УНИВЕРЗИТЕТ У НОВОМ САДУ

ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА

АРХИТЕКТУРА СОФТВЕРСКОГ СИСТЕМА ЗА ЕЛЕКТРОЕНЕРГЕТСКЕ ПРОРАЧУНЕ ЗАСНОВАНА НА МИКРОСЕРВИСИМА

ДОКТОРСКА ДИСЕРТАЦИЈА

Ментор:
Проф. др Дарко Чапко

Кандидат:
Себастијан Стоја

Нови Сад, 2021. године

УНИВЕРЗИТЕТ У НОВОМ САДУ **ОБРАЗАЦ – 5а**

ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА[1]

Врста рада:	Докторска дисертација
Име и презиме аутора:	Себастијан Стоја
Ментор (титула, име, презиме, звање, институција)	Проф. др Дарко Чапко, ванредни професор, Факултет техничких наука, Универзитет у Новом Саду
Наслов рада:	Архитектура софтверског система за електроенергетске прорачуне заснована на микросервисима
Језик публикације (писмо):	Српски (латиница)
Физички опис рада:	Унети број: Страница 120 Поглавља 9 Референци 108 Табела 18 Слика 60 Графикона 0 Прилога 0
Научна област:	Електротехничко и рачунарско инжењерство
Ужа научна област (научна дисциплина):	Примењени софтверски инжењеринг
Кључне речи / предметна одредница:	Микросервиси, *cloud*, архитектура, електроенергетски прорачуни, ДМС

[1] Аутор докторске дисертације потписао је и приложио следеће Обрасце:
5б – Изјава о ауторству;
5в – Изјава о истоветности штампане и електронске верзије и о личним подацима;
5г – Изјава о коришћењу.
Ове Изјаве се чувају на факултету у штампаном и електронском облику и не кориче се са тезом.

Резиме на језику рада:	ДМС систем је систем који повећава ефикасност, управља, надгледа оптимизује рад и спречава преоптерећења у електроенергетским мрежама. У овој докторској дисертацији се проучава такав један систем чије је време одзива од значаја са свим његовим електроенергетским прорачунима, користећи микросервисе. На основу анализе свих електроенергетских прорачуна дата је архитектура једног таквог система за прелазак са монолитног на микросервисно окружење. Архитектура се ослања на микросервисну архитектуру, где је један сервис подељен на више микросервиса, који сваки за себе обавља своју пословну логику. Након фазе истраживања и сагледавања једне такве архитектуре свих електроенергетских прорачуна једног ДМС система на микросервисно окружење, направљен је прототип на *cloud*-у и за платформу коришћен је *Microsoft Service Fabric*. За тестирање такве архитектуре је бирано два електроенергетска прорачуна који су најзначајнији у једном ДМС систему, а то су тополошка анализа и токови снага. Тестови су извршени у шест експеримената где се врши анализа и за монолитну и за микросервисну архитектуру. Хипотезе докторске дисертације су адресиране кроз поступак примене *cloud*-а у једном ДМС систему, примене једне такве архитектуре и доказивање да примена у микросервисном окружењу даје боље резултате него у монолитном окружењу.
Датум прихватања теме од стране надлежног већа:	28.11.2019.
Датум одбране: (Попуњава одговарајућа служба)	
Чланови комисије: (титула, име, презиме, звање, институција)	Председник: др Александар Ердељан, редовни професор, Факултет техничких наука, Нови Сад Члан: др Јелица Протић, редовни професор, Електротехнички факултет, Београд Члан: др Андрија Сарић, редовни професор, Факултет техничких наука, Нови Сад Члан: др Милан Гаврић, доцент, Факултет техничких наука, Нови Сад Члан: др Срђан Вукмировић, ванредни професор, Факултет техничких наука, Нови Сад Члан, ментор: др Дарко Чапко, ванредни професор, Факултет техничких наука, Нови Сад
Напомена:	

UNIVERSITY OF NOVI SAD

FACULTY OF TECHNICAL SCIENCES

KEY WORD DOCUMENTATION[2]

Document type:	Doctoral dissertation
Author:	Sebastijan Stoja
Supervisor (title, first name, last name, position, institution)	Darko Čapko, Phd, Associate Professor, Faculty of Technical Sciences, University of Novi Sad
Thesis title:	Microservice-based software system architecture for electric power functions
Language of text (script):	Serbian language Latin script
Physical description:	Number of: Pages 120 Chapters 9 References 108 Tables 18 Illustrations 60 Graphs 0 Appendices 0
Scientific field:	Electrical and computer engineering
Scientific subfield (scientific discipline):	Applied software engineering
Subject, Key words:	Microservices, cloud, architecture, electric power functions, DMS

[2] The author of doctoral dissertation has signed the following Statements:
5б – Statement on the authority,
5в – Statement that the printed and e-version of doctoral dissertation are identical and about personal data,
5г – Statement on copyright licenses.
The paper and e-versions of Statements are held at he faculty and are not included into the printed thesis.

Abstract in English language:	Distribution Management System (DMS) is a system that enhances efficiency, manages, supervises, optimizes, and prevents overloading of the electricity networks. This dissertation explores such a system based on microservices and whose response time is significant as it relates to its electric power calculations. Based on the analysis of these calculations, this dissertation also offers the architecture of such a system, and thus enables the transfer from a monolithic to a microservice-based environment. This microservice-based architecture is divided into multiple microservices, each of which performs its individual bussines logic. After relevant extensive research and exploration, a prototype of this architecture was created on the cloud, using Microsoft Service Fabric as a platform. In testing this prototype, two electric power calculations that are most significant in a DMS system were chosen: topological analysis and power flows. Six experiments were conducted in which the analyses for the monolithic and microservice-based architecture were performed. The thesis hypotheses are addressed through the cloud application process within a DMS, showing that the application of such an architecture yield better results in a microservice-based environment than in a monolithic one.
Accepted on Scientific Board on:	28th of November 2019.
Defended: (Filled by the faculty service)	
Thesis Defend Board: (title, first name, last name, position, institution)	President: PhD Aleksandar Erdeljan, Full Professor, Faculty of Technical Sciences, Novi Sad Member: PhD Jelica Protić, Full Professor, Faculty of Elecrical Engineering, Belgrade Member: PhD Andrija Sarić, Full Professor, Faculty of Technical Sciences, Novi Sad Member: PhD Milan Gavrić, Assistant Professor, Faculty of Technical Sciences, Novi Sad Member: PhD Srđan Vukmirović, Associate Professor, Faculty of Technical Sciences, Novi Sad Member, Mentor: PhD Darko Čapko, Associate Professor, Faculty of Technical Sciences, Novi Sad
Note:	

ZAHVALNICA

Veliku zahvalnost dugujem svom mentoru prof. dr Darko Čapku, kao i prof. dr Srđanu Vukmirović na nesebičnoj pomoći i smernicama tokom pisanja doktorske disertacije.

Zahvaljujem se svim svojim profesorima koji su mi preneli svoje znanje tokom svih godina školovanja.

Zahvaljujem se svim svojim kolegama i prijateljima na podršci prilikom pisanja doktorske disertacije.

Najveću zahvalnost dugujem svojoj supruzi Sanji, i mojim ćerkama Seleni i Sieni na beskrajnoj podršci i ljubavi, jer bez njih ova doktorska disertacija ne bi bila napisana.

Takođe, zahvaljujem se svojim roditeljima na podršci i na svemu šta su uradili i naučili me kroz život.

Sebastijan

Sanji, Seleni i Sieni

SADRŽAJ

SPISAK KORIŠĆENIH SKRAĆENICA

Skraćenica	**Puni naziv**
DMS	*Distributed Management System*
IT	*Information Technology*
CPU	*Central Processing Unit*
BDD	*Behavior-Driven Development*
SOA	*Service Oriented Architecture*
CI	*Continuous Integration*
CIM	*Common Information Model*
IEC	*International Electrotechnical Comission*
UML	*Unified Modeling Language*
SCADA	*Supervisory control and data acquisition*
HTTP	*Hypertext Transfer Protocol*
TCP	*Transmission Control Protocol*
UI	*User Interface*
OMS	*Outage Management System*
WMS	*Work Management System*
WFM	*Work Force Management*
GIS	*Geographic Information System*
CIS	*Customer Information System*
EAM	*Enterprise Asset Management*
MDM	*Meter Data Management*
AMI	*Advanced Metering Infrastructure*
RTU	*Remote Terminal Unit*
TA	Topološka aaliza
PP	Prosleđivač Poruka

MP	Model Podataka
TS	Tokovi snaga
ES	Estimacija Stanja
NP	Nije Podržano
SK	Servisi kao komponente
OPJ	Organizacija poslovnih jedinica
RP	Razvoj produkta
MK	Mikroservisna komunikacija
DEC	Decentralizacija
AUTO	Automatizacija
DG	Detekcija greške
WCF	*Windows Communication Foundation*
TCP	*Transmission Control Protocol*
IP	*Internet Protocol*
Volt/Var	Volt /Var optimizacija
KPO	Kapacitet prekidača i osigurača
SM	Spremnost modela
API	*Application Progrmaming Interface*
FIFO	*First In First Out*
AWS	*Amazon Web Services*
AECS	*Amazon Elastic Container*
AEKS	*Amazon Elastic Kubernetes Service*
CPU	*Central Processing Unit*
IP	*Internet Protocol*
PMK	Procena mesta kvara
DG	*Distributed Generation*
DER	*Distributed Energy Resource*
RVP	Restauracija velikog područja
KS	Kratki spojevi
RZ	Relejna zaštita
REK	Rekonfiguracija
AOR	*Area of responsability*
IaaS	*Infrastructure as a Service*

PaaS	*Platform as a Service*
SaaS	*Software as a Service*
VSTLF	*Very short-term load forecast*
STLF	*Short-term load forecast*
MTLF	*Medium-term load forecast*
LTLF	*Long-term load forecast*
KI	Isti koreni
KR	Različiti koreni
E1	Eksperiment 1
E2	Eksperiment 2
E3	Eksperiment 3
E4	Eksperiment 4
E5	Eksperiment 5
E6	Eksperiment 6

SPISAK SLIKA

SPISAK TABELA

1. UVOD

Trenutna potražnja za neprekidnom isporukom električne energije na siguran i efikasan način je značajna za svako domaćinstvo, industriju, odnosno za bilo kog potrošača. Kontrola električne energije se najčešće vrši preko distribuirane aplikacije (softvera) tzv. DMS (eng. *Distribution Management System*) sistema i takvi sistemi spadaju u grupu kritičnih infrastrukturnih sistema.

Kada spominjemo distribuiranu aplikaciju, dostupnost, pouzdanost, replikaciju podataka, skalabilnost, u modernom svetu takva primena ukazuje na *cloud* i na ono što on nudi. Kako su se informacione tehnologije, a i arhitekture samih softverskih sistema, usavršavale, sve više softvera u svetu prelazi na *cloud*, odnosno na jedan koncept, a to su mikroservisi. Detaljniji pregled aktuelnog stanja u ovoj oblasti je dat u poglavlju 2.

Istraživanja sprovedena u ovom radu su usmerena na transformaciju DMS sistema u kontekstu proračuna elektroenergetskog sistema. Funkcije jednog takvog distribuiranog sistema služe za nadgledanje, optimizaciju elektroenergetske mreže kao i predviđanje potrošnje električne energije. Zato jedan ovakav sistem koji radi u realnom vremenu treba da bude efikasan, da ima brz odziv i da uvek bude dostupan. Bez nadgledanje i uvek dostupne aplikacije nema ni optimalne eksploatacije električne energije. Opis i korišćenje proračuna značajnih za ovu doktorsku disertaciju su dati u poglavlju 3.

Mikroservisi su jedan od stilova arhitekture softvera koji u poslednjih nekoliko godina privlači dosta pažnje i predstavlja novi način razmišljanja o distribuiranim aplikacijama. Odnose se na razvoj aplikacija kao skupa malih servisa, nezavisnih jedni od drugih, i svaki za sebe je zaseban proces odnosno mikroservis, a komunikacija mikroservisa se odvija preko određenih definisanih protokola. Drugim rečima, mikroservisi su autonomni servisi koji međusobno rade zajedno. Opis mikroservisa kao i sve njegove osobine i funkcije su date u poglavlju 4.

Primena i korišćenja mikroservisa na dve platforme u *cloud* okruženju, kao i sve osobine i upotreba koje one donose su dati u poglavlju 5.

Kad znamo koji su to proračuni elektroenergetskog sistema, treba locirati i njihovo korišćenje u samoj aplikaciji i arhitekturi jednog takvog tradicionalnog DMS sistema. Arhitektura, opis komponenti, opis problema i hipoteze su date u poglavlju 6.

U ovom radu je opisana i data arhitektura jedne takve aplikacije sa osvrtom na mikroservise i na mikroservisnu arhitekturu, pokazajući šta oni donose i koja je svrha i upotreba jedne takve arhitekture prilikom korišćenja datih proračuna jednog elektroenergetskog sistema. Takođe,

nakon analize datih proračuna iz poglavlja 3, dat je predlog transformacije dva proračuna na *cloud* okruženje. Sve ovo je opisano u poglavlju 7.

Transformacija i njeno korišćenje u datom okruženju je testirano u više eksperimenata. Rezultati, prezentacija i analiza rezultata pojedinačnih eksperimenata kao i analiza svih eksperimenata su dati u poglavlju 8.

Na kraju sam zaključak ovog istraživanja je dat u poglavlju 9, gde su predstavljene prednosti korišćenja jedne takve arhitekture u mikroservisnom okruženju, njeni doprinosi u jednom takvom distribuiranom sistemu, kao i pravci daljih istraživanja.

1.1 Predmet i potreba istraživanja

Predmet istraživanja doktorske disertacije je mogućnost transformacije jednog DMS sistema i njegovih proračuna sa monolitne na mikroservisnu arhitekturu. Ova potreba se pojavila kako bi rešili problem IT (eng. *Information Technology*) kompanija koji se odnosi na to da kada dolazi do bile kakve nove funkcionalnosti u pojedinačnom servisu, da bi se ta funkcionalnost postavila na bilo koji server ili *cloud*, mora celi sistem da se postavi. Iz tog razloga treba razumeti šta je DMS sistem, a šta su mikroservisi. DMS sistem je softver za upravljanje i nadgledanje distribucije električne energije u elektroenergetskim mrežama. Kako bi razumeli razliku između mikroservisne i monolitne aplikacije treba izvršiti poređenje: distribuirane aplikacije se najčešće dizajniraju kao višeslojne sastavljene od tri celine: klijenta, serverske strane i baze. Serverska strana je pojedinačni servis odnosno monolitni servis. Svaka izmena funkcionalnosti u sistemu dovodi do ponovnog postavljanja serverske aplikacije. Za rešavanje problema sa kojima se sreću monolitne aplikacije, razmatraju se mikroservisne aplikacije, odnosno, dizajniranje aplikacije kao skupa pojedinačnih malih servisa - mikroservisa koji mogu nezavisno da se skaliraju tamo gde je to potrebno. U skladu sa ovim, aspekti koji su doprineli istraživanju su (Slika 1):

1. Svaka celina je odvojena u posebni servis (proces, odnosno mikroservis).
2. Razvijanje softvera kao proizvoda odnosno težnja ka tome da se klijentu isporuči servis po servis u zavisnosti kako se završava njegova implementacija.
3. Svaka promena u sistemu ne dovodi do postavljanja celog sistema na *cloud* nego samo pojedinačnog servisa.
4. Raznovrsnost tehnologije - mogućnost korišćenja različitih tehnologija u svakom mikroservisu.
5. Elastičnost.
6. Kapsulacija - skrivanje detalja implementacije jednog mikroservisa od drugog.
7. Skaliranje.

Slika 1 - Aspekti istraživanja

1.2 Cilj istraživanja

Cilj istraživanja je transformacija komponenti DMS sistema sa monolitne na mikroservisnu aplikaciju i očekivan rezultat je metodologija koja će omogućiti primenu mikroservisa u takvom sistemu. Metodologija će imati tri celine:

- Analiza DMS sistema i postojeće arhitekture kako bi se definisalo mesto primene mikroservisne arhitekture,
- Primena mikroservisne arhitekture i implementacije prototipa na jednom proračunu DMS sistema, i
- Upoređivanje dobijenih rezultata predloženom metodologijom i monolitnog rešenja.

1.3 Metode istraživanja

Metode koje će biti primenjene u istraživanju su sledeće:

1. Pregled literature i postojećih rešenja.
2. Razvoj prototipa.
3. Eksperimenti.

S obzirom na cilj istraživanja, a to je primena mikroservisa u DMS sistemima vrlo je važno osloniti se na prethodna iskustva prilikom primene mikroservisa u takvim sistemima, a to su sistemi koji zahtevaju brz odziv. Istraživanje nije samo usmereno na DMS sisteme nego i na druge sisteme gde su mikroservisi primenjivi i gde je potreban brz odziv i koji rade u realnom vremenu. Cilj pregleda literature je da se identifikujuju, analiziraju i interpretiraju postojeća

rešenja i sve tvrdnje za naučna pitanja. Na taj način bi se postojeća iskustva iskoristila prilikom dizajniranja predloženog rešenja.

Pregled literature i postojećih rešenja će biti sproveden u tri faze:

1. Planiranje.
2. Sprovođenje.
3. Izveštavanje.

U toku faze planiranja najvažnije je identifikovati naučne radove koji imaju isti predmet istraživanja kao i ova disertacija, a to su:

- Definisanje mikroservisa i njihovo poređenje sa monolitnim aplikacijama.
- Metodologije prelaska sistema sa monolitne na mikroservisnu aplikaciju.
- *Cloud* koje je od suštinskog značaja za prelazak na mikroservisnu aplikaciju.
- DMS sistemi i njegove proračuni.

U toku faze sprovođenja sistemačnog pregleda literature, selektovani radovi se analiziraju, a zatim se određuju koji radovi su prikladni za ovu disertaciju.

U toku faze izveštavanja, kreira se poglavlje u disertaciji koji opisuje odabrane radove, njihovo istraživanje, mane i prednosti i kako oni utiču na ovu disertaciju.

Na osnovu sistematičnog pregleda literature i istraživanja, kreira se **prototip rešenja**. Prototip obuhvata predloženu arhitekturu za elektroenergetske proračune zasnovanu na mikroservisima. Razvijeni prototip omogućava izvođenje eksperimenata u cilju provere njegove primenljivosti.

Treća metodologija koja se koristi su **eksperimenti** i oni treba da iskažu da li su tvrdnje ovom disertacijom tačne ili netačne. U toku ove faze će se odrediti okruženje koje će se koristiti prilikom eksperimenta. Eksperimenti će se izvršavati *n* puta kako bi se dobila tačnost eksperimenta i koristiće se metodologije statističke obrade rezultata kako bi se i pokazala njihova tačnost.

1.4 Mogućnost primene

Rezultat doktorske disertacije je predlog arhitekture za elektroenergetske proračune u mikroservisnom okruženju i primena takve arhitekture u realnim sistemima. Na ovaj način elektrodistributivna preduzeća dobijaju softver za nadzor elektroenergetske mreže pomoću koga na brži i pouzdaniji način vrše isporuku električne energije. U okviru ove disertacije će predloženo rešenje biti testirano na dva proračuna DMS sistema, ali se i implementacija ostalih proračuna izvodi po istoj proceduri i može biti realizovana u nekim budućim istraživanjima.

2. PREGLED AKTUELNOG STANJA U OBLASTI

U prethodnim godinama, sve više aplikacija prelazi na *cloud* i na upotrebu mikroservisa u *cloud* okruženju [1][2], odnosno kompanije transformišu aplikacije sa monolitne na mikroservisnu arhitekturu i okruženje, pokazajući prednosti i performanse korišćenja mikroservisa. Takođe, kompanije i prilikom izrada novih aplikacija koriste mikroservisno okruženju zbog samih osobina i prednosti koje ovo okruženje donosi. Istraživanje ovog rada se oslanja na već postojeća istraživanja i prebacivanja sa monolitne na mikroservisno okruženje [3][4] elektroenergetskih proračuna DMS sistema.

Postoje različita istraživanja u ovoj oblasti koja će u nastavku biti objašnjena. Autori u [5] opisuju korišćenje jednog servisa odnosno mikroservisa u *Docker*-u [6] u tzv. decentralizovani klaster *Serfnode*, gde se klasteri oglašavaju međusobno pružanjem servisa. Pored toga, prikazali su proširivost *Serfnode* tako što su obezbedili rešenje za problem sinhronizacije sistema datoteka. Takođe, *Docker* tehnologija se koristi i u [7] gde autori razmatraju neke koncepte o mikroservisnoj arhitekturi i kako *Docker* može da pomogne u uspešnoj primeni korisne mikroservisne arhitekture. Isto tako i autori u [8] pokazuju značaj postavljanja aplikacija na *cloud* korišćenjem mikroservisa i pokazuju svoju aplikaciju *SmartVM* koja se bazira na *Docker* kontejnerima.

Autori u [9] su napravili poređenja između primene monolitnih i mikroservisnih aplikacija. Oni su analizirali strukturu mikroservisne arhitekture da bi postavili velike aplikacije na *cloud* kao skup malih servisa koji se mogu razvijati, testirati, skalirati, upravljati i nadograđivati nezavisno, omogućavajući kompanijama da steknu agilnost, koji smanjuje kompleksnost i proširuju svoje aplikacije u *cloud* na efikasniji način. Rezultati pokazuju da se smanjuje vreme izvršavanja kod razvijenih aplikacija u mikroservisnom okruženju. Zatim, isti autori nastavljaju svoja istraživanja u [10], gde pokazuju kako postaviti *web* aplikaciju na *cloud* u tri različita scenarija: monolitno okruženje, mikroservisno okruženje nad kojim upravlja korisnik i mikroservisno okruženje nad kojim upravlja *cloud* provajder (Microsoft, Amazon, Google, i drugi). Korišćenjem mikroservisnog okruženja smanjuju se infrastrukturni troškovi i do 13%, ali korišćenje bilo kog *cloud* provajdera znatno smanjuju troškove, čak i do 77%, što dodatno ohrabruje korisnike da koriste mikroservisna okruženja nad kojim upravlja *cloud* provajder.

Takođe, autori u [11] vrše poređenja između monolitnih i mikroservisa sa aspekta implementacije i razvoja aplikacije, ali bez njihovog testiranja sa aspekta performansi. Sličan primer autori pokazuju u [12], postavljenjem (eng *deploying*) mrežnih funkcija na mikroservisno i monolitno okruženje i upoređujući dobijene performanse. Pored toga, autori su računali broj instanci koje su potrebne za skaliranje u oba slučaja i pokazali kako se ponaša CPU (eng. *Central Processing Unit*) u tim slučajevima.

Testiranje je jedna od najvažnijih faza razvoja softvera. Poslednjih godina, razvoj BDD-a (eng. *Behavior-Driven Development*) postao je jedan od najpopularnijih agilnih procesa razvoja softvera i često se koristi u razvoju mikroservisa. Autori rada [13] su objasnili BDD testove kroz različite repozitorijume (eng. *repository*) i pokazali kako treba smanjiti sukob između programera i testera dopuštajući im da samostalno razvijaju svako svoj deo aplikacije na odvojenim repozitorijumima dok postižu iste ciljeve aplikacije. Autori u [14] pokazuje korišćenje mikroservisne arhitekture u DevOps, za koje izdvajaju pet stvari koje su naučili koristeći mikroservise. Prvo, rešili su problem postavljanja svakog zavisnog servisa, odnosno, mikroservisa, da bi pokrenuli izolovani servis na svojoj mašini, dok su istovremeno uvodili kolaboraciju dinamičkog servisa. Drugo, problem interfejsa (eng. *interface*) servisa - promenom interfejsa jednog servisa može dovesti do greške celog sistema. Ovaj problem se može rešiti na dva načina: preko servisa za verzionisanje, ali rezultira kompleksnom implementacijom, ili drugi način je korišćenje dizajn šablona (eng. *design pattern*), gde je tim odgovoran za servis i može biti siguran da su korisnici zadovoljni uslugom. Treće, razvoj distributivnog sistema zahteva kvalifikovane programere. Četvrto, pokazuju koliko je izrada dizajn šablona važna. Konačno (peto), fleksibilnost sistema, što je olakšalo razvoj. Ovo su korisni uvidi pri izradi aplikacija u mikroservisnom okruženju.

Braun i ostali [15] pokazuju koncept servisno orijentisane arhitekture SOA (eng. *Service Oriented Architecture*) za podršku velikih količina podataka oslanjajući se na mikroservise i mikroservisnu arhitekturu. Prototip je implementiran i iskorišćen za evulaciju dve *web* aplikacije koje pripadaju različitim domenima i pokazano je da mikroservisna arhitektura može da se koristi u više aplikacionih domena koji dele podatke, različitih baze podataka korišćenjem onoga šta mikroservisi pružaju, a to je skalabilnost odnosno horizontalna skalabilnost.

