhancement: a review”, J Big Data 6, 111, јул 2019.

[6] WTECH, „IoT, IoS, IoP and IoD – The Key Technologies Driving Industry 4.0 and Beyond“, https://www.techtarget.com/iotagenda/definition/Internet-of-Things-IoT, посјећено 31.07.2023. године

[7] Spiceworks, „What Is the Internet of Everything? Meaning, Examples, and Uses“, https://www.spiceworks.com/tech/iot/articles/what-is-internet-of-everthing/, посјећено 31.07.2023. године

[8] Cisco, „What Is IoT (Internet of Things)?”, https://www.cisco.com/c/en/us/solutions/internet-of-things/what-is-iot.html, посјећено 31.07.2023. године.

[9] Mansaf Alam, Kashish Ara Shakil, Samiya Khan, “Internet of Things (IoT) - Concepts and Applications”, Springer, 2023

[10] Huawei ICT Academy, “Cloud Computing Techology”, Post & Telecom Press,Beijing, 2023

[11] Nick Antonopoulos, Lee Gillam, “Cloud Computing - Principles, Systems and Applications”, Springer, London, 2017

[12] RedHat, “What is the CaaS?, https://www.redhat.com/en/topics/cloud-computing/what-is-caas, посјећено 08.08.2023. године.

[13] Cloudflare, “What is the cloud? | Cloud definition”, https://www.cloudflare.com/learning/cloud/what-is-the-cloud/, посјећено 03.08.2023. године.

[14] Rajkumar Buyya, Satish Narayana Srirama, „Fog and Edge Computing: Principles and Paradigms“, Wiley, 2019

[15] IBM, “What is Edge Computing?”, https://www.ibm.com/topics/edge-computingm posjećeno 29.08.2023. године

[16] Dutta Pramanik, Pijush & Pal, Saurabh & Brahmachari, Aditya & Choudhury, Prasenjit, “Processing IoT Data: From Cloud to Fog—It’s Time to Be Down to Earth”, 10.4018/978-1-5225-4044-1.ch007

[17] Michaela Iorga, Larry Feldman, Robert Barton, Michael J. Martin, Nedim Goren, Charif Mahmoudi, “Fog Computing Conceptual Model”, NIST специјално издање 500-325, 2018

[18] GeekFlare, “Fog Computing: Explained in 5 Minutes or Less”, https://geekflare.com/fog-computing/, посјећено 29.08.2023. године

[19] Zaigham Mahmood, „Fog Computing: Concepts, Frameworks and Technologies“, Springer , 2018 посјећено 28.08.2023. године

[20] Baeldung, “What is Fog Computing?”, https://www.baeldung.com/cs/fog-computing,

[21] Amir M. Rahmani, Pasi Liljeberg, Jürgo-Sören Preden, Axel Jantsch ,“Fog Computing in the Internet of Things”, Springer 2018

[22] OpenFog Consortium, “OpenFog Reference Architecture for Fog Computing”, 2017

[23] Alam Afrojм Qazi Iqbal Naiyar, Raza Khalid, “Fog, Edge and Pervasive Computing in Intelligent Internet of Things Driven Applications in Healthcare: Challenges, Limitations and Future Use”, 10.1002/9781119670087.ch1, 2020

[24] Božić Velibor, “Applications of fog computing for smart sensors”, 10.13140/RG.2.2.22491.75042, 2023

[25] Mouser Electronics, “Critical role of IoT gateways”, https://www.mouser.mx/blog/critical-role-of-gateways-iot, посјећено 02.09.2023. године

[26] Upadrista Venkatesh, „The Smart IoT Gateway“, 10.1007/978-1-4842-7271-8_6, 2021

[27] Multitech, „What is IoT Gateway?“, https://www.multitech.com/landing-pages/what-is-an-iot-gateway, посјећено 01.09.2023. године

[28] Sam Solutions, „IoT Gateway“ , https://www.sam-solutions.com/blog/iot-gateway, посјећено 03.09.2023. године

[29] Emnify, „IoT Gateway“, https://www.emnify.com/iot-glossary/iot-gateway, посјећено 03.09.2023. године

[30] A2ZTechTalks, „IoT Gateway architecture and fog computing“, https://a2ztechtalks.com/iot-gateway-architecture-and-fog-computing, посјећено 03.09.2023. године