Skalabilnost je veoma važna osobina u razvoju i u korišćenju aplikacija u *cloud* okruženju. U radu [16] opisuju skalabilan mehanizam za *stateful* aplikacije u *cloud* okruženju korišćenjem mikroservisa. Autori su izdvojili dva problema. Problem broj jedan se odnosi da li koristiti *stateless* ili *stateful* mikroservise, i zaključak je *stateless* kao što je i opisano u [5][17][18] i problem broj dva se odnosi na pravljenje centralnog servera za komunikaciju sa ostalim mikroservisima u aplikaciji i prosleđivanje (eng. *routing*) poziva. Oslanjajući se na ova dva problema su napravili eksperiment gde su pokazali da je mikro skaliranje u odnosu na makro skaliranje fleksibilnije u monolitnim aplikacijama. Takođe, u radovima [19][20] je postignuta horizontalna skalabilnost, bazirajući se na *actor* modelu koji je postavljen kao set manjih servisa odnosno mikroservisa za masivnu obradu podataka. Iz rada se može zaključiti da se scenariji, za *big data* koji imaju nepovezan ili slabo referenciran model podataka pogodnih za paralelizaciju, najbolje primenjuju sa mikroservisima, a ne monolitno, jer je horizontalna skalabilnost jedna od glavnih stavki u takvim rešenjima.

U radu [21], autori pokazuju koji su potezi potrebni prilikom korišćenja mikroservisa i mikroservisne arhitekture na jednom proizvodu u jednoj firmi. Takođe, prilikom same transformacije pokazuju koji principi transformacije sa monolitne na mikroservisne arhitekture su iskorišćeni i koje lekcije su primenjene, odnosno, naučene prilikom same transformacije. Primera radi: kako jedan mali tim koji razvija softver može lako da pređe na takvu vrstu organizacije i koji su izazovi na koje se nailaze. Takođe, pokazuju tvrdnju da mikroservisi imaju tendenciju da se sve više izučavaju, s obzirom da kompanije razvijaju svoj softver oslanjajući se na mikroservise.

Prilikom transformacije softvera sa monolitne na mikroservisnu arhitekturu nailazi se na razne izazove. Jedan od zaključaka kada treba preći na mikroservise je kada kompanija narasla i

kada kod softver nije više održiv. U radu [22] opisuju probleme i izazove na koje se nailaze prilikom prebacivanja jednog softvera sa monolitne na mikroservisnu arhitekturu, počevši od organizacionih promena do tehničkih detalja na šta sve treba obratiti pažnju i koji su to izazovi prilikom prelaska na mikroservise. Na kraju je dat zaključak o dobiti cele ove transformacije i svega šta mikroservisi donose kao novinu u razvoju softvera i arhitekture softvera.

U [23] opisuju kako identifikovati monolitne servise, koji mogu da se prebacuju u nezavisne male mikroservise i kako to uraditi algoritamski. Oslanjajući se na algoritme, prikazuju transformaciju kroz tri faze: monolitna, grafovska i na kraju mikroservisna faza, za koja postoje dva koraka transformacije između tih faza, a to su: korak konstrukcije i korak grupacije. Uzima se monolitna arhitektura jednog sistema i pretvara se u grafički prikaz pomoću koraka konstrukcije, a zatim se sve to grupiše u drugom koraku, nakon čega se dobija mikroservisna arhitektura. Oslanjajući se na ovo, napravljen je prototip koji pokazuju performanse dobijene upotrebom mikroservisa.

Postavljanje (eng. *deployment*) i CI (*Continuous Integration*) mikroservisa je od suštinskog značaja i to pokazuju autori u [24] gde predlažu automatski sistem koji pomaže u ova dva problema. Rešenje je postavljeno na kontejnerima i testiranjem socijalne mreže, pokazane su performanse mikroservisa, odnosno, kako je mikroservisna arhitektura bolja od monolitne. Takođe u ovom radu je pokazano kako automatski postavljanje rešenja na *cloud* značajno ubrzava i vreme i novac, za razliku od korišćenja običnih virtualnih mašina.

U [25] opisuju mikroservisnu arhitekturu, njihovu implementaciju i upravljanje distribuiranim mikroservisima u konceptu mikroservisne distribuirane aplikacije. Ovaj rad znatno pomaže u razumevanju mikroservisa i sa kojim problemom IT kompanije se suočavaju prilikom transformacije na ovakav model. Takođe, u radu je prikazano i par ključnih arhitektonskih preporuka na šta tačno treba obratiti pažnju prilikom kreiranja mikroservisne aplikacije.

U [26] autori se bave prebacivanjem na *cloud* okruženje jednog dela pametnih distibutivnih mreža, a to je AMI (eng. *Advanced Metering Infrastructure*). Data je arhitektura prelaskom na *cloud* okruženje, deljenjem pojedinih servisa na mikroservise. Arhitektura je testirana sa tri različite analitičke aplikacije: predviđanje energije, potrošnje, predviđanje kvaliteta električne energije i predviđanje krađe energije. Dobijeni rezultati su u nekim slučajevima poboljšani i do 10% što je vrlo dobro rešenje i ukazuje na to da predstavljena metodologija može biti primenjena u pametnim distributivnim mrežama.

U [27] autori, ukazuju i daju rešenje za prebacivanje dva proračuna DMS sistema na *cloud* okruženje, a to su: praćenje naponskog nivoa i estimacija stanja. Arhitektura takvog rešenja je data i testirana, odnosno, simulirana u realnom vremenu i ukazuju na prednosti predložene arhitekture, a to je smanjenje vremena simulacije i količine podataka prebacivanjem na *cloud* okruženje. Takođe jedna takva arhitektura, sa samo dva proračuna, je napravljena da može podržati još takvih proračuna, a da rezultati i dalje budu isti.

U radu [28] je data arhitektura EMS sistema oslanjujući se na mikroservise, koji ukazuje da se korišćenjem takve arhitekture poboljšala skalabilnost, održavanje i pouzdanost sistema. Takođe dati eksperimenti ukazuju da su performanse pouzdanosti sistema znatno bolje nego kod postojećih EMS sistema. Dati predlog sistema je implementiran i koristi se u svakodnevnom radu, što ukazuju još više na potrebu za primenom mikroservisa.

Svi ovi sistemi AMI, EMS, DMS su deo pametnih distributivnih mreža i njihova analiza sa bezbednostih aspekata je data u radu [29], jer prilikom prelaska na *cloud* potrebno je pogledati sve rizike i sve aspekte bezbednosti, kao što su poverljivost, integritet i pouzdanost.

Postojeća istraživanja se uglavnom fokusiraju na mikroservisnu arhitekturu, kako se grade mikroservisi, kako ih koristiti i čine paralelu između monolitnih i mikroservisnih aplikacija. Veoma su značajno doprineli prilikom istraživanja u okviru ove disertacije, jer kao što je i rečeno, nije bio cilj istražiti samo sisteme koji se bavi nadgledanjem i upravljanjem elektroenergetskom mrežom, nego ostale distribuirane sisteme sa kritičnom infrastrukturom.

3. OPIS, KORIŠĆENJE I PRORAČUNI DMS SISTEMA

Elektrodistributivne mreže [30] su deo elektroenergetskog sistema koji služi da distribuciju električne energije od napojnih transformatorskih stanica do krajnjih potrošača [31]. Elektroenergetske mreže su razvijaju iz dana u dan širom sveta i sve distribucije su trenutno u procesu transformacije, što znači da teže ka pametnim distributivnim mrežama (eng. *smart grid*). Jedan od glavnih razloga zašto dolaze do ovih transformacija jeste posledica sledećih uticaja: naponski problemi, dvosmerni tokovi snage, višestruki izvori sa kvarovima, kao i velika dimenzija mreže, jer količina podataka koja opisuje elektrodistributivne mreže većih gradova prevazilazi kapacitete računara koji se koriste u praksi i zato je potrebno razviti programsku podršku za upravljanje elektrodistributivnim mrežama omogućavajući paralelnu obradu. Sve ovo podržava jedan DMS sistem i njegovi proračuni koji mogu da se koriste u datom sistemu.

DMS sistem [32][33] je sistem koji ima za cilj prikupljanje, organizaciju, prikaz i analizu podataka u realnom vremenu elektodistributivnog sistema. DMS sistem takođe može da vrši operacije nadgledanja, planiranja i izvršavanja u cilju povećanja efikasnosti sistema, optimizacije protoka i sprečavanje preopterećenja sistema.

DMS sistem poseduje određene elektroenergetske proračune koje će se u ovoj disertaciji obraditi, a to su:

- Topološka analiza,
- Tokovi snaga,
- Putanja,
- Estimacija stanja,
- Kratki spojevi,
- Procena mesta kvara,
- Restauracija velikog područja,
- Rekonfiguracija,
- Prognoza,
- Spremnost modela,

- Volt/Var optimizacija,
- Kapacitet prekidača i osigurača, i
- Relejna zaštita.

3.1 Modelovanje elektroenergetske mreže

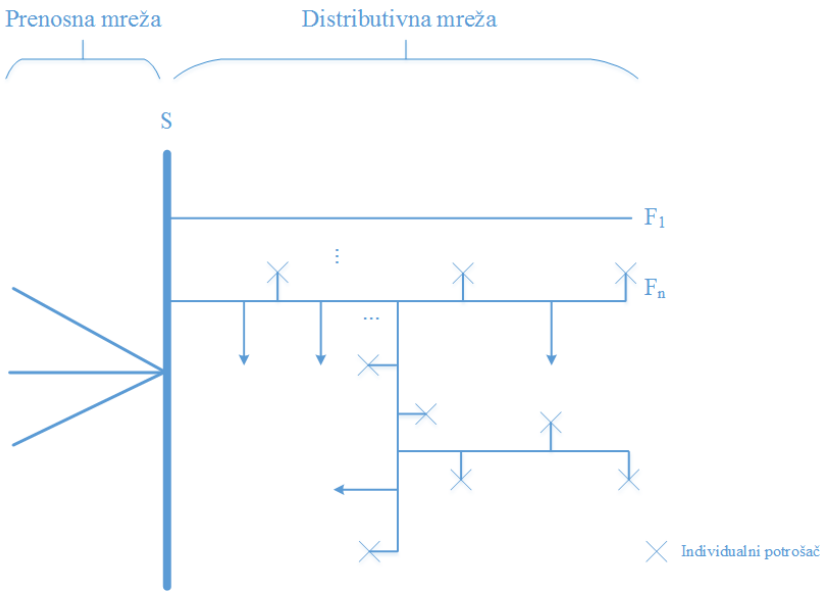
Kako bi se razumeli elektroenergetski proračuni, potrebno je objasniti najpre osnovne pojmove u elektroenergetskoj mreži, kao i teoriju grafova, na osnovu čega se formira model elektroenergetske mreže u obliku grafa.

3.1.1 Osnovni pojmovi elektroenergetske mreže

Svaki elektroenergetski sistem se sastoji od četiri podsistema [34][35]:

1. Podsistema proizvodnje koji predstavlja poizvodnju električne enegije.
2. Podsistema prenosa koji čini prenosna mreža.
3. Podsistema distribucije koji čini distributivna mreža.
4. Podsistema neposredne potrošnje koji čine individualni potrošači.

Iz gore navedenog možemo zaključiti da se elektroenergetska mreža sastoji od prenosa i distribucije (Slika 2).



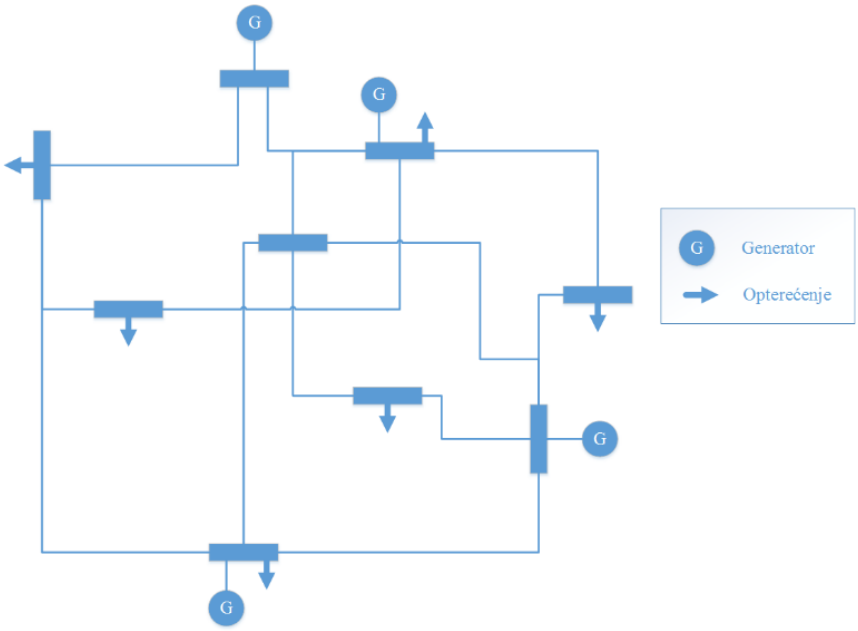
Slika 2 - Podsistem distribucije [34]

Navedena dva segmenta elektroenergetske mreže se mogu okarakterisati kao:

1. Prenosna mreža je mreža za koju se kaže da je "složena". Drugim rečima to je upetljana mreža koja pruža posrednu ili neposrednu vezu proizvodnje električne energije, npr. elektrane sa krajnjim potrošačima [34].

2. Distributivna mreža je mreža koja "spaja" podsistem prenosa i podsistem potrošnje. Mesto na kom se spajaju prenosna mreža i distribucija je na slici iznad (Slika 2) označeno sa *S*. Levo od tačke *S* se nalazi prenosna mreža, a desni deo jeste distributivna mreža. Distribucija se zasniva na fideru, gde fider prolazi pored svakog individualnog potrošača koji se na njega priključuju (obeleženo na slici sa F_1, F_n). Fideri sa svojim grananjem do mesta priključka individualnih potrošača predstavljaju distributivnu mrežu. Drugim rečima distributivna mreža se može nazvati i radijalnom jer do potrošača se stiže samo sa jedne strane, za razliku od upetljane (prenosne mreže) gde se može stići sa više strana do krajnjeg potrošača. Distributivne mreže ili elektrodistributivne radijalne mreže se sastoje od skupa korena i ostrva. Koren čine svi elementi koji su napajani preko jedne napojne grane i to se zove energizovan deo mreže, dok ostrva predstavljaju skup elemenata koji su izolovani sa svih strana od energizovanih delova otvorenim prekidačima [34].

Radijalne mreže uglavnom imaju jednu tačku napajanja i isti smer tokova aktivne snage u svim granama (Slika 3) [34].

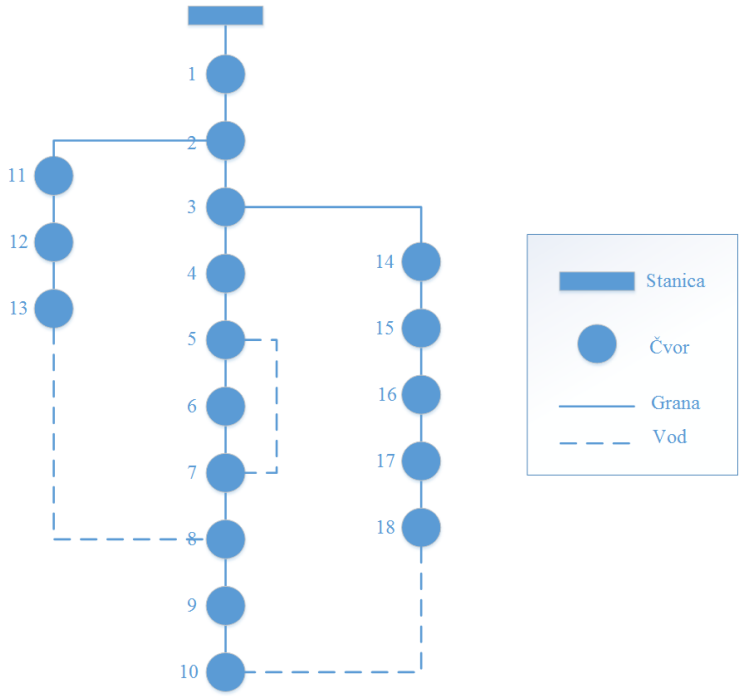


Slika 3 - Primer prenosne (upetljane) mreže

Distributivna mreža se može podeliti na dva tipa [34]:

1. *balansirana* je mreža gde su svi elementi uravnoteženi. Uravnotežen element elektroenergetskog sistema predstavlja bilo koji trofazni element, primer: generator, vod, transformator, potrošač ili drugi, koji svojim priključenjem u elektroenergetski sistem ne remeti režim sistema. Ovakve mreže se mogu pronaći u Evropi.

2. *nebalansirana* je mreža gde je bar jedan element neuravnotežen. Ovakve mreže se mogu pronaći u Americi.



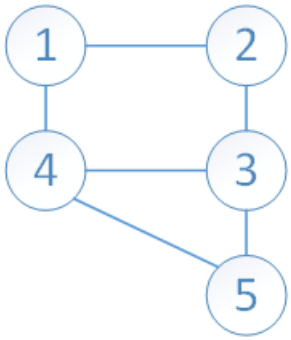
Slika 4 - Primer distributivne (radijalne) mreže

Transformatori služe za prilagođavanje napona uređaja ili mreža koji se povezuju na elektroenegetske mreže [35]. Sa slike iznad (Slika 2) se može videti da su prenosna i distributivna mreža spojene, što znači da napon mora da se reguliše tj. da se transformiše, a to se radi pomoću transformatora. Električna energija se iz prenosne mreže, koja je 400(220)/115 kV/kV, preuzima visokonaponskim transformatorom na sub-prenosnu mrežu koja je 110 kV. Sa sub-prenosne mreže električna energija se preuzima sa srednjenaponskim transformatorom 110/21(10.5) kV/kV. Dalje se električna energija preuzima za distributivnu mrežu sa niskonaponskim transformatorom 20(10)/0.4 kV i dalje električna energija sa voda koji je 0,38 kV stiže do individualnih potrošača [34][35][36].

Model distributivne mreže [37] se sastoji od: transformatorske stanice (eng. *substation)*, koje su povezane vodovima (eng. *AC line segment*), i koje napajaju potrošače (eng. *energy consumer*). Transformatorske stanice sadrže određene opremu (eng. *equipment*) i čvorove (eng. *connectivity node*) koji preko terminala (eng. *terminal*) povezuju datu opremu. Poznatiji tipovi opreme su provodna oprema (eng. *conducting equipment*) i transformatori (eng. *power transformer*) [38].

3.1.2 Teorija grafova

Jedna od oblasti matematike koja je zastupljena i u računarstvu, a njena oblast istraživanja je vezana za osobine i algoritme grafova jeste teorija grafova. Grafovi imaju dva dela i čine ih tačke odnosno čvorovi i linije, odnosno, grane, koje međusobno povezuju dva ili više čvora. Graf *G(V,E)* je uređen par, a elementi su sledeći: *V* je skup čvorova, a *E* je skup grana. Svaka grana ima dva kraja na kojima se nalaze čvorovi. Grafovi se predstavljaju dijagramima, gde su čvorovi predstavljeni kružićima, a grane se prikazuju linijama spajajući odgovarajuće kružiće. Primer ovakvog grafa koji ima 5 čvorova i 6 grana je prikazan na slici ispod (Slika 5).



Slika 5 - Primer grafa elektroenergetske mreže

Grafovi se u većini slučajeva koriste za opis modela ili strukture podataka. Jedan od primera grafova jeste *web* prezentacija, gde su čvorovi grafa same *web* stranice, a grane grafa hiperlinkovi između stranica.

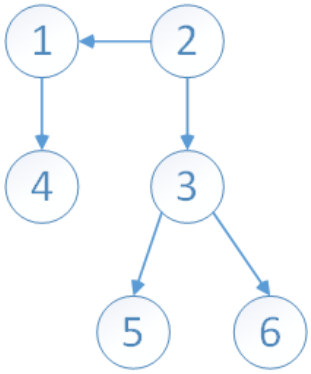
Jedan od najznačajnih delova računarske nauke jeste proučavanje algoritama koji rešavaju probleme upotrebom grafova. Mreže imaju mnogo primena u proučavanju praktičnih aspekata teorije grafova i time se bavi tzv. analiza mreža. Analiza mreža je posebno značajna za probleme modeliranja i analize mrežnog saobraćaja, recimo interneta.

Teorija grafova je grana matematike za koju se može reći da se bavi i proučavanjem osobina geometrijiskih oblika. Veze fizičkih objekata su predstavljene pomoću linija i tačaka. Topologija električnih mreža se bavi izučavanjem osobina grafova. Grafovi imaju i veliku primenu i u nekim drugim naukama kao što su elektrotehnika, računarstvo, hemija, fizika, biologija, sociologija i vojne nauke [35].

Graf je neusmeren ako grana koja spaja dva čvora, primer sa iznad (Slika 5), npr. čvorove 1 i 2 je ista kao i grana koja spaja čvorove 2 i 1. Ako se uzima u obzir da su to dve različite grane onda je graf usmeren [35].

Težinski grafovi su grafovi kojima se dodaje osobine težine i pogodni su za predstavljanje nekih problema, na primer mreže puteva gde se težina odnosi na dužinu puta između dva čvora. Težinski graf koji je usmeren zove se mreža [35].

Grane su paralelne ako se dve ili više grana grafa spajaju u dva ista čvora. Grana koja spaja čvor sa samim sobom zove se petlja (Slika 6) [35].



Slika 6 - Primer stabla

Graf koji nema petlje, a ni paralelne grane zove se prost graf. Graf je prazan ako nema nijednu granu, a nulti graf nema nijedan vrh. Stablo je povezan graf koji nema nijednog ciklusa, odnosno puta koji počinje i završava se u istom čvoru (Slika 6). Stablo u kome je jedan čvor izdvojen naziva se korensko stablo, a čvor se naziva koren stabla [35].

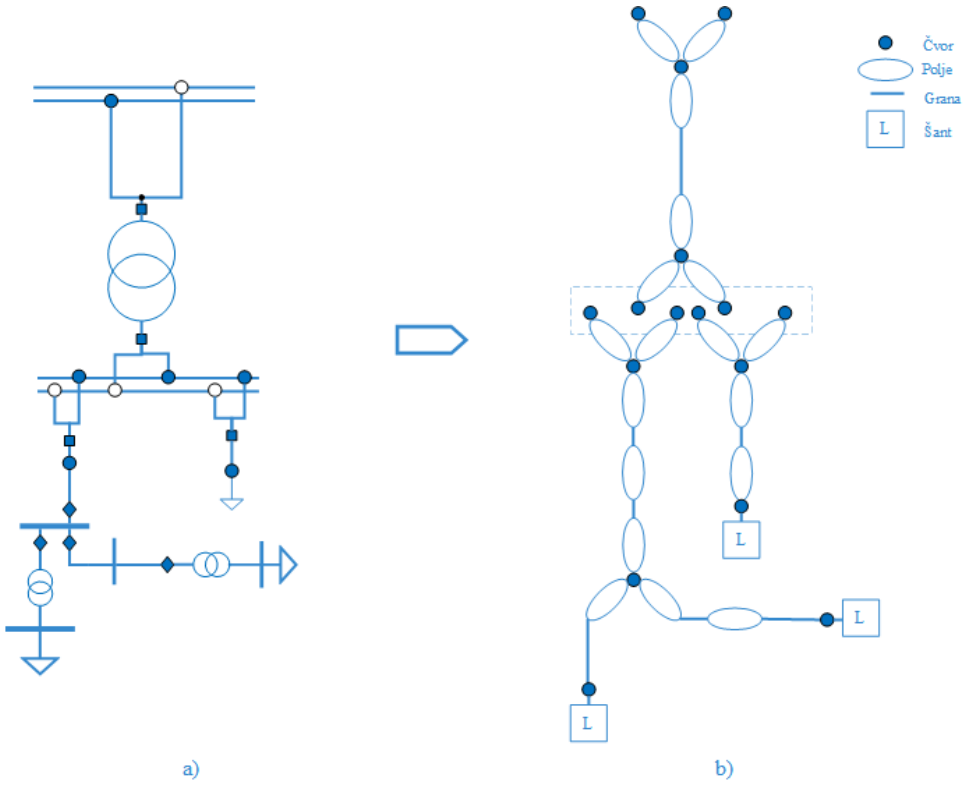
3.1.3 Graf elektroenergetske mreže

Elektroenergetske mreže se predstavljaju grafovima. Čvorovi grafa se predstavljaju sabirnicama visokonaponskih postrojenja, a same grane koje povezuju čvorove grafa - vodove i transformatore. U granama elektroenergetskih mreža mogu biti postavljeni rasklopni uređaji, i relejna zaštita gde se pojedini elementi mogu uključiti odnosno isključiti sa iste. Na krajnje čvorove grafova se postavljaju potrošači i distributivni generatori tzv. šant. Šant je grana koja povezuje čvorove nultog potencijala [35].

Graf elektroenergetske mreže se sastoji od sledećih elemenata:

- čvor,
- grana,
- polje i
- šant.

Jedan primer transformacije distributivne mreže na graf elektroenergetske mreže je prikazan na slici ispod (Slika 7).



Slika 7 - Transformacija distributivne mreže (a) [39] u graf elektroenergetske mreže (b)

Sa slike iznad (Slika 7), sa leve strane (7a) se može videti primer distributivne mreže koji ima sabirnice, prekidače, transformatore i krajnje potrošače, a sa desne strane (7b) može se videti interpretacija takve distributivne mreže preko grafa elektroenergetske mreže sa navedenim elementima. Svaki prekidač je prikazan sa poljem, transformator sa granom, svako uvezivanje elemenata kao čvor i krajni potrošač kao šant [35].

3.2 Elektroenergetski proračuni

U ovom poglavlju će biti opisani svi elektroenergetski proračuni koji se uzimaju u razmatranje za ovu doktorsku disertaciju.

3.2.1 Topološka analiza

Topološka analiza (TA) [40][41] elektroenergetskih mreža ima zadatak da elektroenergetsku mrežu opiše na takav način kako bi funkcije koje se sprovode nad mrežom bile brze i efikasne. Takođe, topološka analiza distributivnih mreža omogućava korisnicima da na bolji način imaju sliku elektroenergetske mreže kako bi ustanovili njenu energizaciju, deenergizaciju, regionalnu pripadost, itd.

Tri zadatka topološke analize, a definišu se preko modela topološke analize [42], su:

1. Utvrđivanje korena mreže, odnosno energizovanih delova mreže i ostrva, odnosno, deenergizovanih delova mreže,
2. Određivanje potrebnih parametara mreže kako bi se odradili elektroenergetski proračuni, i
3. Prikaz korisnicima trenutnog stanja mreže, a to znači sledeće:
    a. *Energizacija* - pokazuje da li je element priključen na izvor napajanja,
    b. *Naponski nivo* - određuje ako je element energizovan, kom naponskom nivou pripada,
    c. *Pripadnost transformatorskoj oblasti* - pokazuje transformatora na koji je element povezan,
    d. *Pripadnost fideru* - prikazuje fider na koji je element povezan,
    e. *Faznost* - daje aktivne faze svakog elementa,
    f. *Napajanje korena* - određuje napojni vod koji napaja elemenat, i
    g. *Upetljanost* - prikazivanje da li element formira petlju.

Obrada topološke analize elektroenergetskog sistema se izvršava u nizu procesa, a to su [41]:

1. Vrši se obrada pojedinačnih faza u smeru od napojne grane. Najpre na početku liste se upisuju indeksi matrice interkonekcije donjeg čvora napojne grane i napojnih grana paralelnih korena, koja je FIFO (eng. *First In First Out*). U toku svake

iteracije uzima se indeks čvora sa početka liste i iz matrice interkonekcije se omogućava dobavljanje njemu incidentnih grana i susednih čvorova, kao i informacija da li se na nekom od krajeva grane nalazi otvoreno polje.

2. Zatim se na svakoj grani postavlja smer koji pravi razliku između gornjeg i donjeg čvora. Grane se definišu po pojedinačnoj fazi, iz razloga što nebalansirane mreže mogu biti dvosmerne za razliku od balansiranih.

3. Vrši se struktuiranje mreže po slojevima i podaci o sledećoj grani koja joj prethodi se čuvaju.

4. Sačuva se aktivna faznost elemenata koji su pod naponom.

5. Vrši se bojenje grafa po energizaciji koji može biti:

    a. *undefined* - neodređen status energizovanosti,

    b. *grounded* - element je uzemljen i pod naponom,

    c. *unenergized* - element nije pod naponom,

    d. *partial* - element je pod naponom ali ne u svim fazama,

    e. *energized* - element je pod naponom, i

    f. *meshed* - element pripada petlji.

6. Vrši se detekcija i obrada petlji. Granu kojom se zatvorila kontura potrebno je sačuvati kako bi kasnije mogla da se radi obrada detektovane petlje.

7. Vrši se detekcija i obrada ostrva. Prilikom skeniranja mreže, sve petlje koje su otvorene se sačuvaju radi detekcije ostrva. Skeniranje detektovanog ostrva je suprotno od skeniranja korena, potrebno je da se struktuira mreža po slojevima. Elementima ostrva potrebno je pridružiti nivo energizacije i na taj način će se obojiti deo distributivne mreže koji nije pod naponom.

3.2.2 Tokovi snaga

Tokovi snaga (TS) [35][43][44][45][46] služe za izračunavanje stabilnih stanja radijalnih i slabo upetljanih primarnih visokonaponskih mreža, kao i stanja sekundarnih niskonaponskih mreža. Proračun tokova snaga može da se izračuna za niskonaponske mreže tako da se u realnom vremenu (eng. *realtime*) otkriju ograničenja napona u mreži i koji su elementi preopterećeni. Proračun tokova snaga je namenjen za određivanje varijabilnih stanja mreže balansiranih i nebalansiranih na osnovu poznatog korenskog napona i podataka o opterećenju svih čvorova. Generalno, model tokova snaga elektroenergetske mreže predstavlja matematički opis aktivne i reaktivne snage u sistemu gde napajanje treba da bude jednako zbiru opterećenja i gubitaka datog sistema. Model se sastoji od skupa složenih jednačina koji opisuje vektor stanja date mreže odnosno skup složenih napona u svim elektroenergetskim čvorovima mreže.

Tokovi snaga uključuju proračun varijabli i kompletno stanje elektroenergetske mreže. Kada je napon korena poznat i postoje podaci o potrošnji odnosno opterećenju čvora, sva ostala

stanja se mogu estimirati. Stanje mreže je prikazano složenim naponima, strujama, tokovima stvarne aktivne i reaktivne snage, padom napona, gubicima, itd. Sve ove promenljive izračunaju tokovi snaga.

Ulazni parametri proračuna tokova snaga, odnosno podaci na osnovu kojih se vrši proračun su [44][47]:

- Model elektroenergetske mreže,
- Topološka analiza,
- Fazne veličine i uglovi korenskih napona,
- Opterećenje potrošača (aktivna i reaktivna snaga),
- Regulatori generatora,
- Baterije (aktivna i reaktivna snaga),
- Regulator napona,
- Regulator kondezatora,
- Granice regulatora napona,
- Operativna ograničenja (naponska ograničenja, ograničenja opterećenja, ograničenja proizvodnje generatora).

Glavni rezultati proračuna tokova snaga su [44][47]:

- Složeni naponi (fazni i linijski naponi) - veličine napona i fazni uglovi za sve čvorove,
- Fazne vrednosti trenutnih veličina i faktora snage za sve sekcije i transformatore,
- Stvarni, reaktivni i prividni protoci snaga svih sekcija, transformatora, transformatora struje, spojnica i uređaja,
- Gubici aktivnih i reaktivnih snaga elemenata mreže,
- Pad napona sekcija i transformatora u sistemu,
- Pozicije teretnog menjača (eng. *tap changer*) - deo transformatora koji omogućuje odabir već definisanih vrednosti samog transformatora korak po korak birajući željenu vrednost,
- Nulta komponenta struje,
- Maksimalni strujni, reaktivni i prividni protoci snage transformatorskog sloja za transformatorske i servisne transformere (eng. *transformer bank and service transformer*),

- Neuravnoteženost napona kod potrošača, i
- Kvalitet rezultata proračuna u sledećem obliku: nepoznat, loš, dobar i upitan.

Proračun tokova snaga se može izračunati na sledeće akcije u DMS sistemu:

- Izmena topologije - otvaranje odnosno zatvaranje nekog prekidača na mreži, promena pozicije teretnog menjača, itd.,
- Promena vrednosti regulatora na regulatoru napona i regulatora na generatoru,
- Periodično na zahtev korisnika - konfiguracija kad može proračun da se pokreće, i
- Na trenutni zahtev korisnika.

3.2.3 Putanja

Putanja je specifičan proračun odnosno analiza grafa elektroenergetskog sistema, koji se koristi na zahtev korisnika. Usko je vezana sa topološkom analizom i modelom elektroenergetske mreže koji pokazuje putanju od specifične tačke i prikazaje koji su to elementi afektovani u datoj putanji. Putanja se završava na elementima koje korisnik specifira da želi da nađe u datoj putanji [39].

Postoje tri metode putanje [39]:

1. Putanja na gore prikazuje putanju na gore od tačke A do željene tačke B, šetajući se po grafu elektroenergetske mreže ka izvoru napajanja.
2. Putanja na dole prikazuje putanju na dole od tačke A do željene tačke B, šetajući se po grafu elektroenergetske mreže na dole.
3. Putanja u oba pravca prikazuje putanju u oba smera od tačke A do tačke B, šetajući se po grafu elektroenergetske mreže na gore i na dole.

Input parametri putanje su:

- Specifična tačka od koje počinje putanja,
- Model elektroenergetkse mreže, i
- Topološka analiza.

Izlazni parametri za sva tri tipa putanje, su svi elementi na datoj putanji sa datim opcijama za dati tip putanje.

3.2.4 Estimacija stanja

Jedan od najvažnijih proračuna u elektroenergetskim sistemima je estimacija stanja (ES) [48]. Ona služi za optimizaciju ostalih proračuna kao što je prethodno opisana tokovi snaga [31][46].

Proračun estimacija stanja sadrži dva koraka:

1. Transformacija svih merenja u vektor stanja čiji je cilj da se ta merenja dobijaju kao pouzdan vektor promenljivih stanja. Vektor promenljivih stanja podrazumeva skup koji opisuje stanje sistema, odnosno skup napona, čvorova i nenominalnih odnosa transformacije klasičnih i faznih regulacionih transformatora [31].

2. Proračun ostalih sistemskih promenljivih koje su značajne za datu konfiguraciju [31].

Drugim rečima proračun estimacija stanja služi za estimaciju kompletne elektroenergetske mreže uzimajući u obzir uređaje koje su pod SCADA (eng. *Supervisory control and data acquisition*) sistemom [49] i podatke o opterećenju.

Ulazni podaci koji su potrebni za proračun estimacije stanja su [48]:

- Analogna telemetrisana merenja,
- Pseudo merenja koja su potrebna samo ako nema dovoljan broj uređaja koje su pod SCADA-om,
- Parametri elemenata mreže,
- Topološka analiza odnosno topološki model elektroenergetske mreže, i
- Aktivna i reaktivna snaga.

Rezultati estimacije stanja, mogu biti [48]:

- Procenjeno opterećenje i proizvodnja,
- Procenjeni naponi (veličina i ugao po svakoj fazi) korena elektroenergetske mreže,
- Procenjeno stanje mreže - složeni naponi, kao i druge promenljive stanja, kao što su: veličine faktora struje i snage, aktivni i reaktivni tokovi snage za sve sekcije, transformatore i šantove (potrošače, generatore, motore i kondenzatore),
- Procenjena zadata vrednost regulatora napona,
- Procenjeni status regulisanih kondenzatora,
- Procenjeni položaj prekidača regulacionih transformatora,
- Procenjena vrednost merenja (trenutna, aktivna i reaktivna snaga, prividna snaga, faktor snage, veličina napona i ugao napona),
- Kvalitet procenjene vrednosti merenja,
- Alarm za merenja kod kojih je procenjeno-izmereno odstupanje veće od dozvoljene unapred definisane vrednosti, i

- Statistička analiza izvršenja i kvaliteta merenja.

3.2.5 Kratki spojevi

Proračun kratkih spojeva (KS) služi za proračun svih kvarova u elektroenegetskoj mreži. Obrađuje proračun celog sistema na koje utiču kratki spojevi, uključujući sve napone i struje u sistemu sa kvarom. Struja kvara se definiše kao struja koja se javlja kao posledica kvara bilo gde u sistemu. Ova struja se veoma razlikuje od struje u normalnom stanju tj. neutralne struje. Struja kvara ima veću veličinu od normalne struje i usmerena je ka tački kvara. Da bi se pravilno instalirale električne instalacije i odabrala potrebna oprema, kao i da bi se pravilno postavili zaštitni uređaji, potrebno je izračunati struju kvara (kratkog spoja) za svaku tačku u mreži. Stoga je proračun kvara jedan od osnovnih proračuna koji se vrši tokom projektovanja i analize neispravnih električnih sistema [35][46].

Rezultati proračuna kvara koriste se u brojne svrhe. Neki od njih su:

- Analiza kvarnog stanja (praćenje i predviđanje efekata simuliranih kvarova u sistemu),
- Projektovanje trafostanica u vezi sa opremom i sistemom uzemljenja - utvrđivanje da li su struje zemljospoja u granicama koje diktira bezbednost postupak i preporuka,
- Provera da li su struje kratkog spoja unutar prekidača i mogućnosti prekida osigurača, i
- Verifikacija rada zaštite i osetljivosti i određivanje mesta kvara pomoću trenutne metode.

Za rad proračuna kratkih spojeva su potrebni sledeći ulazni parametri [46]:

- Model elektroenergetske mreže,
- Topološka analiza,
- Tokovi snaga, i
- Ulazne opcije za izvršavanje samog proračuna u elektroenergetskoj mreži.

Izlazni parametri proračuna kratkih spojeva su [46]:

- Struja kvara,
- Napon kvara,
- Napon na mestu kvara,
- Struja na mestu kvara,
- Vršna (eng. *peak*) vrednost struje,

- Putanje kvara, i
- Dužina od izvora,

3.2.6 Procena mesta kvara

U proračunu procena mesta kvara (PMK) se uključuju set aplikacija koje se koriste za lokaciju i izolaciju kvara i snabdevanje potrošača koji su ostali bez električne energije odnosno napajanja. Prekid napajanja je stanje kod koga treba preuzimati mere da bi se obezbedilo-nastavilo napajanje, i u skladu sa tim tri dela čine procenu mesta kvara [46][50][51]:

1. Lokacija kvara - procenjuje se gde se nalazi mesto kvara, pa zatim se locira mesto kvara, odnosno u kom je delu elektroenergetske mreže.
2. Izolacija kvara - set akcija koje moraju da se odrade da bi se kvar izolovao i na bezbedan način restaurirao.
3. Restauracija kvara - određuje se plan za obnavljanje napajanja na delu elektroenergetske mreže gde je kvar lociran. Za restauraciju se koriste razne metode, ali sve zavisi u kom delu elektroenergetske mreže se kvar nalazi i od tipa kvara.

Ulazni parametri ovog proračuna su:

- Model eletroenergetske mreže,
- Topološka analiza, i
- Tokovi snaga.

Izlazni parametri procena mesta kvara su:

- Rezultat gde su prikazani svaki element elektroenergetske koji je deo krvara,
- Dužina kvara od jednog kraja do drugog,
- Lista akcija koje moraju da se odrade da bi se kvar izolovao, i
- Lista akcija da bi se napanje vratilo nazad potrošačima.

3.2.7 Restauracija velikog područja

Restauracija velikog područja (RVP) je proračun koji na identičan način kao i proračun procena mesta kvara određuje kvar samo što RVP to radi na srednje i visokonaponskim stanicama i u prenosnom delu elektroenergetske mreže. Opšta svrha ovog proračuna služi za obnavljanje velikih područja elektroenergetske mreže i utvrđivanje plana restauracije za dato područje [52].

U proračunu restauracije velikog područja se pokreću tri dela kako bi se ustanovio i restaurirao kvar [52]:

1. Klasifikacija kvara - klasifikacija kvara na osnovu tipa kvara. Kvar može biti opisan pomoću tipa elementa i naponskog nivoa, primer: transformer na visokonaponskom nivou od 100kV do 400kV.

2. Izolacija kvara - izolacija u delu gde relejna zaštita nije odradila izolaciju.

3. Restauracija kvara - obnavljanje napajanjem velikog dela elektroenergetske mreže.

Ulazni parametri proračuna su [52]:

- Model elektroenergetske mreže,
- Topološka analiza,
- Tokovi snaga, i
- Tip proračuna koje može biti: napajanje sabirnice, napajanje fidera, brisanje fidera.

Izlazni parametri na osnovu tipova ulaznog parametra mogu biti [52]:

- Napajanje sabirnice - nudi opcije na koji način se sabirnica može ponovo napajati gde se svaka opcija rangira,
- Napajanje fidera - nudi opcije na koji način se fider može napajati kao i sve potrebne podatke za date potrošače kao što su aktivna snaga, broj potrošača, prioritet samih potrošača itd., i
- Brisanje fidera sa određene sabirnice - daje rezultate za svaki fider pojedinačno.

3.2.8 Rekonfiguracija

Rekonfiguracija (REK) elektroenergetske mreže je jedan od značajnih proračuna. Služi za optimalnu konfiguraciju mreže, što podrazumeva proračun statusa uključenosti rasklopnih uređaja, od koje zavisi trenutna topološka analiza elektroenergetske mreže [46][53].

Rekonfiguracija mreže se najčešće koristi da bi se postigla ekonomičnost, sigurnost i pouzdanost date elektroenergetske mreže, a najčešće da bi odredili kvalitet napona, smanjenja gubitka aktivne snage, kritični pad napona, pouzdanost napajanja, troškove manipulacija, debalans opterećenja na izvodima, debalans opterećenja na visoko i srednjenaponskim transformatorima.

Ulazni parametri rekonfiguracije su [46]:

- Model eletroenergetske mreže,
- Topološka analiza, i
- Tokovi snaga.

Izlazni parametri rekonfiguracije su [46]:

- Optimalna konfiguracija elektroenergetske mreže prema navedenom cilju optimizacije,

- Lista akcija za dati proračun koji moraju da se odrade na samu elektroenergetsku mrežu radi primene rekonfiguracije,

- Spisak prekidača koji su preskočeni tokom izračunavanja za normalno otvorene prekidače i razlog za izuzeće ovih prekidača, i

- Prednosti koje pruža optimalna konfiguracija, tj. razlika između stvarne i optimalne konfiguracije s obzirom na gubitke aktivne snage, indeks neravnoteže transformatora, trenutne struje, itd.

3.2.9 Prognoza

Proračun prognoze [54][55] je jedan od ključnih proračuna koji pomažu u radu sa eletroenergetskim mrežama, u smislu kupovine i prodaje električne energije, dodavanja u mrežu novog agregata, održavanja, prebacivanja tereta i planiranja infrastrukture elektroenergetske mreže. Predviđanje proizvodnje električne energije i opterećenje, pruža bolju sliku o tome kako će se sistem ponašati u budućnosti i da li treba preduzeti neke intervencije.

Predviđanjem opterećenja, topološke analize elektroenergetske mreže i parametara elemenata i objekata elektroenergetske mreže, ispunjavaju se uslovi za izračunavanje stanja mreže (korišćenjem energetskog proračuna tokova snaga). Poznavanje stanja mreže pretpostavlja poznavanje npr. napona izračunatog na svakom čvoru u sistemu, protoka struja, protoka aktivne i reaktivne snage, itd. Poznavanje stanja mreže u određenom trenutku pruža puno mogućnosti za donošenje odluka u sadašnjem stanju, kao i za obavljanje analiza „šta ako“.

Postoje četiri tipa proračuna za predviđanje prognoza [56]:

1. Bliska prognoza opterećenja (eng. *very short-term load forecast* VSTLF) se koriste za procenu električnih opterećenja u satnim periodima između 10 do 30 minuta, uzimajući u obzir istorijsko procenjeno opterećenje čiji parametri će biti navedeni dole kao ulazni parametri proračuna, istorijska vremenska prognoza na osnovu istorije vremenskih prilika, vrste dana kao što su radni dani, vikendi, praznici i nedeljni i spoljne uticaje kao što su predstojeći vremenski uslovi [57][58].

2. Kratkoročno predviđanje opterećenja (eng. *short-term load forecast* STLF) se koristi za procenu električnih opterećenja u jednakim vremenskim intervalima (30 minuta) do 198 sati unapred, uzimajući u obzir istorijsko procenjeno opterećenje, istorijska vremenska prognoza na osnovu istorije vremenskih prilika, vrste dana i spoljne uticaje kao što su predstojeći vremenski uslovi [59][60].

3. Srednjoročno predviđanje opterećenja (eng. *mid-term load forecast* MTLF) se koristi za procenu električnih opterećenja u jednakim vremenskim intervalima (primera radi 60 minuta) do 52 nedelje uzimajući u obzir istorijat procene opterećenja, istorijsku vremensku prognozu na osnovu istorije vremenskih prilika, vrstu dana kao što su radni dani, vikendi i praznici i spoljne uticaje kao što su predstojeći vremenski uslovi [61].

4. Dugoročno predviđanje opterećenja (eng. *long-term load forecast* LTLF) služi za predviđanje godišnjih električnih opterećenja za budući period od jedne do 20 godina [62][63][64].

Zajednički ulazni parametri za proračun kratkoročnu i srednjoročnu prognozu su [57][58][59][62]:

- Istorijski podaci estimirane vrednosti npr. aktivne i reaktivne snage.
- Istorijski podaci o vremenskoj prognozi na osnovu istorije vremenskih prilika.
- Dani u nedelji.
- Godišnji raspored za prognozu neobnovljivih generatora.
- Dodatni prametar za proračun kratkoročne prognoze je predikcija prognoze vremena, kako će vremenska prognoza izgledati u narednih osam dana za kratkoročno predviđanje i jedan dan za blisko predviđanje. Za ovu predikciju su potrebni sledeći parametri:
    - Isti vremenski podaci koji su konfigurisani da se koriste kao istorijski podaci,
    - Brzina vetra - koristi se samo za vetrenjače, i
    - Izolacija - koristi se samo za solarne generatore.
- Dodatni parametar za proračun srednjoročnu prognozu opterećenja su rezultati vremenske prognoze, primera radi kako će izgledati u naredna 52 dana, i to sledeća tri parametra:
    - Temperatura,
    - Brzina vetra - koristi se samo za vetrenjače, i
    - Izolacija - koristi se samo za solarne generatore.

Ulazni parametri za dugoročno prognozu opterećenja su [63]:

- Istorijski podaci,
- Podaci o vremenskoj prognozi,
- Vremenski faktori,
- Ekonomski i demografski podaci i njihova vremenska prognoza,
- Geografski položaj, i
- Vremenske nepogode.

Izlazni parametri za prognoze su sledeći [57][58][59][63]:

1. Za blisku prognozu izlazni parametri su prognoza opterećenja i proivodnje za narednih 30 minuta.

2. Za kratkoročnu prognozu izlazni parametri su prognoza opterećenja i proizvodnje za narednih 8 dana.

3. Za srednjoročno predviđanje izlazni parametri su prognoze opterećenja i proizvodnje za naredne 52 nedelje.

4. Za dugoročno predviđanje izlazni parametri su godišnja prognoza opterećenja i proizvodnja za narednih od 1 do 20 godina.

3.2.10 Spremnost modela

Proračun spremnost modela (SM) [65] određuje nivo trenutne pripremljenosti elektroenergetske mreže za neke od naprednih proračuna, npr. Volt/Var optimizacija, kako bi se ti napredni proračuni mogli pokretati.

Kako bi se odredilo spremnost modela elektroenergetske mreže definiše se indeks spremnosti modela od 0 do 10. Primera radi ako elektroenergetska mreža ima indeks 10 znači da ima 100% ispunjenost modela i da napredni proračuni su spremni za korišćenje. U suprotnom može doći do toga da deo funkcija neće biti u potpunosti iskorišćen.

Izvršavanje proračuna sastoji se od sledećih koraka:

- Proračun stanja mreže procenom stanja (tokovi snaga) nakon okidača (promena topološke analize, promena zadate vrednosti, promena izmerenih vrednosti izvan unapred definisanih pragova, isticanje unapred definisanog vremena) i pružanje rezultata,

- Obrada svih relevantnih rezultata proračuna estimacije stanja i tokova snaga, i kao rezultat dobijanjem stvarnog indeksa spremnosti modela i ocene za svaki od kriterijuma,

- Obrada stvarnih i prethodno izvedenih kriterijuma spremnosti modela i izvođenje kumulativnog indeksa i kumulativne ocene za sve kriterijume, i

- Izvođenje kvaliteta spremnosti modela.

Podaci koji su potrebni za proračun spremnosti modela su [65]:

- Model elektroenergetske mreže,

- Tokovi snaga, i

- Estimacija stanja.

Rezultati proračuna spremnosti modela su [65]:

- Indeks spremnosti modela koji ukazuje na tačnost podataka elektroenergetske mreže,

- Kriterijum spremnosti modela koji ukazuje na stvarni trenutak i posmatrani period, i

- Kvalitet spremnosti modela (loš, delimičan, dobar, itd.) koji se definiše na osnovu indeksa spremnosti modela.

3.2.11 Volt/Var optimizacija

Proračun Volt/Var optimizacija (Volt/Var) upravlja svim naponima i reaktivnim snagama u elektroenergetskoj mreži. Proračun određuje optimalnu Volt/Var strategiju za postizanje operativnog cilja i to u pet režima [66]:

1. Nadgledanje i kontrola. Služi za nadgledanje i kontrolu elektroenergetske mreže kako bi se elektroenergetska mreža i resursi održavali u optimalnom stanju u skladu sa ulaznim parametrima za optimizaciju [66].

2. Optimizacija gubitaka je jedan od glavnih režima kad se radi o radu sa elektroenergetskim mrežama, a to je smanjenje reaktivne snage i minimizacija gubitka aktivne snagem [66].

3. Smanjenje potražnje je režim koji služi za smanjenje maksimuma opterećenja u elektroenergetskoj mreži [66].

4. Neoptimizacioni režim je režim koji smanjuje napon u elektroenergetskoj mreži na neoptimizovan način koji se koristi u vrlo retkim slučajevima i to samo kad je potrebno brzo da se smanjuje maksimum opterećenja [66].

5. Hitan režim se koristi kada se mreža nalazi u neočekivanom stanju zbog velikog broja kvarova. Tipičan primer je oluja kad dosta potrošača ostaju bez napajanja [66].

Podaci koji su potrebni za Volt/Var optimizaciju su [66]:

- Model elektroenergetske mreže,

- Topološki model elektroenergetske mreže,

- Aktivna i reaktivna snaga, i

- Estimacija stanja.

Rezultati Volt/Var optimizacije su [66]:

- Preporučene sekvence za akcije nad elektroenergetskom mrežom,

- Rezultati za smanjenje opterećenja,

- Rezultati uštede energije,

- Vandredni režimi koji izbegavaju uštede opterećenja, i
- Ušteda energije smanjenje gubicima u elektroenergetskoj mreži.

3.2.12 Kapacitet prekidača i osigurača

Proračun za proveru kapaciteta prekidača i osigurača (KPO) koriste se za proveru prekida (prekida i stvaranja) kapaciteta preklopnih uređaja tj. prekidača i osigurača u elektroenergetskim mrežama. Glavni cilj ovog proračuna jeste da obezbedi jednostavnu korisničku proveru kapaciteta prekidačkih uređaja za datu topološku analizu elektroenergetske mreže i za razmatrani tip kvara u elektroenergetskoj mreži. Provera se može izvršiti za jedan ili više biranih uređaja elektroenergetske mreže [67].

Ovaj proračun je značajan u oblastima kad se dodaje novi generator u elektroenergetskoj mreži koji potencijalno mogu imati značaj uticaj na vrednost i parametre struje kvara. Dodavanje novih elemenata u elektroenergetskoj mreži može nekad promeniti vremenske konstante koji mogu rezultovati različitim uslovima od uslova koji su navedeni za originalni dizajn i izgradnju elektroenergetske mreže.

Rezultati proračuna za prekidače su [67]:

- Maksimalna struja kvara,
- Maksimalni pik (eng. *peak*) struje,
- Maksimalna toplotna struja,
- Maksimalna radna struja, i
- Maksimalni radni napon.

Rezultati proračuna za osigurače su [67]:

- Maksimalna prekidna struja,
- Maksimalna radna struja,
- Maksimalni radni napon, i
- Maksimalna struja magnećenja.

Proračun se koristi od strane ljudi koji rade na analizi elektroenergetske mreže u *offline* modu. To znači da proračun ne utiče na rad elektroenergetske mreže u realnom vremenu, nego isključivo za analizu. Sa druge strane proračun pomaže da se detektuje da li negde u mreži postoji kvar ili dolazi do istog na osnovu gore opisanih funkcija.

3.2.13 Relejna zaštita

Relejna zaštita (RZ) je proračun u kome se daju svi rezultati vezano za proračune u elektroenergetskim mrežama i obezbeđuje funkcionalnost relejne zaštite [46].

Relejna zaštita u elektroenergetskim mrežama omogućuje sigurnost, pouzdanost i kvalitet snabdevanja pri čemu se moraju uvažiti sledeći principi [46]:

- Pouzdanost - sposobnost zaštite da pravilno radi. U slučaju pojave kvara, sistem relejne zaštite mora pružiti sigurnost ispravnog rada, kao i mogućnost da se izbegne nepravilan rad tokom kvarova.

- Brzina - sposobnost zaštite da ukloni kvar u minimalnom roku, kako bi se izbegla oštećenja opreme.

- Selektivnost - sposobnost održavanja kontinuiteta napajanja isključivanjem minimalnog dela mreže potrebnog za izolaciju kvara.

- Troškovi - maksimalna zaštita uz najniži mogući trošak.

Za rad relejne zaštite kao ulazni parametri potrebno je [46]:

- Model elektroenergetske mreže,

- Topološka analiza,

- Tokovi snaga, i

- Proračun kvara.

Izlazni parametri relejne zaštite su [46]:

- Za specifičan proračun kvara i tip samog kvara, svi releji koji se nalaze na samoj putanji,

- Svi kvarovi na svakom vodu u zoni relejne primarne zaštite,

- Svi kvarovi na svakom vodu za odabrani *circuit*,

- Podešavanje svakog releja, i

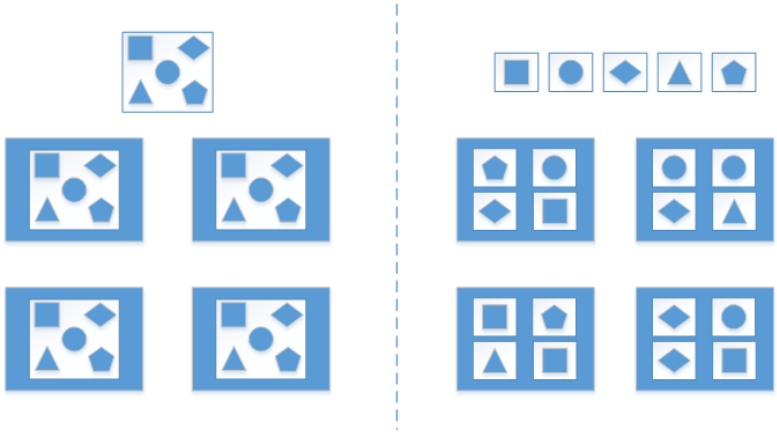
- Izveštaj koji sadrži sve informacije o relejima, minimalna stuja/impedansa kvara, itd.

4. MIKROSERVISI

Poslednjih godina jedan od čestih termina u softverskoj arhitekturi su mikroservisi. Ovaj termin se može objasniti na sledeći način: razvoj jednog softvera kao skupa paketa pojedinačnih usluga, gde je svaki paket celina za sebe i svi paketi međusobno komuniciraju preko određenih protokola: HTTP (eng. *Hypertext Transfer Protocol*), TCP (eng. *Transmission Control Protocol*), i drugi. Drugim rečima, mikroservisi su autonomno servisi koji svi rade zajedno [18][22][68][69][70][71][72][73].

Kako bi se što bolje razumeo koncept mikroservisa važno je napraviti poređenje sa monolitnim aplikacijama i to na primeru aplikacija koje sadrže tri celine: klijenta, serversku stranu i bazu podataka. Serverska strana obično dobija zahtev od klijenta preko određenog protokola za komunikaciju (primer: HTTP, TCP, i drugi), izvršava određeni proračun ili logiku i nakon toga upisuje podatke u bazi i šalje nazad podatke klijentskom interfejsu za prikaz. Serverska strana ovakve aplikacije je monolitna. Izmenom bilo kakve logike na serverskoj strani zahteva ponovo rekreiranje i postavljanje nove aplikacije. Uzimajući u obzir mane monolitnih aplikacija formirana je mikroservisna arhitektura, tj. izrada aplikacija kao skupova malih pojedinačnih servisa [73].

Poređenje monolitne i mikroservisne aplikacije najbolje ilustruje slika ispod (Slika 8). Levi deo slike ilustruje monolitnu aplikaciju i kao što se može zaključiti da sve funkcionalnosti koje aplikacija nudi se stavlja u jedan servis, i taj servis se skalira na *n* servisa. Sa druge strane (desni deo slike), mikroservisna aplikacija u svaki servis stavlja samo neophodnu funkcionalnost i ti servisi se skaliraju na određene servere tamo gde je potrebno [18].



Slika 8 - Poređenje monolitnih i mikroservisnih applikacija [18]

Korišćenjem mikroservisa, posle promene jednog servisa ne mora da se cela aplikacija ponovo kreira i postavi, dovoljno je samo taj pojedinačni mikroservis gde je logika izmenjena. Ovo dovodi do bržeg postavljanja aplikacije nezavisno od ostatka sistema. Ako se pojavi bilo kakav problem, on može da se izoluje u pojedinačnom mikroservisu i brže se vrši povratak u prethodno stanje. Ovo znači da klijenti koji koriste aplikaciju oslonjenu na mikroservise dobijaju znatno brži odziv, pošto klijent nije ni svestan potencijalnih problema u aplikaciji, jer ne dolazi do postavljanja cele aplikacije nego samo pojedinačnog mikroservisa [18][23].

4.1 Mikroservisna arhitektura

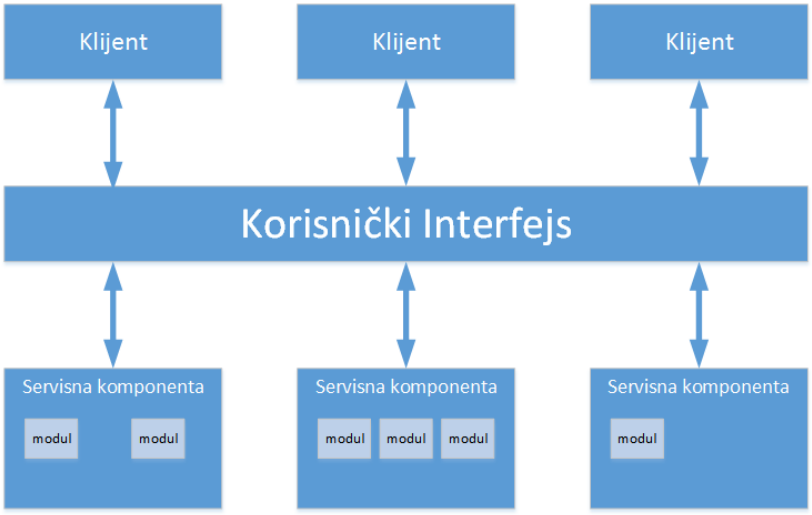
Najčešće prilikom razvoja softvera programeri ne koriste obrasce (eng. *pattern*) softverske arhitekture i tad dolazi do neorganizovanog koda koji nema jasnu ulogu i vremenom kod postaje sve teži i teži za održavanje. Komponente u aplikacijama koje su razvijene bez unapred planirane arhitekture, su vrlo često čvrsto povezane, i teške su za bilo kakvu promenu. U ovakvim aplikacijama je vrlo teško primeniti bilo kakav arhitektonski obrazac, bez razumevanja funkcionalnosti svih komponenti i modula sistema. Zato je potrebno da se koriste obrasci softverske arhitekture prilikom razvoja bilo koje aplikacije.

Softverska arhitektura podrazumeva skup odluka u organizaciji softvera. Odluke se odnose na kreiranje i dokumentovanje kako bi se razumeo sistem softvera. Svaka struktura se sastoji od elemenata softvera i njihovih međusobnih odnosa [74].

Arhitektonski obrasci pomažu u definisanju osnove za ponašanje neke aplikacije. Postoje više različitih tipova arhitektonskih obrasca, ali je za ovu doktorsku disertaciju bitna mikroservisna arhitektura.

Mikroservisna arhitektura [75] je evoluirala na razvoju monolitnih aplikacija koji koristi obrazac za razvijanje distribuiranih aplikacija odnosno slojevite arhitekture kroz SOA obrazac.

Mikroservisni arhitektonski obrazac [76] je obrazac koji se koristi kod aplikacija koji su izgrađene kao monolitne ili u servisno-orijentisanoj arhitekturi. Svaka servisna komponenta ima pojedinačne module odnosno mikroservise kao posebne jedinice, i na taj način omogućava bržu postavku cele aplikacije što dovodi do povećanja skalabilnosti i nivoa primenljivosti (Slika 9).



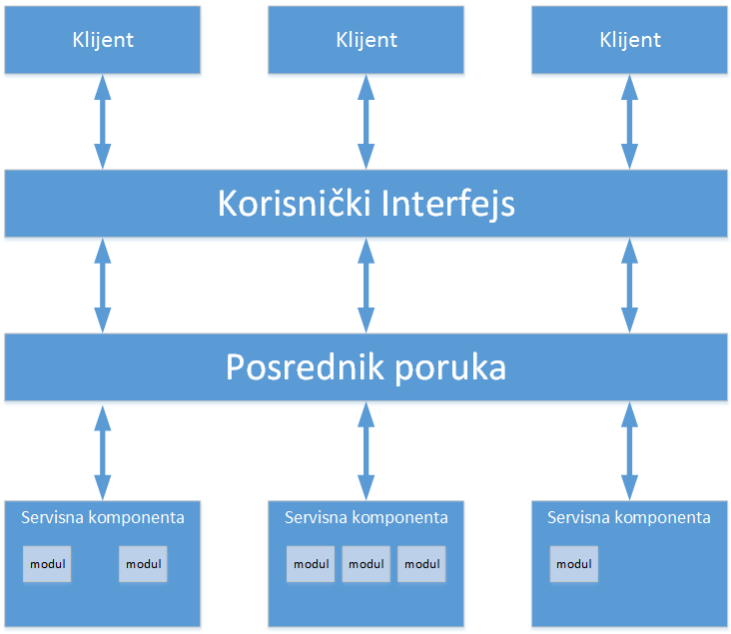
Slika 9 - Obrazac mikroservisne arhitekture [76]

Pre korišćenja mikroservisne arhitekture [70][77] dva koncepta su važna za razumevanje:

1. Svaka servisna komponenta (Slika 9) može se sastojati od jednog ili više modula. Moduli su mikroservisi koji imaju svoju biznis logiku i sama komunikacija između servisa se odvija preko određenih protokola. Projektovanje servisne komponente je najveći izazov prilikom izgradnje mikroservisne aplikacije.

2. Mikrosevisna arhitektura je ujedno i distribuirana arhitektura, što znači da su sve komponente odvojene i može im se pristupiti preko određenih protokola.

U mikroservisnoj arhitekturi postoje različite topologije razvoja aplikacije. Programeri biraju topologije u zavisnosti od svrhe aplikacije. U ovoj disertaciji, pošto je u pitanju sistem koji radi u realnom vremenu i gde je odziv od suštinskog značaja, izabrana je topologija centralizovanog slanja poruke (eng. *centralized messaging topology*), tako što se komunikacija između klijenta i servera vrši preko posrednika. Posrednik ima ulogu da rutira poruke primljene od klijenta ka serverskim komponentima, gde se vrši obrada zahteva (Slika 10). Ovom topologijom se može dobiti, kroz primenu asinhronih poruka, skalabilnost, *load balancing*, rukovanje greškama, itd.



Slika 10 - Mikroservisna arhitektura i topologija centralizovanog slanja poruke [76]

4.2 Karakteristike mikroservisa

U ovom poglavlju su date karakteristike mikroservisa koje su objašnjene od strane Martina Fowlera u [18]. Ove karakteristike može, a ne mora, svaki mikroservis da ima.

4.2.1 Servisi kao komponente

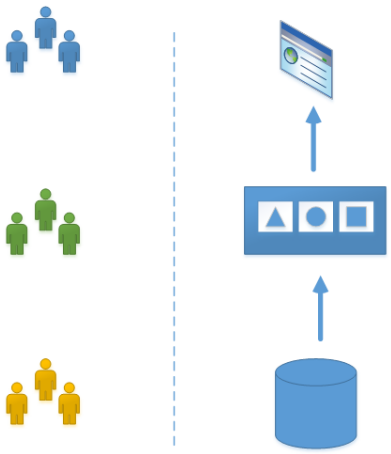
Komponente su jedinice softvera koje se nezavisno mogu zameniti i nadograditi. Jedan od glavnih razloga za korišćenje servisa kao komponente (eng. *componentization via services*) je to što se servisi mogu nezavisno postaviti na bilo koji server ili na bilo koju *cloud* platformu. Aplikacija koja sadrži više biblioteka u pojedinačnom servisu i svaka promena u bilo kojoj

komponenti dovodi do promene cele aplikacije u određenom serveru. Ali, ako je ta aplikacija transformisana u *n* servisa, samo taj izmenjeni servis može ponovo da se postavi na server što rezultuje minimizacijom postavljanja servisa aplikacije na određenom serveru. To nije apsolutno, neke promene će promeniti i interfejs (eng. *interface*) što rezultuje veće izmene u aplikaciji, ali cilj mikroservisa jeste da sve ovo minimizuje [18].

Prilikom podele servisa na više malih servisa, nailazi se na projektantski zadatak - kako i po kojim principima izgraditi servise. Korišćenje servisa kao komponente, daje veliku granularnost prilikom pravljenje plana izvršavanja i isporuke koji su značajni za bilo koje izdavanje nove verzije softvera kako bi klijent znao koja će nove funkcionalnosti dobiti. Sa monolitnim aplikacijama je svaka promena tražila ponovno postavljane cele aplikacije na server, dok sa mikroservisima to nije slučaj. Ovo dodatno ubrzava i pojednostavljuje plan isporuke jednog softvera [18].

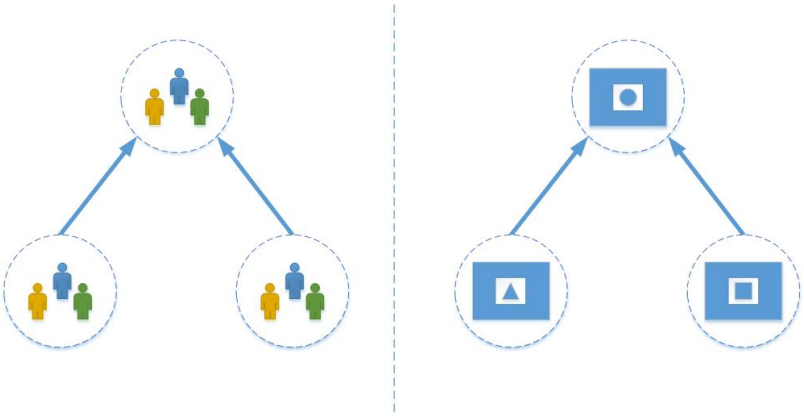
4.2.2 Organizacija poslovnih jedinica

Organizacija poslovnih jedinica (eng. *organized around business capabilities*) je jako važna u svim kompanijama. Najčešće prilikom razvoja određene aplikacije se timovi dele na tri tima: tim za kreiranje korisničkog interfejsa (UI - eng. *User Interface*), tim za kreiranje serverske strane i tim za kreiranje baze podataka. Kada su timovi ovako podeljeni svaka promena u aplikaciji mora biti ispraćena u svakom od tri tima. Jedan tim menja bazu podataka, zatim tim za serversku stranu radi podršku za tu izmenu i na kraju tim za UI obezbeđuje prikaz na samom interfejsu aplikacije (Slika 11) [18].



Slika 11 - Podela timova na osnovu specijalnosti [18]

Sa mikroservisima je proces izmene drugačiji od gore navedenog. Kada se aplikacija deli na *n* servisa omogućen je razvoj softvera sa više biznis procesa, odnosno tim može da sadrži različite profile kao što su za korisnički interfejs, za serversku stranu i za bazu podataka. Na ovaj način svaki tim je zadužen za svoj poslovnu logiku, a timovi međusobno sarađuju ako njihovi servisi komuniciraju (Slika 12) [18].



Slika 12 - Podela timova u mikroservisnom okruženju [18]

4.2.3 Razvoj proizvoda

Današnje kompanije su organizovane da razvijaju softver kao proizvod (eng. *product not projects*). To znači da se aplikacija završava po biznis kriterijumima, daje korisniku na korišćenje i projektni tim se raspušta. Mikroservisi imaju tendenciju da ne koriste ovaj model, nego da se aplikacija razvija kao proizvod. Ovo znači da tim ima odgovornost za softver u produkciji, što dovodi do međusobne komunikacije inženjera koji je radio na razvoju aplikacije i samih klijenata koji omogućava dugotrajnu i bolju komunikaciju između klijenata i inženjera. Inženjeri direktno skupljaju zahtevi od krajnjih korisnika i na taj način dobijaju veću svest o tome kako klijent koristi razvijenu aplikaciju, koje su njene mane i prednosti i dobijaju nove ideje kako da preduprede nedostatke aplikacije [18].

4.2.4 Mikroservisna komunikacija

Većina kompanija za svoje aplikacije bira napredne mehanizme za komunikaciju između samih mikroservisa kao što je *Service bus* koji nudi rutiranje, primena pojedinačnih poslovnih pravila, itd. Mikroservisi imaju tendenciju za biranjem komunikacionih protokola koji distribuira samo podatke između pojedinačnih komponenti tj. mikroservisnu komunikaciju (eng. *smart endpoints and dumb pipes*) [18].

Najveći izazov prilikom transformacije sa monolitne na mikroservise jeste sama komunikacija između mikroservisa. Što znači da aplikacije koje se izgrađene od mikroservisa koriste protokole za komunikaciju između ostalih mikroservisa, ali svaki mikroservis čuva svoj domen i biznis logiku za sebe i ne prikazuje je drugim mikroservisima [18].

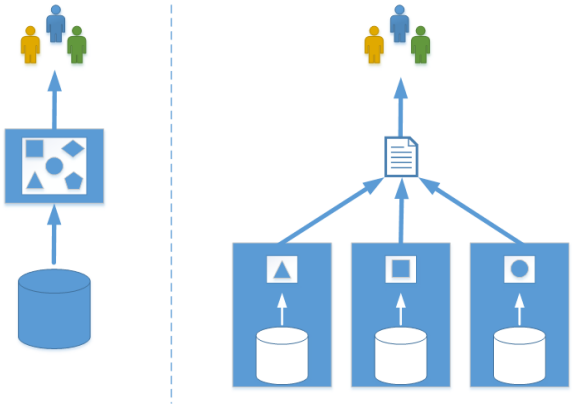
Mikroservisna komunikacija može da bude i preko srednjeg sloja tj. *Publisher/Subscriber* komunikacije [78][79] koja može da sadrži svoju sopstvenu bazu podataka [80].

4.2.5 Decentralizacija

Decentralizacija je ključna u mikroservisima, što omogućuje korišćenje različitih programskih jezika u različitim mikroservisima, svaki mikroservis skladišti svoje podatke i sam mikroservis je vlasnik tih podataka, dok monolitne aplikacije centrališu svoje podatke [18].

Kao što je gore pomenuto da mikroservisi decentrališu skladištenje podataka, znači da svaki servis upravlja sa svojom bazom podataka. Sa donje slike (Slika 13) se može videti

ilustracije centralizacije kod monolitnih, dok kod mikroservisa svaki mikroservis koristi svoju bazu podataka [18].



Slika 13 - Poređenje monolitnih i mikroservisnih aplikacija sa korišćenjem baze podataka [18]

Decentralizacija (eng. *decentralized governance*) ima dodatnu odgovornost jer preko mikroservisa ima implikacije za upravljanje upisom u baza podataka. Uobičajeni pristup koji se bavi upisom jeste transakcija koji garantuje doslednost prilikom ažuriranja više resursa. Ovaj pristup se često koristi u okviru monolitne aplikacije [18].

Korišćenje transakcija poput ove pomaže u doslednosti, ali postoji i problem, a to su da distribuirane transakcije teške za implementaciju i kao posledica toga mikroservisna arhitektura naglašava koordinaciju između mikroservisa koja mora da se implementira. Ali bez distribuiranih transakcija nema ni efikasnih distribuiranih sistema [18].

4.2.6 Automatizacija

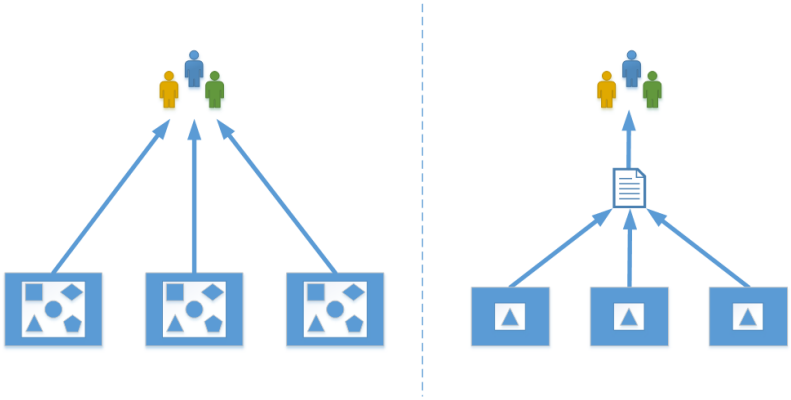
Razvojem *cloud*-a automatizacija (eng. *infrastructure automation*) je dostigla visok nivo i veliku primenu koja pomaže aplikacijama za lakše postavljanje, nadogradnju i nadgledanje aplikacija [18].

Za razvoj aplikacije bazirane na mikroservisima treba obratiti pažnju i razmisliti o kontinualnoj isporuci (eng. *continous delivery*) aplikacije klijentu. Definicija kontinualne isporuke je izrada softvera na takav način da se može bilo kog momenta pustiti u produkciji. Ovo znači da svaka aplikacija prilikom njenog razvoja treba da ispoštuje sve faze ciklusa softvera, a to su [18]:

- *Unit* testiranje,
- *Acceptance* testovi,
- Funkcionalno testiranje,
- Testiranje performansi, i
- Regresivno testiranje.

Razlika između monolitne i mikroservisne aplikacije, jeste što kod monolitnih više modula je stavljeno u jedan servis dok kod mikroservisa nije to slučaj što dovodi do lakše automatizacije

i lakši prolazak kroz faze nabrajane gore (Slika 14). Na ovaj način svaka razvijena aplikacija je spremna za korišćenje u produkciji i dovodi do manjih grešaka [18].



Slika 14 - Upoređenje monolitne i mikroservisne aplikacije [18]

4.2.7 Otpornost na greške

Jedna od važnih osobina mikroservisa i njihovog korišćenja jeste otpornost na greške (eng. *design for failure*). Na svaki pad servisa ili bilo koju grešku treba što pre reagovati kako klijent koji koristi aplikaciju ne bi bio blokiran. Ovo je jedna od mana koju mikroservisi imaju za razliku od monolitnih aplikacija. Posledica jeste što prilikom razvoja bilo koje mikroservisne aplikacije treba da se razmisli kako da se dizajnira i implementira da greške ne utiču na korisnike. Pošto servisi mogu da otkažu u bilo kom periodu, potrebno je uvrstiti sistem za praćenje (eng. *monitoring*) aplikacije u realnom vremenu [18].

Ovo je veoma važno za mikroservise, jer se na ovaj način može reagovati na bilo koju grešku u bilo koje vreme. Praćenje je od suštinskog značaja za brzo reagovanje na kvar. Sa druge strane ne mora na ovo da se gleda kao neku manu, pošto nekim sistemima, naročito sistemima u realnom vremenu je potrebno praćenje i stalni nadzor, tako da sa ovom karakteristikom neki sistemi, kao što su sistemi razmatrani u ovoj disertaciji, mogu imati samo dobitak.

4.3 Prednosti i mane mikroservisa

Prilikom razvoja i korišćenja mikroservisa [18][81] mogu se izdvojiti sledeće prednosti:

1. **Podela servisa** na *n* mikroservisa je najveća i najvažnija prednost mikroservisa. Ova podela omogućava svakom pojedinačnom servisu da radi nezavisno jedan od drugog. U svakoj aplikaciji postoje različite poslovne logike i tek kad softver raste i kad se dodaje još poslovne logike pojavljuje se potreba za podelu na *n* mikroservisa. Primer ove prednosti jeste perzistencija podataka, gde svaki mikroservis koristi svoju bazu podataka opisana u poglavlju 4.2.5 gde je jasno definisano da svaki mikroservis koristi svoju bazu podataka i da sama komunikacija između mikroservisa se odvija pomoću određenog protokola. Ovaj način decentralizacije isključuje integracione baze podataka koja ima mogućnost da više mikroservisa koriste jednu istu bazu podataka.

2. **Postavljanje aplikacije** na određeni *cloud* server se može reći da je druga u nizu od prednosti mikroservisa. Ključna stvar jeste da se svaki mikroservis može

nezavisno postaviti na bilo koji server. Ovo je jako važno kad dolazi do otkaza nekog mikroservisa ili prilikom određenog testiranja. Kako bi aplikacija mogla nesmetano da radi, postavljanje pojedinačnog mikroservisa na *cloud* bez prekida rada ostatka aplikacije je od velikog značaja. Kad jedan mikroservis prestane sa radom, tu je drugi da preuzme njegovu ulogu. Sa ovakvim pristupom, kao što je i gore objašnjeno, mikroservisi omogućavaju kontinualnu isporuku proizvoda koja smanjuje vreme između novih zahteva i dostavljanje klijentima novi implementirani zahtev. Ovo znači da klijent može da dobije funkcionalnost po funkcionalnost, a da pritom to ne utiče na ceo sistem i aplikaciju. Iz ovog razloga kompanije sve više koriste mikroservise, jer na taj način brže reaguju na tržište i brže mogu da odgovore klijentima na zahteve [82].

3. **Raznolikost tehnologije** je treća u nizu prednosti mikroservisa. Ova prednost omogućava da se u svakom pojedinačnom mikroservisu koristi neka druga tehnologija. U monolitnim aplikacijama kad je bilo potrebe za ovakvom izmenom, odnosno za korišćenjem drugog programskog jezika ili tehnologije, to je uticalo na celu aplikaciju i na celi sistem. Kompanije teže ka tome da apsorbuju što više različitih tehnologija pa samim tim biraju mikroservise zbog ove prednosti.

4. **Izolacija**. Prilikom pada određenog mikroservisa, važno je da to ne utiče na pad celog sistema. U ovom slučaju problem može da se izoluje i da se sistem neprekidno koristi. Kod monolitnih servisa su postojale višestruke mašine koje su obezbeđivale da sistem ne padne, dok sa mikroservisima to nije slučaj. Ova prednost se po Martinu zove elastičnost sistema [18].

5. **Kapsulacija**. Sposobnost da svaki mikroservis skriva detalje implementacije od drugih je sledeća u nizu od prednosti mikroservisa. Svaki mikroservis treba da bude odgovoran za svoj deo posla i ne treba da bude svestan implementacije koje se nalazi u drugim mikroservisima. Primer ove prednosti jeste i baza podataka pošto svaki mikroservis koristi svoju bazu podataka.

6. **Skalibilnost**. Monolitni servisi moraju da se skaliraju zajedno, pošto bilo kakva promena u monolitnim aplikacijama zahteva skaliranje celokupnog sistema. Za razliku od monolitnih, kod mikroservisnih aplikacija, skaliranje se odvija tako što se skaliraju samo oni koji trebaju, što znači da u svakom pojedinačnom mikroservisu se skalira onoliko instanci koliko je u tom trenutku potrebno. Kasnije se broj tih instanci može menjati u zavisnosti od trenutne potrebe.

7. **Monitoring**. Praćenje sistema je vrlo važno, kako bi se pratio rad mikroservisa, odnosno rad celokupnog sistema je još jedna u nizu prednosti mikroservisa. Takođe, logovi i statistike su vrlo važni prilikom pronalaženja problema kako bi se našao uzrok i utvrdio način rešavanja.

8. **Zamenljivost**. Zamena jednog mikroservisa sa drugim koji nudi istu funkcionalnost je veoma značajna prednost mikroservisa. Rešenja koja se baziraju na mikroservisima treba da imaju mogućnost podrške da se jedan mikroservis zameni sa drugim. Ova prednost je veoma česta u kompanijama, što je ranije bio slučaj sa monolitnim aplikacijama.

Prilikom razvoja mikroservisa Martin Fowler [18] je odvojio par mana koje se odvajaju, a to su:

1. Mikroservisi koriste distribuirane sisteme koji poboljšavaju modularnost sistema. U distribuiranim sistemima se mora obratiti pažnja na performanse, jer je sam distribuiran sistem kompleksan, a veliki problem su udaljeni pozivi između mikroservisa. Što je više takvih poziva veći je problem sa performansama.

2. Asinhrona komunikacija, pošto ako postoje više asinhronih poziva u isto vreme, to dodatno usporava praćenje rada aplikacija, ali ovo ujedno je i mana distribuiranih aplikacija, ali se IT svet suočava sa ovim iz dana u dan i sa zahtevima prilikom razvoja.

3. Mikroservisi donose decentralizaciju, a sama decentralizacija zahteva konzistentnost podataka prilikom bilo kojeg ažuriranja baze podataka, odnosno, sve izmene se vrše kroz jednu transakciju pa treba voditi računa o tome prilikom razvoja mikroservisne aplikacije.

4. Broj mikroservisa u jednom sistemu raste kako raste i sama aplikacija, odnosno, poslovna logika. Neophodna operativna složenost i upravljanje svim tim servisima i njihovo praćenje, implicira formiranjem još jednog tima ljudi koji će sve ove operacije obavljati.

Slika 15 - Prednosti i mane mikroservisa

5. PRIMENA MIKROSERVISA

Cloud computing [83][84] omogućuje fleksibilan, lokacijski nezavisan pristup računarskim resursima koji se veoma brzo dodaju i oduzimaju u skladu sa potražnjom odnosno korišćenjem. Računarski resursi se virtualizuju i korisnicima se daju u vidu usluga. Naplaćivanje resursa se vrši u odnosu na korišćenje resursa. *Cloud computing* [85] je distribuirana tehnologija za koju su vezani problemi zaštite podataka i sigurnosti podataka. Prednosti korišćenja *cloud computing*-a su:

- Neograničenost resursa - platforma pruža mogućnost korisnicima da iskoriste pristup neograničenim resursima. Korisnik ne brine o nedostatku resursa ili nedostatku skladištenja.

- Minimizacija infrastrukturnog rizika - upotrebom *cloud*-a se smanjuje rizik da se neka oprema kupuje i posle ne može više da se vrati, naprotiv omogućava lako širenje i smanjivanje u zavisnosti od potreba aplikacije.

- Skalabilnost - laka proširivost količine resursa potrebnih za izvršavanje klijentske aplikacije u cilju opsluživanja što većeg broja korisnika ili uklanjanje nepotrebnih resursa u slučaju manjeg broja korisnika.

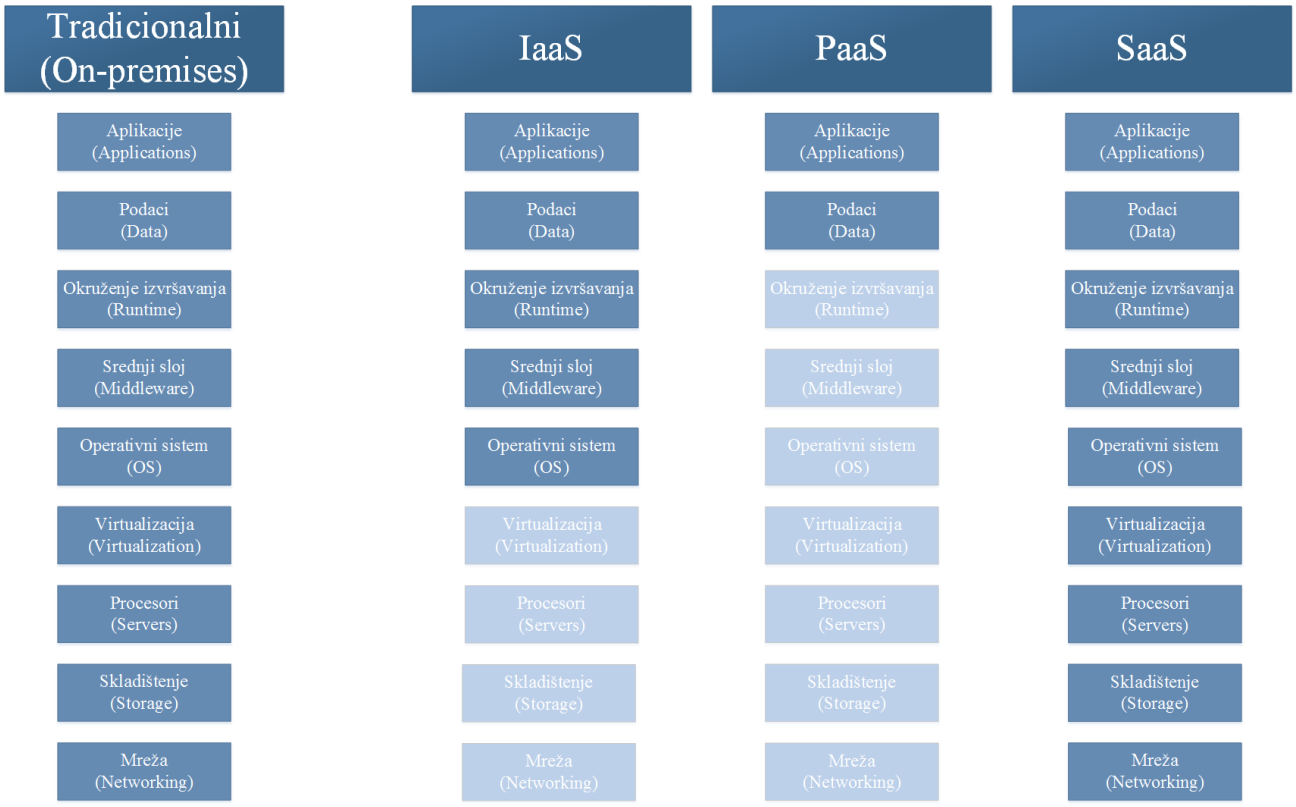
- Plaćanje samo onog što se koristi (eng. *Pay as you go model*) za razliku od tradicionalnih rešenja za rad sa *cloud*-om nisu potrebni nikakvi početni troškovi i zahvaljujući skalabilnosti platforme može se postignuti maksimalan stepen iskorišćenosti plaćenog hardvera u zavisnosti od potražnje.

Danas u *cloud computing*-u se mogu razlikovati tri modela sa aspekta odgovornosti koje pruža [86][87][88], za razliku od tradicionalnih gde su kompanije bile odgovorne za sve (Slika 16):

1. Infrastruktura kao servis (eng. *Infrastructure as a service* - IaaS). Procesiranje, skladištenje, mreža i druge računarske resurse se iznajmljuju od provajdera, a korisnici modela (npr. kompanije) na to mogu postaviti softver, gde mogu birati operativni sistem i aplikaciju. Klijenti ne upravljaju *cloud* infrastrukturom, ali upravljaju srednjim slojem i operativnim sistemom.

2. Platforma kao servis (eng. *Platform as a service* - PaaS). Korisnik modela (npr. kompanije) postavlja podatke i samu aplikacije, a sve ostalo kao što su mreža, skladištenje, server, operativni sistemi i drugi, se iznajmljuje od strane provajdera. U ovom modelu klijenti ne upravljaju *cloud* infrastrukturom.

3. Softer kao servis (eng. *Software as a service* - SaaS). Korisnik modela (npr. kompanije) koristi *cloud* aplikacije koje se izvršavaju nad *cloud* infrastrukturom. U ovom modelu klijent ne upravlja niti *cloud* infrastrukturom niti aplikacijom i podacima. Jedino što je potrebno da klijent podesi su uglavnom neke klijentske konfiguracije.



Slika 16 - *Cloud* modeli [86]

Primena mikroservisa jeste u *cloud* okruženju, korišćenjem platformi koje nam daju velike gigantske multinacionalne kompanije kao što su Google, Amazon Web Services (AWS), Microsoft, IBM i drugi.

U ovoj doktorskoj disertaciji će se koristiti jedna platforma, a to je: *Microsoft* Azure *Service fabric* (u daljem tekstu: *Service Fabric*), ali će biti dodatno obrađena i platforma *Amazon Web Services*. Obe platforme će biti objašnjene u narednim poglavljima.

5.1 *Microsoft Azure Service Fabric*

U ovom poglavlju i narednim će biti opisan *Service Fabric* [89][90], i većina opisa data u ovom odeljku je preuzeto iz radova o *Service Fabric*-u [89][90].

Service Fabric [90] je platforma za distribuirane sisteme koja se koristi za izvršavanje skalabilnih, pouzdanih i lako održivih aplikacija za *cloud* okruženje. Zadatak *Service Fabric*-a jeste razvoj aplikacija, kao i upravljanje tim aplikacijama u *cloud* okruženju. *Service Fabric* omogućava izgradnju i upravljanje skalabilnim i pouzdanih aplikacijama koje se baziraju na mikroservisima pokrenutim na deljenim mašinama odnosno *Service Fabric* klasteru.

Service Fabric klaster predstavlja skup određenih čvorova koji mogu biti fizičke ili skup virtualnih mašina koje su povezane međusobno mrežom. Čvor je adresa na nekom klasteru, koja se može dodati i skloniti iz klastera. Uzimajući za primer, *Azure SQL* klaster koji je pokrenut na *Service Fabric*, se sastoji od nekoliko stotina mašina pokrenutih sa deset hiljada kontejnera koje *host*-uju ukupno stotine hiljada baza (svaka baza je *Service Fabric stateful* mikroservis). Drugim rečima, *Service Fabric* klaster je tehnologija koja grupiše virtualne ili fizičke instance operativnih sistema na zajedničkoj mašini deljenih resursa. Takođe, pruža okruženje za izgradnju skalabilnih *steteless* i *stateful* mikroservisa i aplikacija za postavljanje (eng. *deploy*), praćenje (eng. *monitoring*), unapređenje (eng. *upgrading*) istih odnosno *patch*-ovanje i brisanje postavljenih aplikacija.

Današnje *Internet scale* aplikacije su napravljene koristeći mikroservise [90]. *Service Fabric* je mikroservisna platforma koja svakom mikroservisu, koji može biti ili *stateless* ili *stateful*, daje jedinstveno ime.

U nastavku su navedene osnovne prednosti koje donosi korišćenje *Service Fabric* platforme:

- *Service Fabric* pruža izuzetan brz *failover*. Za svaki mikroservice u *Service Fabric* (primarni mikroservis) se, po potrebi, kreira njegova replika (sekundarni mikroservis). Ako primarni padne, sekundarni će biti proglašen za primarni i preuzima posao.

- Skalabilnost mikroservisa: Mikroservisi se mogu brzo i lako povećavati za nekoliko hiljada instanci i posle smanjiti za nekoliko instanci u zavisnosti od toga koji su resursi potrebni. *Service Fabric* može da se iskoristi za kreiranje i upravljanje životnim ciklusom ovih skalabilnih *cloud* mikroservisa.

- Korišćenje *stateful* aplikacije omogućuje brzo pisanje i čitanje. Na primer, sistem za upravljanje distributivnim sistemima (DMS) koje rade u realnom vremenu - ovde je vreme čitanja i pisanja od suštinskog značaja, kako bi korisnik dobio brzu interakciju i reagovao na probleme.

- *Stateful* aplikacije pružaju mogućnost pamćenja sesije što je značajno za korisnike koji koriste DMS sistem, jer prilikom rada dolazi do skoka sa jedne funkcije na druge, pa uvek postoji potreba da se vrati na prethodnu.

- Distribuirana obrada grafikona što dovodi do brzog skaliranja i paralelnog procesuiranja opterećenja.

- Brzo čitanje i pisanje *Service Fabric*-a omogućava aplikacijama pouzdani tok podataka. Takođe omogućava aplikacijama da opišu distribuirano procesiranje, gde rezultati moraju biti pouzdani i bez gubitaka, npr. transakcije u distribuiranim sistemima, gde je značajno održati konzistentan sistem, naročito u DMS sistemu koji upravlja elektroenergetskom mrežom.

5.1.1 *Stateless* i *stateful*

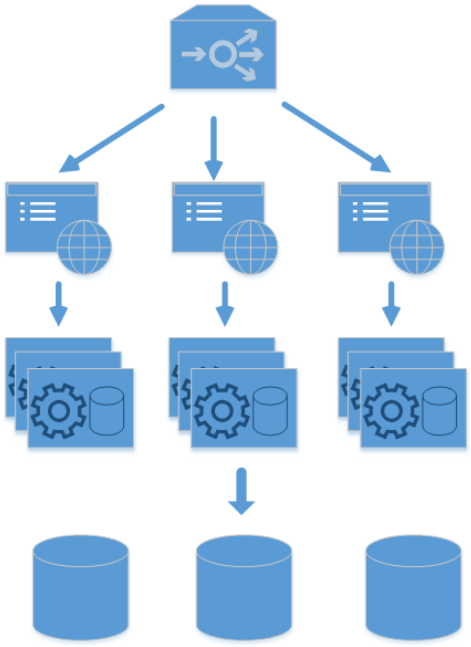
Stateless mikroservisi su mikroservisi koji ne vode računa o svom stanju. *Worker role*-a u Azure je primer takvog mikroservisa.

Stateful mikroservisi, za razliku od *stateless*, prate stanje mikroservisa. Primer takvih mikroservisa su: korisnički nalozi, baze podataka, redovi, bankovni računi, itd. *Stateful* mikroservisi su od suštinskog značaja, pošto se takvi mikroservisi koriste za izgradnju: mikroservisa gde je potreban visok protok (eng. *high-throughput*) i nisko vreme odziva (eng. *low-latency*), *Internet of things* sistema, sistema za detekciju prevara, itd.

Aplikacije koje su dizajnirane kao *stateful* mikroservis, nemaju potrebu za korišćenjem dodatnih funkcionalnosti kao što su redovi (eng. *queues*), dodatnih bafera (eng. *caches*) korišćenim u *stateless* aplikacijama.

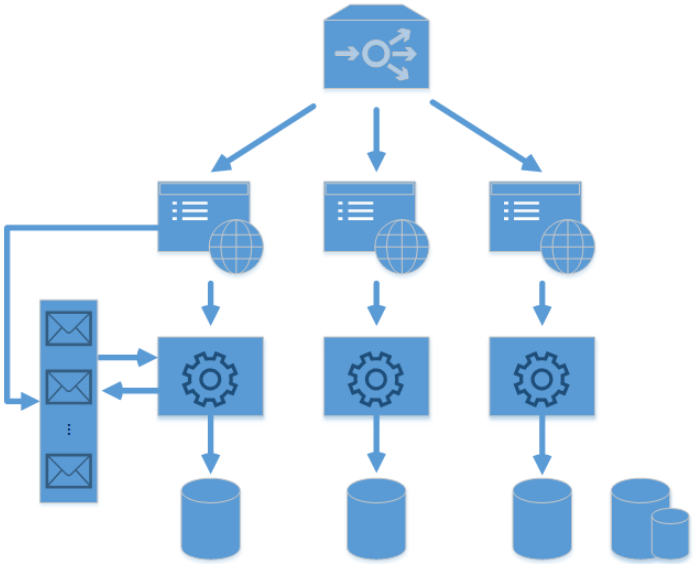
Da bi objasnili razliku u kreiranju aplikacija izgrađenih od *stateless* i *stateful* mikroservisa uzećemo u obzir aplikaciju čija se arhitektura sastoji od gornjeg sloja aplikacije koja je *stateless web* aplikacija, povezane sa srednjim slojem aplikacije koji je izgrađen od *stateless* i *stateful* mikroservisa postavljenih u *Service Fabric* klaster. Svaki od ovih mikroservisa je nezavisan i omogućuje skaliranje, pouzdanost i upotrebu resursa koji poboljšavaju agilnost u razvoju i upravljanju životnim ciklusom aplikacije. Na slikama ispod prikazane razlike između ova dva tipa mikroservisa u *Service Fabric* (Slika 17 i Slika 18).

Na slici ispod (Slika 17) je prikazana *stateful* aplikacija. Aplikacija se sastoji kao što je i gore objašnjeno od *stateless web* aplikacije (gornji sloj), a u srednjem sloju se nalaze *stateful* mikroservisi koji svaki mikroservis koristi bazu podataka za skladištenje podataka. Takva aplikacija nam donosi brzo čitanje i pisanje, stanje aplikacije je uvek dostupno korišćenjem *Reliable Services* i *Reliable Actors* i ovakav vid aplikacije donosi particionisanje mikroservisa koje će isto biti objašnjeno u poglavlju 5.1.4.



Slika 17 - *Stateful* aplikacija [90]

Slika 18 prikazuje primer *stateless* aplikacija. Za razliku od *stateful* ovde se koristi redovi i baferi u samoj arhitekturi. Ovakva aplikacija donosi: skaliranje sa particionisanjem skladištenja podataka, povećanje pouzdanosti korišćenjem redova i smanjenje vremena odziva čitanja upotrebom bafera.



Slika 18 - *Stateless* aplikacija [90]

5.1.2 *Reliable* kolekcije

Reliable kolekcije [92] su kolekcije koje dolaze sa *Service Fabric* i obezbeđuju visoku dostupnost i skalabilnost prilikom izrade *cloud* aplikacija. Ove kolekcije se najčešće koriste u *stateful* mikroservisima, tamo gde je potrebno da se stanje mikroservisa sačuva. Koristi se iz razloga što se kolekcije repliciraju na ostale sekundarne mikroservise istog tipa. Primera radi: ako jedan mikroservis sadrži jednu primarnu instancu i dve sekundarne instance, prilikom bilo kog upisa u *Reliable* kolekcije u okviru samog mikroservisa, automatski API (eng. *Application Programming Interface*) to replicira i na sekundarne instance. Ovo se radi iz razloga ako jedan nod padne odnosno primarni mikroservis padne, da bi ostali sekundarni mikroservisi bili svesni stanja mikroservisa i nastavili sa radom, tamo gde je primarna instanca stala.

Ako se vrši paralela običnih *.net* kolekcija koje omogućavaju rad u jednoj niti sa konkuretnim kolekcijama koje omogućavaju višestruke niti, dobijaju se *Reliable* kolekcije (Slika 19) [92].

Slika 19 - Upoređivanje kolekcija [92]

Reliable kolekcije pružaju sledeće [92]:

- Replikaciju - stanje mikroservisa se replicira na ostale instance kako bi obezbedili visoku dostupnost,
- Asinhroni API koje *Reliable* kolekcije pružaju - nit nije blokirana prilikom bilo kog upisa,

- Transakcije - u okviru jednog mikroservisa se može upravljati sa više *Reliable* kolekcija istovremeno koji obezbeđuje transakcioni upis, i

- Perzistenciju - svaki se podatak perzistira na mikroservis, ako npr. dolazi do nestanka električne energije u nekom *data center*-u, da bi se podaci ne bi izgubili.

Reliable kolekcije pružaju snažne garancije konzistentnosti. Doslednost se postiže tako što se transakcija završi tek nakon što se celokupna transakcija evidentira na svim replikama, čak i na primarnoj instanci mikroservisa. Da bi se doslednost postigla, aplikacije mogu da zatraže nazad odgovor od klijenta pre nego što se asinhroni poziv završava [92].

Postoje tri tipa *Reliable* kolekcija, a to su [92]:

1. *Reliable dictionary* - predstavlja repliciranu, transakcionu i asinhronu kolekciju parova ključ/vrednost.

2. *Reliable queue* - predstavlja replicirani, transakcioni i asinhroni red koji je FIFO.

3. *Reliable concurent queue* - predstavlja replicirani, transakcioni i asinhroni red koji je FIFO i služi za za visoku propusnost.

Reliable kolekcije se koriste u sledećim slučajevima kada [92]:

- Stanje treba da bude dostupno sa malim vremenom odziva,

- Mikroservis treba da kontroliše konkurenciju ili granularnost transakcionih operacija preko jedne ili više *Reliable dictionary*,

- Je potrebno kontrolisati particionisane šeme svakog mikroservisa,

- Nije potrebno voditi računa o konkurentnosti između niti,

- Aplikacija treba dinamički da kreira ili uništava *Reliable dictionary* ili *Reliable queues* u toku njihovog izvršavanja,

- Ima potrebe za kontrolom *Service Fabric*-a koji obezbeđuje rezervu i *restore* funkcije za stanje mikroservisa,

- Aplikacija treba da čuva istoriju promena svog stanja mikroservisa, i

- Potrebno je razvijati ili koristiti *state provider*-e nekog trećeg lica.

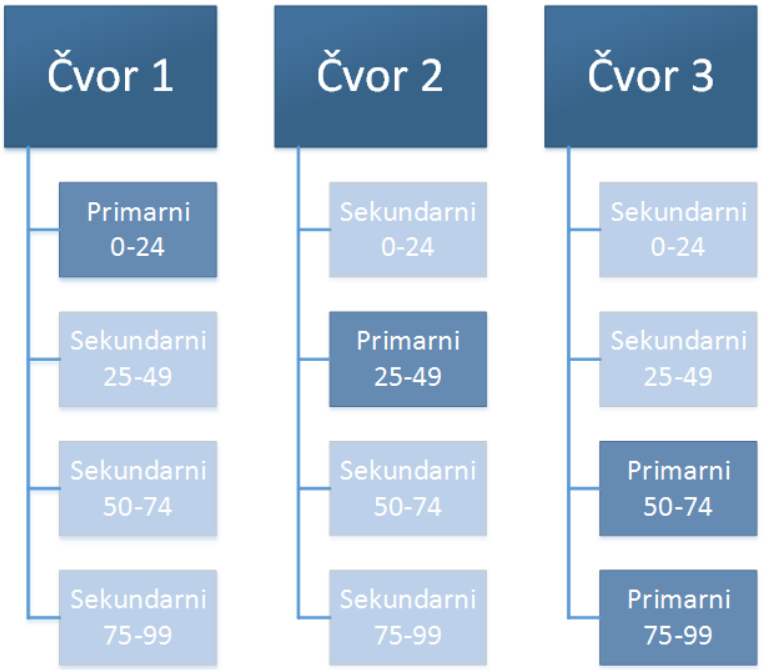
5.1.3 Skaliranje

Service Fabric daje mogućnost za izgradnju skalabilnih aplikacija, particionisanje i kreiranje njihovih replika na čvorovima klastera, koji daje potpunu iskorišćenost resursa.

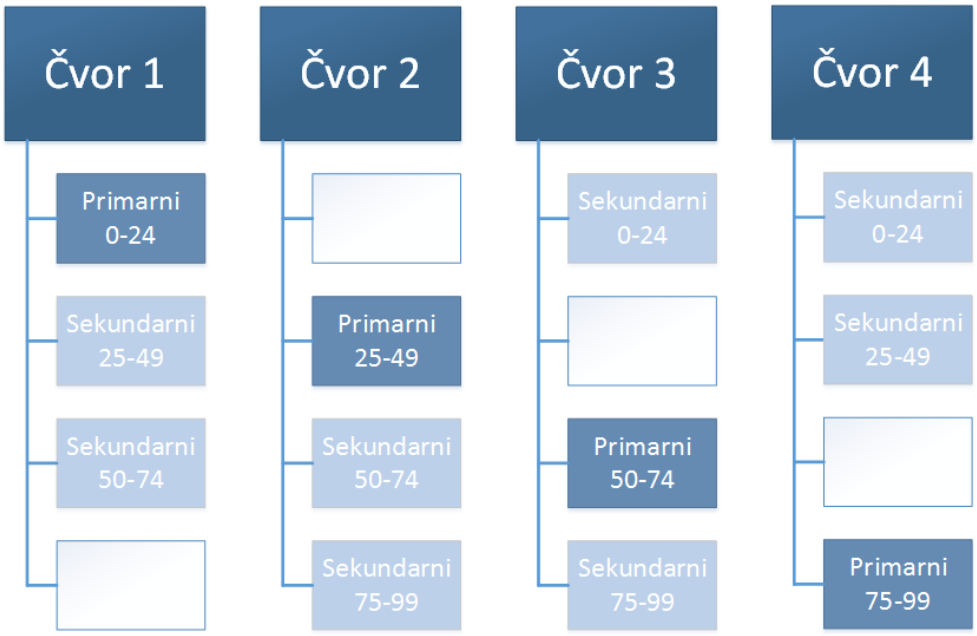
Skalabilnost u *Service Fabric* se može postići na dva načina:

1. Skaliranje na nivou particije - svaki mikroservis koji je postavljen na *Service Fabric* može da se particioniše na više particija. Ako uzimamo za primer mikroservise gde

particije sa brojevima od 0 do 99, od kojih svaki ima 4 particije koje treba da se postave na klaster sa 3 čvora. Mikroservisi će biti raspoređeni na 3 čvora, za koji svaki ima dve replike tako da se svi resursi iskoriste (Slika 20). Ako se tokom rada poveća broj čvorova na *Service Fabric*-u, prazni resursi se popunjavaju praznim kopijama na svim čvorovima. Primera radi, ako se sad u primeru sa slike ispod (Slika 20) dodaje još jedan čvor, dobija se situacija prikazana na slici ispod (Slika 21), na svakom čvoru će biti tri replike što daje bolju iskorišćenost resursa.



Slika 20 - Primer skaliranja servisa na 3 čvora [90]



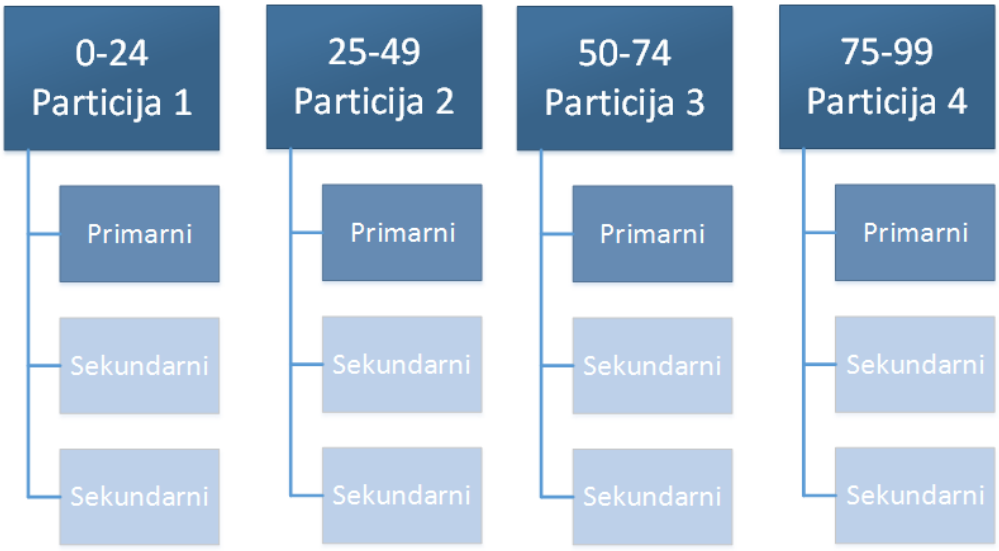
Slika 21 - Primer skaliranja servisa na 4 čvora [90]

2. Skaliranje na nivou imena servisa – prilikom kreiranja servisa određuje se particija koja će se koristiti. Prvi nivo skaliranja je po imenu servisa, koji daje mogućnost kreiranja novih instanci mikroservisa, sa različitim nivoima particionisanja, ako su stari mikroservisi zauzeti, što dovodi do toga da se manje koriste zauzeti mikroservisi. Jedna opcija je da se povećava kapacitet ili da se povećava ili smanjuje broj particionisanja prilikom kreiranja novih instanci mikroservisa sa novom particijom, ali tad korisnik servisa mora da zna kada i kako se koriste različita imena mikroservisa.

5.1.4 Particionisanje

Particionisanje *Service Fabric* aplikacija se odvija preko *stateful* mikroservisa i svaka particija radi sa podskupom svih raspoloživih stanja. Svaka particija je generisana da bude visoko dostupna, a replike se distribuiranju preko čvorova na klasteru.

Particionisanje se radi preko šeme koja je prikazana na slici ispod (Slika 22). Uzimajući primer 4 particije sa opsegom od 0 do 99 i 4 mikroservisa sa 2 dve replike. Zajednički pristup je da se kreira niz jedinstvenih ključeva na osnovu zadatih ključeva u datom skupu. Primer jedinstvenih ključeva mogu biti jedinstveni matični broj građana, registarska tablica automobila, jednistveni broj radnika u jednoj firmi, itd. Da bi se dobio jedinstveni ključ može da se generiše *hash code*. Takođe, može da se definiše donja i gornja granica dozvoljenog ključa.



Slika 22 - Primer particionisanja [90]

5.2 *Amazon Web Services*

Amazon Web Services (AWS) je *cloud* platforma koja nudi preko 175 različitih servisa u različitim *data center*-ima. Nudi pouzdane, skalabilne i jeftine usluge za *cloud* okruženje. Servisi mogu biti za skladištenje podataka, sistem poruka, sistem plaćanja, baze podataka i za razvoj mikroserervisa [93].

Kao što je gore pomenuto, AWS ima integrisani lanac blokova koji podržava razvoj mikroservisa. Dva najpopularnija pristupa su *AWS Lambda* i *docker* kontejneri u *AWS Fargate*.

5.2.1 *AWS Lambda*

Koristeći *AWS Lambda*, kod se postavlja na *Lambda* servis i sam servis se brine za pokretanje aplikacije i skaliranje koda sa velikom dostupnošću. *AWS Lamba* podržava nekoliko programskih jezika i može da se poziva iz bilo kod AWS servisa ili direktno sa neke *web* ili mobilne aplikacije. Jedna od najvećih prednosti *AWS Lambda* jeste što programer koji koristi može da se fokusira na biznis logiku, jer za bezbednost i skaliranje brine i upravlja AWS [94].

Prednosti koje *AWS Lambd*a pruža su [94]:

1. Nema servera za upravljanje. *AWS Lambda* automatski pokreće kod bez potrebe da se upravlja infrastrukturom. Na programerima je samo da napišu kod, o sve ostalom brine *AWS Lambda*.

2. Kontinualno skaliranje. *AWS Lambda* automatski svaku postavljenu aplikaciju skalira. Kod se izvršava u paraleli, svaki proces se okida individualno i skalira se prema veličini radnog okruženja.

3. Merenje ispod sekunde. *AWS Lambda* naplaćuje svakih 100ms kad se kod izvršava i pokreće. Drugim rečima, plaća se samo vreme koje se potroši.

4. Performanse. Sa *AWS Lambda* može da se optimizuje vreme izvršavanja koda odabirom odgovarajuće veličine memorije za svaku funkciju.

Tok izvršavanja sa postavljanjem koda na *AWS Lambda* može se videti na slici ispod (Slika 23). Najpre se kod postavlja na *AWS Lambda*, zatim se podešava i namešta komunikacija sa ostalim AWS servisima, pokreće se *AWS Lambda* i na kraju kao finalno naplaćuju se samo iskorišćeni sati.

Slika 23 - Tok izvršavanja koda na *AWS Lambda* [94]

5.2.2 *AWS Fargate*

Zajednički pristup za smanjenjem operativnih stvari za postavljanje (eng. *deployment*) su kontejneri. Kontejnerske tehnologije poput *Docker* su poslednjih nekoliko godina veoma popularne po osobinama kao što su: prenosivost, produktivnost i efikasnost. *Amazon Elastic Container Service* (AECS) i *Amazon Elastic Kubernetes Service* (AEKS) eliminišu potrebu za instaliranjem, nadgledanje, i skaliranje infrastrukture klastera [95]. AECS je kontejener koji služi za orkestraciju servisa, dok AEKS je kontejener za orkestraciju *Kurbenets* (*open source* za postavljanje i upravljanje kontejnerima aplikacije). Kada se koristi AECS ili AEKS koristi se *AWS Fargate.*

AWS Fargate [96] je servis koji služi za kontejnere, gde se može pristupiti i pokretati kontejnere bez servera, bez razmišljanja o obezbeđivanju, konfigurisanju i skaliranju klastera virtualnih mašina gde se klasteri izvršavaju.

AWS Fargate na početku alocira dovoljan broj klastera i time eliminiše da se dinamički alociraju dodatni resursi tokom rada aplikacije. Ovo se postiže inicijalizacijom većeg broja instanci i skaliranjem kapaciteta klastera i to što je najbitnije plaća se samo ono što se koristi, odnosno što se izvršava u kontejnerima, a ne ono šta se alocira [95].

Prednosti koje *AWS Fargate* pruža su [95][96]:

1. Postavljanje i upravljanje aplikacijama, a ne infrastrukturom. Sa *AWS Fargate* se može usredsrediti na izgradnju i upravljanje aplikacijama bez obzira da li se izvršavaju na AECS ili AEKS. Samo iskorišćeni kontejneri se plaćaju, izbegavajući operativne troškove skaliranja i upravljanje serverima. *AWS Fargate* obezbeđuje da je infrastruktura na kojim se kontejnerima aplikacija izvršava uvek ažurirana potrebnim zakrpama (eng. *patches*).

2. Bezbednost. Svaka aplikacija koja se izvršava da li na AECS ili AEKS se izvršava na pojedinačnom serveru koji ne deli CPU, memoriju ili mrežu. Što znači da je svaka aplikacija bezbedna.

3. Ravnomerno raspoređivanje resursa sa fleksibilnim opcijama za plaćanje. *AWS Fargate* pokreće i skalira izračunavanje kako bi usko odgovarao zahtevima za resursima koje se navedu za kontejner. Sa *AWS Fargate*-om nema rezervisanja i plaćanja dodatnih servera.

4. Bogata uočljivost aplikacija. Koristeći *AWS Fargate* dobija se bolja komunikacija i povezanost sa drugim AWS servisima. Omogućava skupljanje metrika i logova za praćenje aplikacija kroz raznih alata.

Tok izvršavanja sa postavljanjem koda na *AWS Fargate* može se videti na slici ispod (Slika 24). Najpre se izgrađuju kontejneri, zatim se definiše memorija i svi ostali neophodni resurni na klasteru, nakon toga aplikacija se izvršava i može da se upravlja, a na kraju se plaća samo ono šta je iskorišćeno [95].

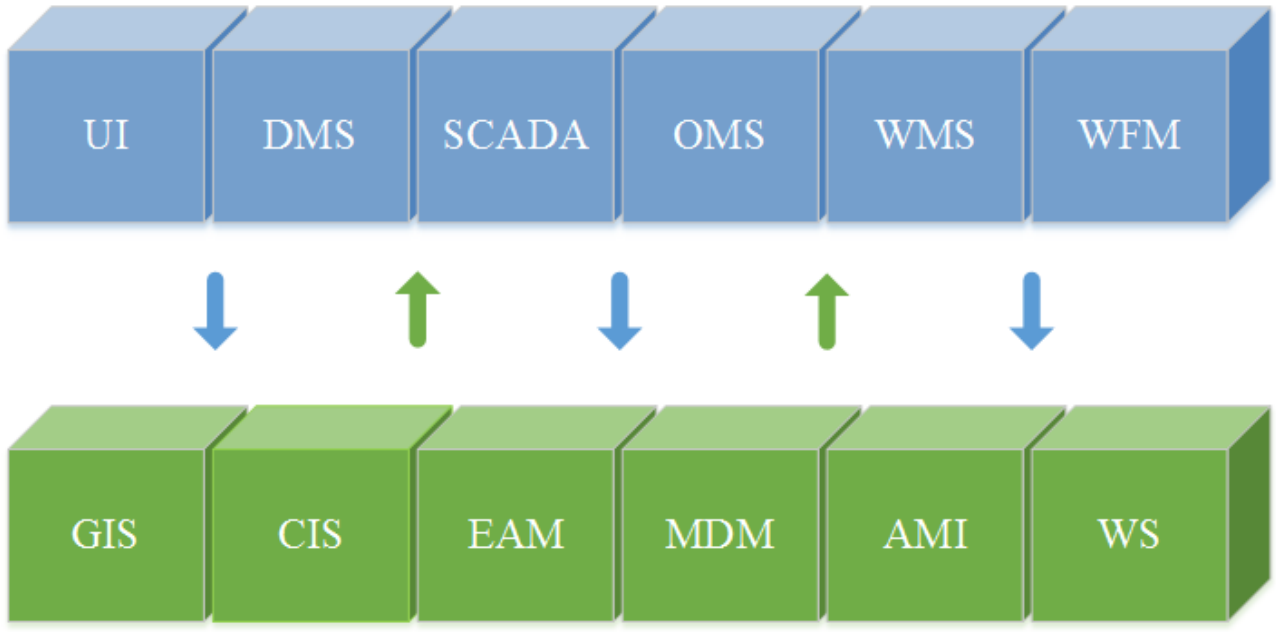
Slika 24 - Tok izvršavanja aplikacija na *AWS Fargate* [95]

6. TRADICIONALNI DMS SISTEM

Cilj ovog istraživanja je analiza i transformacija DMS sistema [32][33][97] sa monolitne na mikroservisnu aplikaciju. Da bi se prvo razumeo tradicionalni DMS sistem, mora da se shvata pojam pametnih distributivnih mreža (eng. *smart grid*) čiji je DMS sistem deo toga.

6.1 Pametne distributivne mreže

U današnje vreme sve više dolazi do povećanja broja uređaja koji se koriste u distributivnim sistemima, kao što su sistemi za upravljanje, nadzor, kontrolu, pametne kuće, punjače električnih vozila i mnogi drugi. Sve ove uređaje treba spojiti u jedan jedinstveni sistem kako bi dobili što bolju efikasnost i korišćenje svih tih uređaja, i taj sistem zove se pametna distributivna mreža [97][98]. Ako gledamo sliku ispod (Slika 25) mogu se videti svi značajniji sistemi pametne distributivne mreže [31].



Slika 25 - Komponente pametnih distibutivnih mreža

Jedna takva arhitektura se može podeliti na više sistema [31]. Kao što se može videte sa slike iznad (Slika 25) sistemi se mogu podeliti i na dva sloja. Gornji sloj su sistemi za upravljanje i planiranje pogona i radova primarne distributivne mreže u realnom vremenu, a donji sloj za dopunske poslove i skladištenje podataka. To su sledeći sistemi [31]:

1. **UI/Web** prikazuje korisnicima datu mrežu. Kroz dati interfejs korisnici koriste sve date funkcije jedne takve aplikacije.

2. **DMS** predstavlja centralni sistem koji služi za nadzor elektroenergetske mreže. Suština ovog sistema je estimacija stanja na nivou cele mreže, topološka analiza mreže, kao i proračun upravljačkih akcija.

3. **SCADA** je sistem koji radi na komunikaciji sa poljem i svodi se na prikupljanje podataka u realnom vremenu od RTU-eva (eng. *Remote Terminal Unit*) i koristi za nadzor i upravljanje telemetrisanim tačkama mreže u realnom vremenu.

4. **OMS** (eng. *Outage Management System*) je sistem koji služi za detekciju, analizu i upravljanje, svim planiranim i neplaniram ispadima u elektroenergetskoj mreži. Takođe, služi i za određivanje problema koje se dešavaju u polju, a nisu neki ispadi. Neke od mogućih primena gde se ovaj sistem koristi su prilikom vremenskih nepogoda: oluja, tornada, uragana, požara i ostale vremenske nepogode. OMS sistem se koristi i u delu koji je pod SCADA sistemom i delom koji nije. Skoro svako distributivno preduzeće ima implementiran ovaj sistem.

5. **WMS** (eng. *Work Management System*) je sistem koji služi za vođenje planiranih i neplaniranih radova u elektroenergetskoj mreži, pri čemu radovi najčešće nisu neki ispadi, nego planirani radovi koji se svakodnedno dešavaju u jednoj elektroenergetskoj mreži kao što su: provera postrojenja, instalacija brojila, izgradnja nove ulice i priključenje novih potrošača na nova brojila, i drugi. Distributivna preduzeća koja imaju dosta planiranih radova najčešće teže korišćenju ovakvog sistema.

6. **WFM** (eng. *Work Force Management*) je sistem koji služi za čuvanje podataka o svim posadama i pojedincima koji rade u polju na nekom određenom radu. Distributivna preduzeća koji imaju velike radove teže za jednim ovakvim sistemom kako bi bolje isplanirali svoj svakodnevni rad.

7. **GIS** (eng. *Geographic Information System*) je osnovni sistem koji služi kao osnovni izvor podataka o elektrodistributivnoj mreži. Ovaj sistem sadrži podatke o svakoj lokaciji, faznoj konektivnosti i pripadnosti kao i druge podatke o samim uređajima. Svako distributivno preduzeće treba da ima implementiran ovaj sistem.

8. **CIS** (eng. *Customer Information System*) je sistem koji sadrži podatke o individualnim potrošačima, odnosno korisnicima elektroenergetske mreže. Svaki novi priključak se prvo beleži u ovaj sistem, tako da ovaj sistem sadrži sve privatne podatke potrošača. Svaki distributivno preduzeće mora da ima implementiran ovaj sistem.

9. **EAM** (eng. *Enterprise Asset Management*) predstavlja sistem koji sadrži podatke o svakom elementu (bilo da je električni ili ne) koji je deo elektrodistributivne mreže. Svaki novi uređaj se prvo uvodi u ovaj sistem kako bi imali jasnu sliku o broju svih elemenata koji postoje u jednom distributivnom preduzeću. Podaci iz ovog sistema se posle koriste prilikom izvođenja određenih radova, jer na taj način postoji jasan pregled svake opreme koje se koristi. Velika većina distributivnih preduzeća ima ovaj sistem.

10. **MDM** (eng. *Meter Data Management*) je sistem koji služi za skladištenje podataka o individualnoj potrošnji električne energije. Ovaj sistem nije baš zastupljen u distributivnim preduzećima.

11. **AMI** (eng. *Advanced Metering Infrastructure*) predstavlja sistem koji služi za daljinsko očitavanje potrošnje električne energije kod invidualnih potrošača. AMI se sastoji od pametnih brojila, koncentratora podataka i softverskog modula. U funkciji daljinskog očitavanja, AMI se u preduzećima razlikuju po vremenskom horizontu. Što znači da se podaci mogu očitavati u intervalima i to na nivou od 24 sata, 1 sat, 15 minuta, 5 minuta, ili 1 minut. Ovakvo prikupljenje podatka se zove kvazi-telemetrisani, što znači da vremenski horizont ne odgovara realnom vremenu kao kod SCADA sistema. Vrlo mali broj distributivnih preduzeća ima implementiran AMI na nivou cele elektroenergetske mreže.

12. **Sistemi vremenske prognoze** (eng. *Weather systems*) predstavlja sistem koji služi za prikupljanje trenutne vremenske prognoze, ali takođe sistem ima kapacitet da odredi prognozirane vremenske prilike. U distributvnim preduzećima je jako značajno da budu svesni poznavanje vremenske prognoze, naročito prognoze oluja, jer od toga zavisi potrošnja/proizvodnja i kompletan proces preventivnog upravljanja elektroenergetskom mrežom. Uglavnom velika preduzeća imaju implementirane ove sisteme.

U ovoj doktorskoj disertaciji, težište i sam fokus je na sam DMS sistem pošto on se u njemu izvršavaju proračuni (opisani u poglavlju 3.2) od značaja za koje će se utvrditi analiza.

6.2 DMS sistem

Ako pogledamo još dublje šta jedan DMS sistem ima i čemu on služi to je mnogo više proračuna u odnosu na one opisane u poglavlju 3. Arhitektura jednog takvog sistema je prikazana ispod (Slika 26).

Na slici ispod (Slika 26) se može videti uprošćena arhitektura jednog DMS sistema, sa sledećim komponentima koji su od značaja za ovaj rad:

- **Klijent** - je korisnički interfejs, koji je namenjen kljentima u kontrolnoj sobi koji vrše svakodnevni rad kao i neko buduće planiranje nad jednom elektroenergetskom mrežom. Mogu se uočiti sledeće neophodne funkcionalnosti na klijentskoj strani:

    - prikaz elektroenergetske mreže sa raznih aspekata: šematski, geografski, prikaz elemenata u podstanicama, itd.,

    - manipulacija elementima nad elektroenergetskom mrežom. Prilikom manipulacije nad elementima, primera radi otvaranje i zatvaranje prekidača, proračuni topološka analiza i tokovi snaga se automatski pokreću, i

    - razne opcije za pokretanje navedenih proračuna objašnjenih u poglavlju 3. Klijent na samom interfejsu ima mogućnost da sam pokrene na osnovu zadatog korena sledeće proračune: tokovi snaga, putanja, estimacija stanja, procena mesta kvara, restauracija velikog područja, rekonfiguracija, prognoza, kratki spojevi, spremnost modela, Volt/Var optimizacija,

kapacitet prekidača i osigurača i relejna zaštita. Klijent ima razne opcije za pokretanje ovih funkcija, i za svaku od njih se zadaju ulazni parametri kao što je i opisano za svaku pojedinačnu funkciju u poglavlju 3. Takođe, za svaku pojedinačni proračun se korisniku omogućava da vidi rezultate proračune koji će biti objašnjeni u nastavku.

- **Servis za proračun** - u ovom servisu se vrši obrada trinaest proračuna, a to su tokovi snaga, putanja, estimacija stanja, kratki spojevi, procena mesta kvara, restauracija velikog produčja, rekonfiguracija, prognoza, Volt/Var optimizacija, kapacitet prekidača i osigurača, spremnost modela i relejna zaštita. Dobijanjem zahteva za jedan od ovih trinaest proračuna, sprema se model i trenutna topološka analiza tako što se očita iz TA servisa, kao što je u objašnjeno u poglavlju 3 da navedeni proračuni direktno zavise od topološke analize elektroenergetske mreže ali i od ovih proračuna koji se nalaze u samom servisu. Ako za dati proračun je potreban i rezultat drugog proračuna, to znači da unutar samog servisa se vrši poziv drugog proračuna odnosno servisa ako je to potrebno. Zatim, šalje se zahtev na *Host servisu* koji vrši obradu proračuna. Posle obrade proračuna zahtev se šalje nazad servisu za proračun odakle ide nazad klijentu i rezultati se prikazuju:

    - Za tokove snaga u vidu:

        - Prikaza rezultata na zahtev korisnika, za određeni deo elektroenergetske mreže,

        - Prikaz pored svakog elementa elektroenergetske mreže,

        - Prikaz izveštaja sa određenim i dodatnim sadržajem za sam proračun,

        - Skaliranje samo na određenog dela elektroeneretske mreže i time se vrši ušteda samog sistema.

    - Za putanju u vidu obeležavanja elemenata na elektroenergetskoj mreži koji se nalaze na traženoj putanji.

    - Za estimaciju stanja:

        - Prikaz izveštaja,

        - Prikaz pored svakog elementa na elektroenergetskoj mreži rezultat obrade tokove snaga,

        - Bojenje samih elemenata elektroenergetske mreže koji prikazuju trenutno stanje i korisniku sugeriše da na bolji način vidi prikaz cele elektroenergetske mreže. Na ovaj način korisnik ima grafički prikaz i može brže da reaguje na neke promene, i

        - Prikaz rezultata na zahtev korisnika, samo tamo gde je potrebno. Na ovaj način se servisi ne opterećuju dodatno.

    - Za kratke spojeve:

- Prikaz izveštaja za navedene parametre: struja kvara, napon kvara, napon na mestu kvara, struja na mestu kvara, vršna (eng. *peak*) struja, putanje kvara i dužina od izvora.

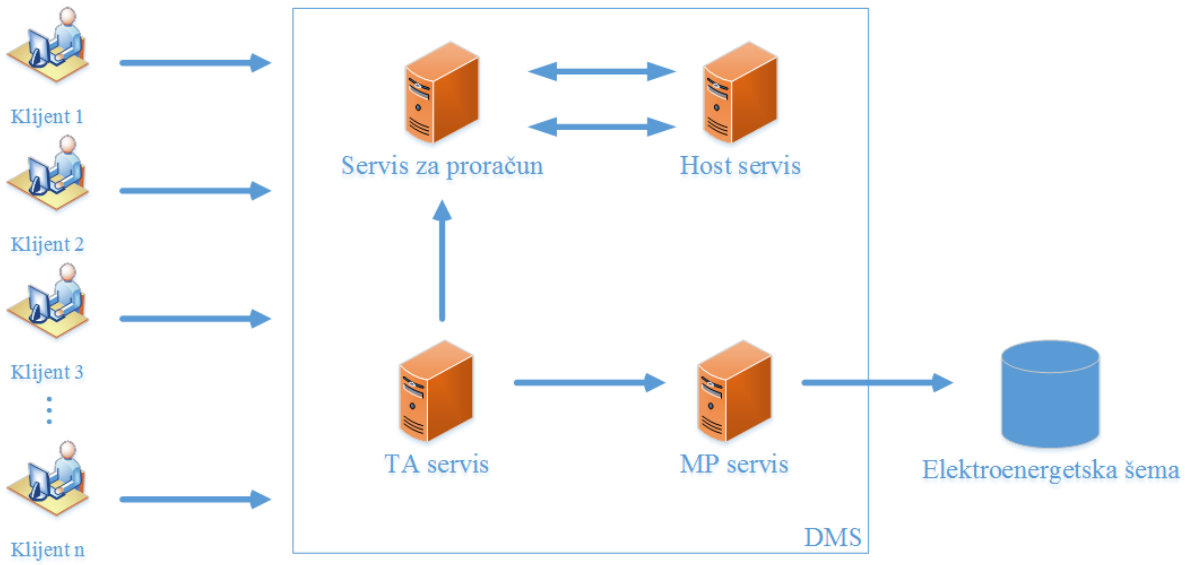
- Za procenu mesta kvara u vidu:
  - Prikaz izveštaja sa svim datim kvarovima,
  - Prikaz akcija koje treba preduzeti da bi se kvar otklonio, i
  - Prikaz kako bi se napanje vratilo nazad potrošačima.
- Za restauraciju velikog područja:
  - Izveštaj kako može elektroenergetska mreža ponovo da se napaja.
- Za rekonfiguraciju:
  - Prikaz akcija radi primene rekonfiguracije nad elektroenergetskoj mreži.
- Za prognozu u vidu:
  - Prikaza izveštaja prognoza opterećenja i proizvodnje za naredna 30 minuta, 8 dana, 52 nedelje i od 1 do 20 godina.
- Za spremnost modela u vidu:
  - Prikaza izveštaja za indeks spremnosti modela,
  - Prikaza izveštaja za kriterijum spremnosti modela, i
  - Prikaz izveštaja za kvalitet spremnosti modela.
- Za Volt/Var optimizaciju:
  - Prikaz izveštaja i rezultati optimizacije, i
  - Prikaz sekvence za optimizaciju.
- Za kapacitet prekidača i osigurača:
  - Prikaz izveštaja za svako stanje prekidača,
  - Prikaz izveštaja za svako dodavanje ili brisanje privremenih elemenata u eletroenergetskoj mreži, i
  - Za određeni vremenski interval.
- Za relejnu zaštitu u vidu:

- Prikaza izveštaja za sve kvarove u elektroenergetskoj mreži za ovaj tip proračuna,
- Prikaza izveštaja za sve kvarove u elektroenergetskoj mreži za one koje se nalaze u zoni relejne primarne zaštite, i
- Prikaza izveštaja za sve kvarove u elektroenergetskoj mreži za one koje se nalaze na vodu za odabrani *circuit*.

- **TA servis** - u ovom servisu se vrši obrada topološke analize elektroenergetske mreže. Takođe, i ovaj servis je u komunikaciji sa servisom modela podataka, jer sam proračun zavisi direktno od modela elektroenergetske mreže. Rezultati se prikazuju na elektroenergetskoj mreži tako što se graf boji kao što je objašnjeno u poglavlju 3.2.1. Ovaj proračun se nalazi u posebnom servisu za razliku od drugih proračun jer klijent može direktno da koristi rezultate topologe analize i da vidi obojenu elektroenergetsku mrežu sa topološkom analizom, bez rezultata drugih proračuna iz drugih servisa.

- **MP servis** - je servis koji je direktno zadužen za model podataka elektroenergetske mreže na osnovu CIM modela [99][100][101][102]. Model podataka se puni iz baze i interno se čuva i zadaje se svim učesnicima gde je to potrebno u transakciji.

- **Baza za elektroenergetsku mrežu** - u ovoj bazi se čuvaju svi podaci elektroenergetske mreže koji su potrebni samom klijentu za prikaza i servisima za određene proračune. Na osnovu ove baze se model popunjava u MP servisu i šalje dalje ostalim servisima za proračun.



Slika 26 - Arhitektura tradicionalnog DMS sistema

6.3 Opis problema

Osnovni problem koji se pokušava rešiti je transformacija jednog takvog - tradicionalnog DMS sistema (Slika 26) na sistem koji je razvijen u mikroservisnom okruženju. Razlozi transformacije na *cloud* i mikroservisno okruženje su sledeći:

- Održivost samog sistema. Sistem koji je u upotrebi godinama postaje iz dana u dan sve teži za održavanje. Klijenti koji koriste takav sistem sve teže ga koriste i imaju potrebu za promenu istog.

- Serveri koji se koriste prilikom postavljanja sistema iz dana u dan postaju sve slabiji, pa samim tim i ceo sistem postaje slabiji.

- Skalabilnost korišćenjem jednog od mikroservisnih okruženja daje veću mogućnost i širenja aplikacije.

- Klijent jednog dana ako želi da proširi svoj softver, mora i nove servere da kupuje, a možda su stari i neuopotrebljivi sa novijom arhitekturom, što znači da finansijski to treba ispratiti.

- Jedan takav tradicionalni sistem ne može da garantuje pouzdanost, a i da može teško ga je napraviti i često dolazi do neke greške.

Jedna takva transformacija mora imati ozbiljne promene u samoj arhitekturi, kako bi na kraju bila isplativa i samom klijentu i onome ko je pravi. Jako puno zavisi od trenutne arhitekture i od samih servisa kako su napisani, da li su SOA napisani ili nisu. Ako sama arhitektura nije SOA bazirana onda dolazi do dodatnih poteškoća prilikom transformacije. Primera radi, jedan servis po mikroservisnoj arhitekturi može da se podeli na male mikroservise. Cilj transformacije jeste da se svaka poslovna logika odvija u posebnom mikroservisu. Sa druge strane klijenti u ovim sistemima su klijenti koji nisu *web* orijentisani. Takvi sistemi sa takvim klijentima zahtevaju još veće promene prilikom transformacije. Tu postoje dva načina:

1. Klijent koji nije *web* orijentisan da ostane takav kakav jeste. U ovom slučaju dolazi do uštede i manjih promena u samoj implementaciji, i

2. Prebacivanjem klijenta na *web*. Ovo zahteva veliku transformaciju, odnosno pisanje jednog takvog klijenta od nule, ali vremenom ovo postaje isplativije i samo okruženje *cloud*-a će moći bolje da se iskoristi.

Velika je razlika između klijenta koji se nalazi u *cloud* na nekoj platformi i klijenta koji se postavlja na svim mašinama dostupnim za to. Za ovaj drugi slučaj dolazimo do gore pomenute transformacije i razloge zašto se to radi. To dovodi i do kupovine samih mašina gde se ti klijenti postavljaju i zamenjuju ako dolazi do nekog kvara. Takođe, ukoliko dođe do pada jedne mašine taj klijent je neupotrebljiv. Sa druge strane za prvi slučaj pomenut iznad sve je drugačije. Platforma brine o svemu, ako jedan klijent na nekom serveru padne postoji njegova replika, i svi učesnici mogu svog klijenta i dalje da koriste, kako ne bi došlo da zagušenja tokom korišćenja istih.

6.4 Hipoteze

Cilj i upotreba rezultata istraživanja sprovedenih u okviru ove doktorske disertacije se svodi na njeno korišćenje i isplativost za samog klijenta i onome ko formira sistem. S obzirom da je fokus ove disertacije na transformaciji jednog DMS sistema na mikroservisno okruženje, treba se oslanjati na to šta donosi mikroservisno okruženje i mikroservisna arhitektura. Iz svega ovog proizlazi i prva hipoteza:

H1: Moguće je razviti arhitekturu softverskog sistema za elektroenergetske proračune u mikroservisnom okruženju.

Cilj predložene arhitekture za mikroservisno okruženje jeste njena upotreba i primena u DMS sistemima, odnosno transformacija monolitne aplikacije u mikroservisnu aplikaciju. Ideja je da se takva monolitna aplikacija postavi na *cloud*, što dovodi do uštede vremena i resursa. Ovim je formulisana naredna hipoteza:

H2: Predloženo rešenje se može primeniti u DMS sistemima i dobiti transformacijom monolitne aplikacije na mikroservisnu aplikaciju.

Nakon primene mikroservisa i iskoristivosti svih prednosti same arhitekture i platforme, treba i dokazati njenu primenu u odnosu na monolitnu aplikaciju, što je ujedno i hipoteza same doktorske disertacije:

H3: Predloženo rešenje ima prednosti u odnosu na konvencionalna rešenja.

U sprovedenim eksperimentima su izvršena poređenja mikroservisnih i monolitnih rezultata i prikazane prednosti koje je donelo novo okruženje.

7. PREDLOG ARHITEKTURE SISTEMA I TRANSFORMACIJA PRORAČUNA

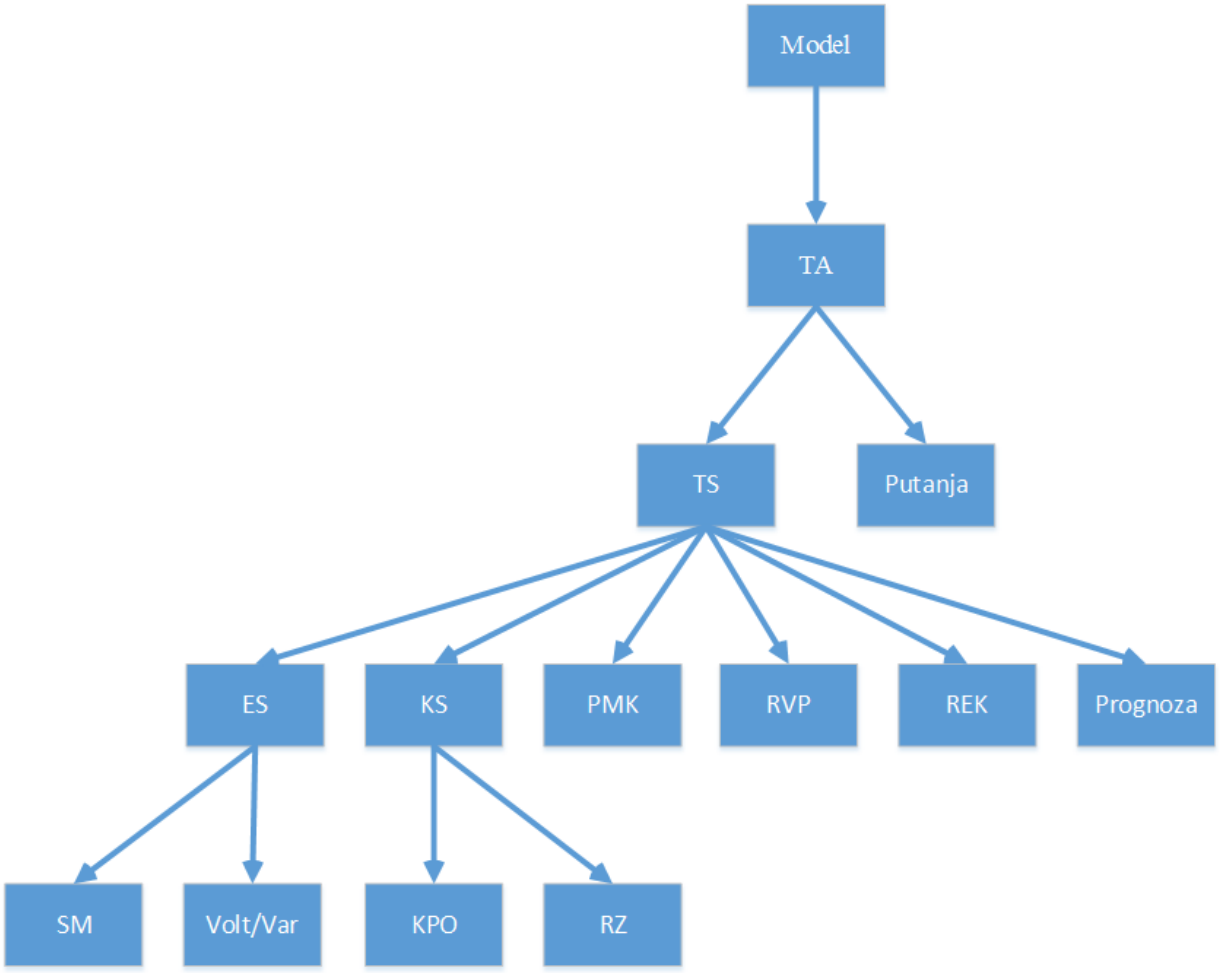
U ovom poglavlju će biti dat predlog transformacije DMS sistema opisanog u poglavlju 6 sa aspekta primene elektroenergetskih proračuna opisanih u poglavlju 3. Algoritmi optimalnog izvršavanja proračuna, kao i algoritmi za dodavanje i uklanjanje proračuna iz sistema, su opisano kako bi ukazalo na primenu arhitekture u sistemu. Takođe, na osnovu predložene arhitekture dat je predlog izmene samih proračuna i njihova primena u mikroservisnom okruženju na platformi *Service Fabric*.

7.1 Arhitektura sistema

Uzimajući u obzir elektroenergetske proračune opisane u poglavlju 3, u cilju razumevanja njihove veze i međusobne zavisnosti je prikazano na dijagramu zavisnosti, gde se mogu uočiti sledeće zavisnosti:

- TA zavisi od modela elektroenergetske mreže,
- TS zavisi i od modela elektroenergetske mreže i od TA,
- Putanja zavisi i od modela elektroenergetske mreže i od TA,
- ES zavisi od modela elektroenergetske mreže, TA i od TS,
- KS zavisi od modela elektroenergetske mreže, TA i od TS,
- PMK zavisi od modela elektroenergetske mreže, TA i od TS,
- RVP zavisi od modela elektroenergetske mreže, TA i od TS,
- REK zavisi od modela elektroenergetske mreže, TA i od TS,
- Prognoza zavisi od modela elektroenergetske mreže, TA i od TS,
- SM zavisi od modela elektroenergetske mreže, od TA, TS i ES,
- Volt/Var, zavisi od modela elektroenergetske mreže, od TA, TS i ES,
- KPO zavisi od modela elektroenergetske mreže, od TA, TS i KS, i

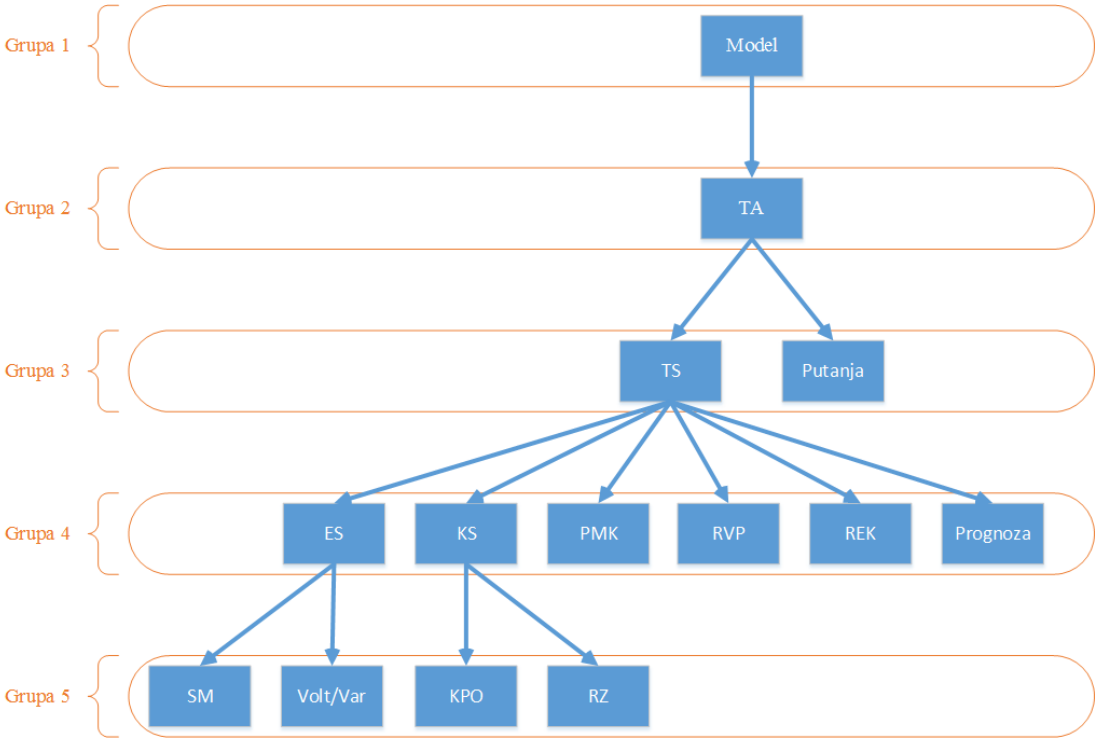
- RZ zavisi od modela elektroenergetske mreže, od TA, TS i KS.



Slika 27 - Dijagram zavisnosti elektroenergetskih proračuna

Nakon analize opisanih zavisnosti može se zaključiti da se pojedini proračuni nalaze na istom nivou dijagrama (imaju iste roditelje i potomke). Imajući to u vidu, oni mogu da se grupišu na osnovu zavisnosti u odnosu na druge proračune. Na taj način su napravljene grupe proračuna kao što je prikazano na slici ispod (Slika 28). Iz svega ovog se mogu definisati 5 grupa, koji sadrže sledeće:

1. Grupa 1 (G1) sadrži model eletroenergetske mreže.
2. Grupa 2 (G2) sadrži proračun TA.
3. Grupa 3 (G3) sadrži proračune: TS i Putanju.
4. Grupa 4 (G4) sadrži proračune: ES, KS, PMK, RVP, REK i Prognozu.
5. Grupa 5 (G5) sadrži sledeće proračune: SM, Volt/Var, KPO i RZ.



Slika 28 - Dijagram zavisnosti podeljen po grupama

Nakon utvrđivanja zavisnosti elektroenergetskih proračuna i njihovog grupisanja, a na osnovu arhitekture tradicionalnog DMS sistema (Slika 26) i uzimajući u obzir objašnjenu mikroservisnu arhitekturu i same mikroservise, u daljem tekstu je opisana predložena arhitektura DMS sistema u *cloud* okruženju (Slika 29). Arhitektura sistema (Slika 29) se sastoji od sledećih komponenti:

1. Klijenta,
2. PP (Prosleđivač Poruka) mikroservisa,
3. MP (Model Podataka) mikroservisa tj. G1 mikroservisa,
4. G2 mikroservisa,
5. G3 mikroservisa,
6. G4 mikroservisa, i
7. G5 mikroservisa.

Klijent je korisnički interfejs (UI) koji služi za prikaz elektroenergetske mreže i svih proračuna objašnjenih u poglavlju 3.2. Klijent na osnovu date elektroenergetske mreže šalje zahtev da se za određeni koren izvrši određeni proračun. Kao što može videti na slici ispod (Slika 29) sa klijenta se šalje zahtev u sistem koji se nalazi u *cloud* okruženju. Svaki od klijenata je u direktnoj komunikaciji sa PP mikroservisom što znači da se svaki poziv, zahtev i svaka interakcija klijenta upućuje ka PP mikroservisu. Na taj način, klijent koji se nalazi na svakom pojedinačnom računaru je u direktnoj komunikaciji sa privatnim *Load balancer*-om na *cloud*-u. Privatni *Load balancer* na *cloud*-u služi da rutira poziv PP mikroservisu, i on odlučuje kojoj

instanci PP mikro servisa će proslediti zahtev od klijenta. *Load balancer* se koristi baš u ovim slučajevima, kada je klijent van *cloud*-a, i to u ovakvim arhitekturama. *Load balancer* pruža klijentima IP (eng. *Internet Protocol*) adrese za koje se smatra da su privatne i za koje samo klijent zna kako bi uspostavio komunikaciju sa *cloud*-om.

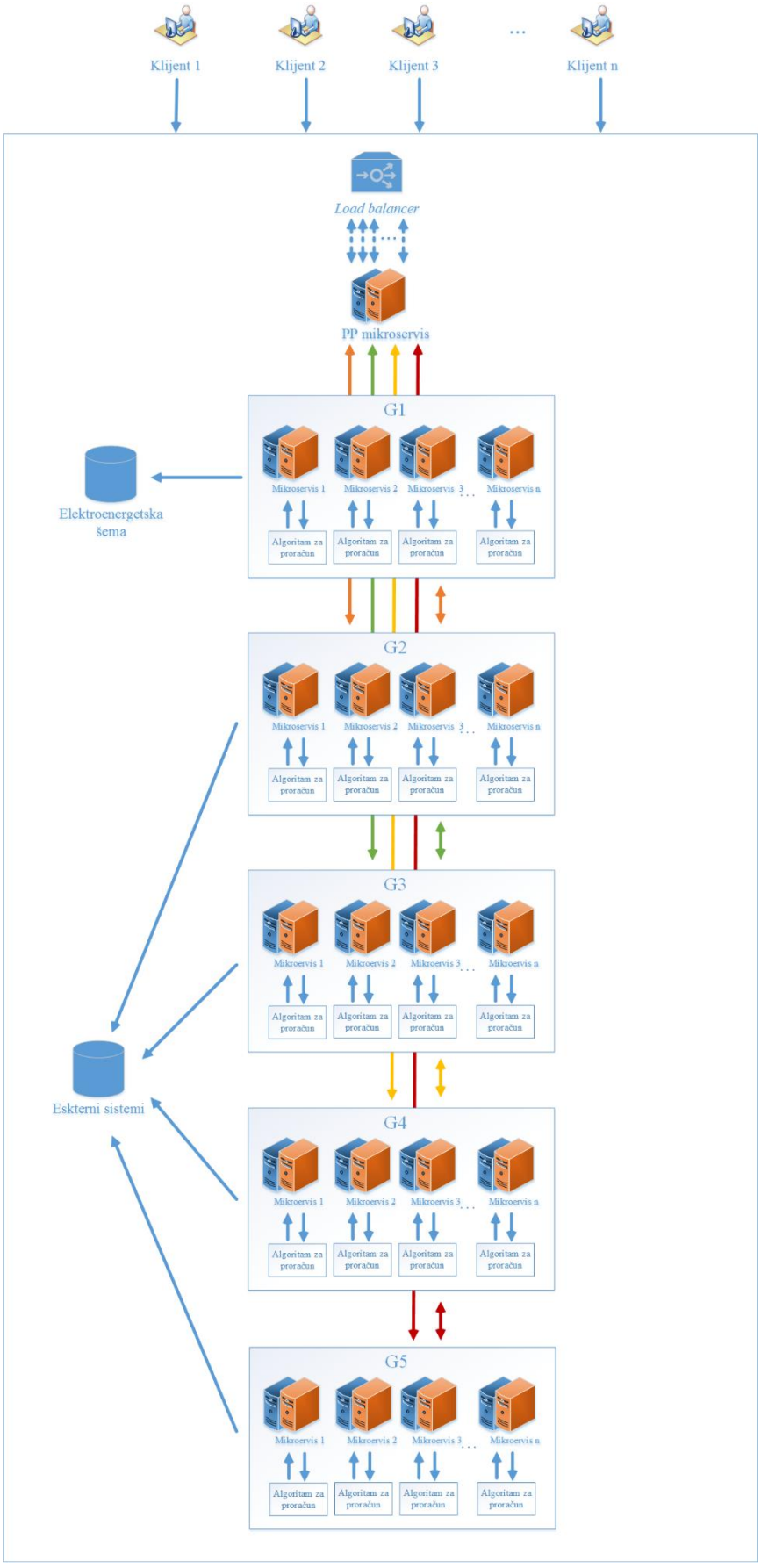
PP mikroservis je mikroservis koji je po mikroservisnoj arhitekturi objašnjen u poglavlju 4.1, mikroservis koji ne vrši nikakvu orkestraciju, transformaciju ili bili kakav zahtev. Služi samo za prosleđivanje odnosno rutianje poruka od klijenta do G2-G5 mikroservisa. Rutiranje poziva se vrši tako što za svaki koren PP mikroservis aktivira određenu instancu G2-G5 mikroservisa. Mikroservisna arhitektura koja sadrži mikroservis ovakvog tipa zove se topologija mikroservisnog slanja poruke [76]. Ovaj tip mikroservisne arhitekture [76] se koristi za velike aplikacije, a svi navedeni proračuni su sami po sebi značajni jer se koriste u velikim i kritičnim aplikacijama koji trebaju da obezbede dobru skalabilnost u softveru sistema za nadzor i upravljanje distributivne mreže. Ubacivanjem jednog takvog mikroservisa u sistem dobija se na ubrzanju i do 30 puta, kreirajući uvek asinhrone pozive koji održavaju distribuiranu transakciju samog sistema.

G1 mikroservisi (MP mikroservisi) su mikroservisi koji služe za čitanje elektroenergetske mreže datog sistema iz baze, i kreiranje modela koji je potreban za elektroenergetske proračuna. Prilikom kreiranja navedenog modela su potrebni sledeći podaci: grane, čvorovi, korene i informacije o svim elementima za datu elektroenergetsku mrežu. Svi ovi podaci se nalaze u bazi kojoj ima pristup jedino ovaj mikroservis. Pored toga, MP mikroservis vrši dodatnu raspodelu za koji deo mreže se vrši neki proračun. Naime, svi dati zahtevi od klijenta se obrađuju na nivou korena, s obzirom da klijent šalje zahtev za proračun za određene korene. U ovom mikroservisu, prilikom kreiranja datog modela, vrši se particionisanje pristiglih korena i raspodela korena kako bi svi mikroservisi u samoj arhitekturi bili iskorišćeni. Primera radi, ako treba da se izvrši proračun za tri korena, u MP mikroservisu će se kreirati tri modela, za svaki koren ponaosob i tri mikroservisa koja treba da obrađuju zahtev, a to su: TA, TS i ES mikroservisi sa svojim algoritmom i proračunima.

G2 mikroservisi su mikroservisi koji su namenjeni za proračun topološke analize elektroenergetske mreže. Na osnovu zahteva prosleđenog od PP mikroservisa, G2 kontaktira G1 mikroservis za dati model, i datog korena obrađuju se neažurni koreni elektroenergetske mreže, što znači da mikroservis pamti stanje topološke analize za već izračunat koren. Kada se za dati koren izračunava topološka analiza, rezultati se vraćaju nazad klijentu preko PP mikroservisa.

G3 mikroservisi su mikroservisi koji su namenjeni za proračune: tokove snaga i putanje. Mikroservisi su u direktnoj komunikaciji sa G2 mikroservisima, što znači da kad se zahtev dobija od PP mikroservisa, za G3 proračune je potreban i model topološke analize, odnosno, rezultat topološke analize, pa G3 mikroservis inicira G2 mikroservis po potrebi, a G2 mikroservis inicira G1 mikroservis za model eletroenergetske mreže. Kada G3 mikroservis ima sve rezultate i podatke potrebne za proračune, proračun može da se izvrši. Nakon završetka obrade, rezultati se vraćaju nazad klijentu preko PP mikroservisa.

G4 mikroservisi su mikroservisi koji su namenjeni za proračune: estimacija stanja, kratki spojevi, procene mesta kvara, restauracije velikog područja, rekonfiguracije i prognoze. Mikroservisi ove grupe su u direktnoj komunikaciji sa G3 mikroservisima. G4 mikroservis po potrebi inicira G3 mikroservis, pa G3 inicira G2 i G2 inicira G1 mikroservis, jer za date proračune su potrebni rezultati tokove snaga, topološke analize i modela elektroenergetske mreže. Kada svi



Slika 29 - Arhitektura sistema za predložene elektroenergetske proračune

ovi rezultati stižu u G4 mikroservisu, proračun može da se izvrši i nakon obrade rezultati se vraćaju nazad klijentu preko PP mikroservisa.

G5 mikroservisi su mikroservisi koji su namenjeni za proračune: spremnosti modela, Volt/Var optimizacije, kapaciteta prekidača i osigurača i relejne zaštite. Mikroservis dobija zahtev od PP mikroservisa za proračun za dati koren. Za date proračune su potrebni određeni proračuni topološke analize, tokova snaga i modela eletroenergetske mreže pa se zadati rezultati dobijaju respektivno od G3 mikroservisa, G2 mikroservisa i G1 mikroservisa. Takođe za proračune u ovim iz ove grupe potrebni su i određeni proračuni G4 mikroservisa. Za spremnost modela a i Volt/Var optimizacije je potreban proračun estimacije stanja, a za kapacitet prekidača i osigurača i relejne zaštite je potreban proračun kratkih spojeva, pa se rezultati dobijaju tako što se inicira poziv G4 mikroservisima. Kada se proračun završava rezultati se preko PP mikroservisa šalju nazad do klijenta.

Eksterni sistemi su podaci iz eksternog sistema koji služe u G2-G5 mikroservisima kako bi odredili proračuni. Primera radi u G4 mikrosrvisima se koristi u proračunu prognoze kako bi se dobili podaci o vremenskoj prognozi, geografskom položaju, demografiji i drugih podataka koji su potrebni za proračun koje je opisano u poglavlju 3.2.9.

Na osnovu predložene arhitekture može se reći da je hipoteza H1 zadovoljena, što znači da je arhitektura bazirana na mikroservisnoj arhitekturi objašnjenoj u poglavlju 4.1 i ima sledeće prednosti:

- Svaka poslovna logika je odvojena u poseban mikroservis.
- Svaki mikroservis komunicira sa ostalim mikroservisima preko određenih definisanih protokola.
- Svaka promena postojećeg mikroservisa u aplikaciji ne dovodi do postavljanja celog sistema na *cloud*, nego samo tog pojedinačnog mikroservisa.
- Sa ovakvom arhitekturom se može koristiti raznovrsnost tehnologija.
- Skrivanje detalja implementacije jednog mikroservisa od ostalih mikroservisa u sistemu.
- Skaliranje mikroservisa.

Predložena arhitektura je transformisana sa monolitne na mikroservisno okruženje i ima primenu u DMS sistemu što znači da hipoteza H2 je zadovoljena.

7.2 Analiza arhitekture sistema

Ako se analiziraju karakteristike svakog mikroservisa i arhitektura sa slike iznad (Slika 29) dobijaju se rezultati prikazani u tabeli ispod (Tabela 1). U tabeli je prikazano za svaku komponentu sa DA je opisano da je karakteristika primenjena, u suprotnom sa NE. Ako za neku komponentu neka karakteristika ne može da se primeni, primera radi za klijenta koji se ne nalazi u mikroservisnog okruženju, polje u tabeli se označava sa NP (Nije Podržano). Kao što je i Martin Fowler istakao u [18], svaki mikroservis može, a ne mora, da sadrži sledeće karakteristike (što je ovde i praktično potvrđeno):

- Karakteristika servis kao komponenta (SK) se implementira u mikroservisima koji imaju neku poslovnu logiku a to su svi mikroservisi Grupa 1-5, ne uzimajući u obzir PP mikroservis jer je to mikroservis bez poslovne logike,

- Organizacija poslovnih jedinica (OPJ) je karakteristika koja neće biti primenjena u ovoj aplikaciji, pošto sam razvoj aplikacije nije moguće podržati i organizovati na takav način kao što ova karakteristika nalaže,

- Razvoj proizvoda (RP) je karakteristika koja je primenjena u svim mikroservisima, i kao takvu se može dati bilo kom klijentu,

- Mikroservisna komunikacija (MK) se može reći da je jedna i od ključnih karakteristika ove aplikacije, svi mikroservisi komuniciraju preko određenih protokola u ovom slučaju se koristi WCF (eng. *Windows Communication Foundation*) komunikacija koja koristi TCP protokol. Karakteristika je primenjena u svim mikroservisima,

- Decentralizacija (DEC) je karakteristika koja nije podržana samo u PP mikroservisu, jer je to mikroservis koji nema nikakvu poslovnu logiku, služi samo za prosleđivanje poruka od klijenta ka G2-G5 mikroservisa. Ostali mikroservisi decentrališu svoje podatke, tj. svaki mikroservis je sam za sebe, i svoju poslovnu logiku ne prikazuje drugim mikroservisima,

- Automatizacija (AUTO) je karakteristika koja je podržana u svim mikroservisima, jer na takav način treba da se isporuči svaki softver klijentu. Samim korišćenjem *cloud*-a i *Service Fabric*-a dodatno olakšava ovaj posao, i

- Detekcija greške (DG) je karakteristika koja je podržana u svim mikroservisima. Svi proračuni su važni sa aspekta korisnika i korišćenja elektroenergetske mreže i treba da se greška detektuje ako dolazi do iste.

Tabela 1 - Poređenje komponenti sa karakteristikama mikroservisa

Komponenta	**Karakterisitke mikroservisa**						
	SK	**OPJ**	**RP**	**MK**	**DEC**	**AUTO**	**DG**
Klijent	*NP*	*NP*	*NP*	*NP*	*NP*	*NP*	*NP*
PP mikroservisi	*NE*	*NE*	*DA*	*DA*	*NE*	*DA*	*NE*
G1 mikroservisi	*DA*	*NE*	*DA*	*DA*	*DA*	*DA*	*DA*
G2 mikroservisi	*DA*	*NE*	*DA*	*DA*	*DA*	*DA*	*DA*
G3 mikroservisi	*DA*	*NE*	*DA*	*DA*	*DA*	*DA*	*DA*
G4 mikroservisi	*DA*	*NE*	*DA*	*DA*	*DA*	*DA*	*DA*
G5 mikroservisi	*DA*	*NE*	*DA*	*DA*	*DA*	*DA*	*DA*
Baza	*NP*	*NP*	*NP*	*NP*	*NP*	*NP*	*NP*
Eksterni sistemi	*NP*	*NP*	*NP*	*NP*	*NP*	*NP*	*NP*

PP mikroservis je servis bez poslovne logike i od svih karakteristika, podržava razvoj proizvoda, mikroservisnu komunikaciju i automatizaciju. Razvoj proizvoda je podržan iz razloga što se sama aplikacija razvija kao proizvod i klijent dobija aplikaciju kao model PaaS. Mikroservisna komunikacija je WCF komunikacija i mikroservisi su povezani kao što je i prikazano u arhitekturi aplikacije (Slika 29). Automatizacija je važan pojam mikroservisne karakteristike, s obzirom da na platformi se nudi da se svaki mikroservis posebno postavlja na *cloud*.

G1-G5 mikroservisi imaju iste karakteristike, tj. podržava se: servis kao komponenta, razvoj proizvoda, mikroservisna komunikacija, decentralizacija, automatizacija i detekcija greške. Jedina karakteristika koja nije podržana je organizacija poslovnih jedinica. Servis kao komponenta je podržana u svim mikroservisima iz razloga što svaki mikroservis ima neku poslovnu logiku, odnosno svaki mikroservis iz svake grupe podržava neki proračun. Razvoj proizvoda je podržan zato što se sama aplikacija razvija kao proizvod i klijent dobija aplikaciju kao model PaaS. Mikroservisna komunikacija između svakog mikroservisa je WCF, a mikroservisi međusobno komuniciraju kako je naloženo u arhitekturi sistema (Slika 29). Decentralizacija je jedna od važnih karakteristika, jer se na ovaj način ukazuje da je arhitektura napravljena tako da svaka poslovna celina decentrališe svoje podatke. Svaki proračun je postavljen u jednom mikroservisu i sam mikroservis je vlasnik svih podataka. Ako je nekom mikroservisu potreban neki podatak to se zatraži WCF komunikacijom. Automatizacija je podržana u svakom mikroservisu, samim tim ako je svaki mikroservis decentralizovan, decentralizacija ima veću primenu jer svaki mikroservis može nezavisno da se postavlja na *cloud*. Detekcija greške je karakteristika koja se primenjuje u ovim mikroservisima sa poslovnom logikom, kako bi klijent bio svestan ako dođe do neke greške u sistemu. S obzirom da je DMS sistem takav da proračuni koji se koriste moraju biti tačni jer se upravlja nad elektroenergetskom mrežom, može se uočiti značaj ove karakteristike. Sa druge strane, organizacija poslovnih jedinica nije podržana iz razloga što sama aplikacija, a i timovi, ne mogu da se organizuju na način kako ova karakteristika nalaže, tako da se ona neće primeniti.

Klijent, baza i eksterni sistemi su komponente nad kojima karakteristike mikroservisa se neće primeniti. Klijent i eksterni sistemi se ne nalaze u *cloud* okruženju, a baza je komponenta nije mikroservis, kako bi karakterstike bile primenljive.

Uzimajući u obzir opisanu arhitekturu i poslovne logike servisa mogu se izdvojiti četiri vrste slučaja za koje klijent zahteva proračun:

1. Prvi slučaj je kada klijent traži proračun od G2 za određene korene elektroenergetske mreže (Slika 30). Ovde postoje šest koraka, u zavisnosti da li mikroservisi imaju sačuvan rezultat za dati koren ili ne:

    a. Klijent šalje asinhroni poziv PP mikroservisu za datim korenima za koji je potreban proračun.

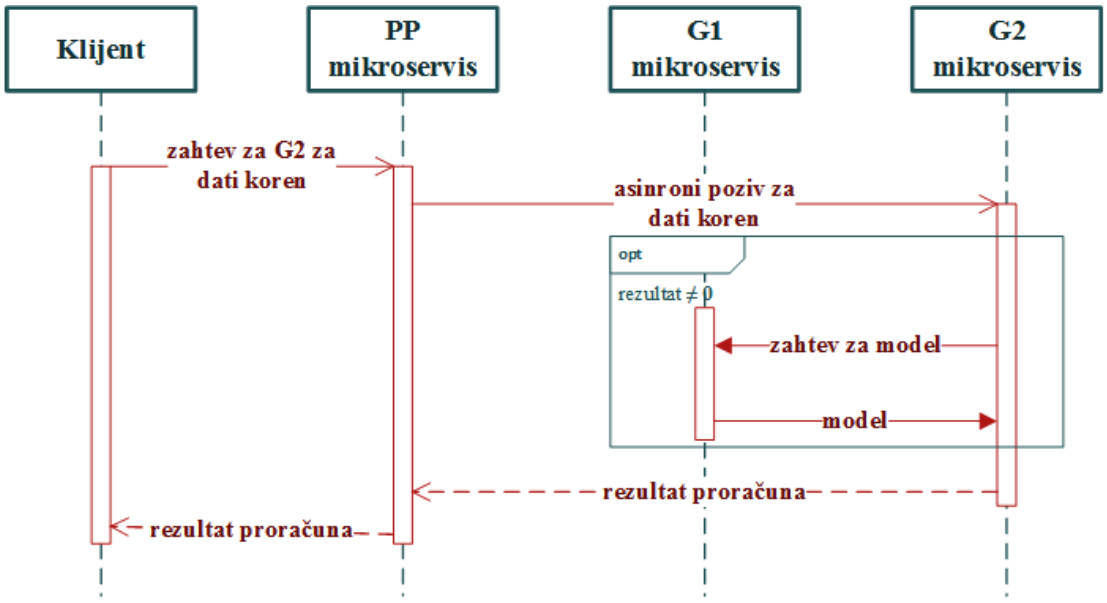
    b. PP mikroservis šalje asinhroni upit G2 mikroservisu.

    c. G2 mikroservis, ako ima sačuvan rezultat za dati koren, rezultat proračuna prosleđuje nazad klijentu i prelazi na korak *f*, a ako nema sačuvan rezultat za dati koren šalje se zahtev G1 mikroservisu.

d. G1 mikroservis kreira model elektroenergetske mreže na osnovu pristiglih korena koji je potreban za proračun G2 i takav model se šalje G2 mikroservisu.

e. Obrada proračuna G2 se vrši u G2 mikroservisu nakon svih pristiglih rezultata.

f. Nakon završetka rezultat se šalje klijentu preko PP mikroservisa.



Slika 30 - Dijagram sekvence G2 mikroservisa

2. Drugi slučaj je zahtev proračuna G3 za određene korene elektroenergetske mreže (Slika 31). Ovaj postupak može imati osam koraka, u zavisnosti od toga da li mikroservisi imaju sačuvan rezultat za dati koren ili ne:

   a. Klijent šalje asinhroni poziv PP mikroservisu za dati koren gde treba da se vrši G3 proračun elektroenergetske mreže.

   b. PP mikroservis po svojoj strukturi vrši samo rutiranje poziva ka G3 mikroservisu.

   c. G3 mikroservis, ako ima sačuvan rezultat za dati koren šalje se rezultat proračuna nazad klijentu i prelazi se na korak *h*, a ako nema sačuvan rezultat za dati koren šalje zahtev G2 mikroservisu jer za G3 proračune su potrebni i G2 proračuni i model elektroenergetske mreže.

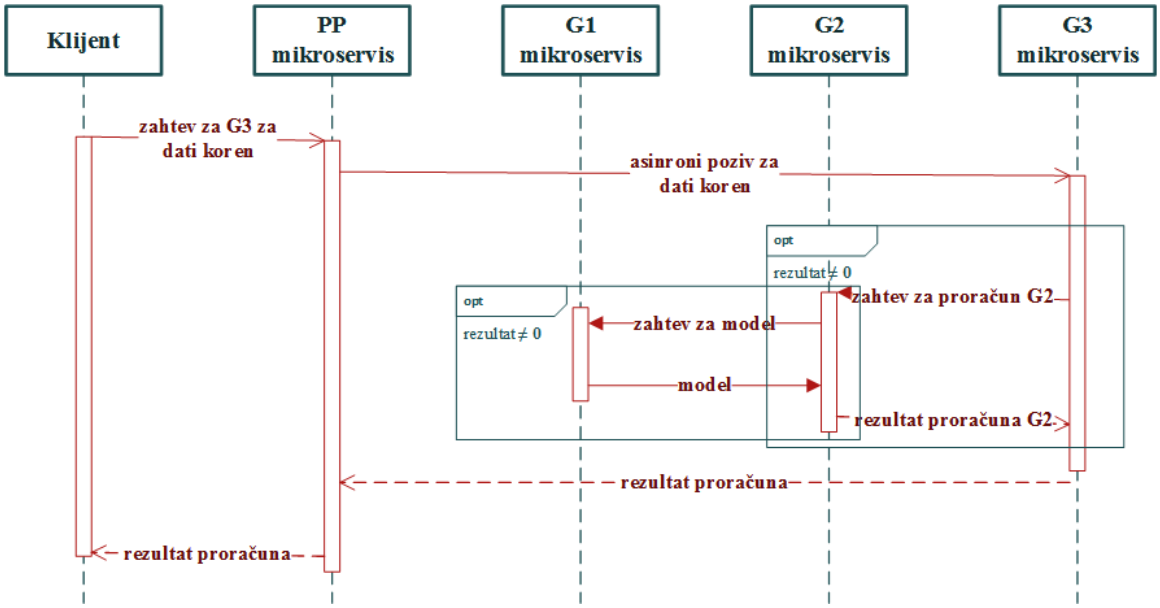
   d. G2 mikroservis, na isti način kao i G3 mikroservis, ako ima sačuvan rezultat za dati koren prosleđuje rezultat proračuna nazad G3 mikroservisu i prelazi se na korak *g*, a ako nema sačuvan rezultat za dati koren šalje zahtev G1 mikroservisu.

   e. U G1 mikroservisu kreira se model na osnovu pristiglih korena koji je potreban za proračun G2 i G3. Takav model se šalje G2 mikroservisu.

f. U G2 mikroservisu vrši se obrada proračuna G2 proračuna, pa nakon obrade rezultat se šalje nazad G3 mikroservisu.

g. U G3 mikroservisu nakon svih pristiglih rezultata, G3 proračun može da se obradi.

h. Nakon završetka rezultat se šalje klijentu preko PP mikroservisa.



Slika 31 - Dijagram sekvence G3 mikroservisa

3. Treći slučaj je zahtev za proračun G4 (Slika 32) se sastoji od sledeće procedure (deset koraka):

a. Klijent šalje asinhroni poziv PP mikroservisu za date korene elektroenergetske mreže za koje treba da se vrši proračun.

b. PP mikroservis po svojoj strukturi vrši samo rutiranje poziva ka G4 mikroservisu.

c. G4 mikroservis, ako ima sačuvan rezultat za dati koren šalje se rezultat proračuna nazad klijentu i prelazi se na korak *j*, a ako nema sačuvan rezultat za dati koren šalje zahtev G3 mikroservisu jer za G4 proračune su potrebni i G3 i G2 proračuni i model elektroenergetske mreže. Slanje zahteva G3 mikroservisu može da bude *n* puta u zavisnosti od delta greške proračuna utvrđena proračunom G3.

d. G3 mikroservis ako ima sačuvan rezultat za dati koren vraća rezultat nazad G4 mikroservisu i prelazi se na korak *h*, a ako ne, šalje se zahtev G2 mikroservisu.

e. G2 mikroservis ako ima sačuvan rezultat vraća nazad G3 mikroservisu i prelazi se na korak *h*, a ako nema sačuvan rezultat šalje se zahtev G1 mikroservisu.

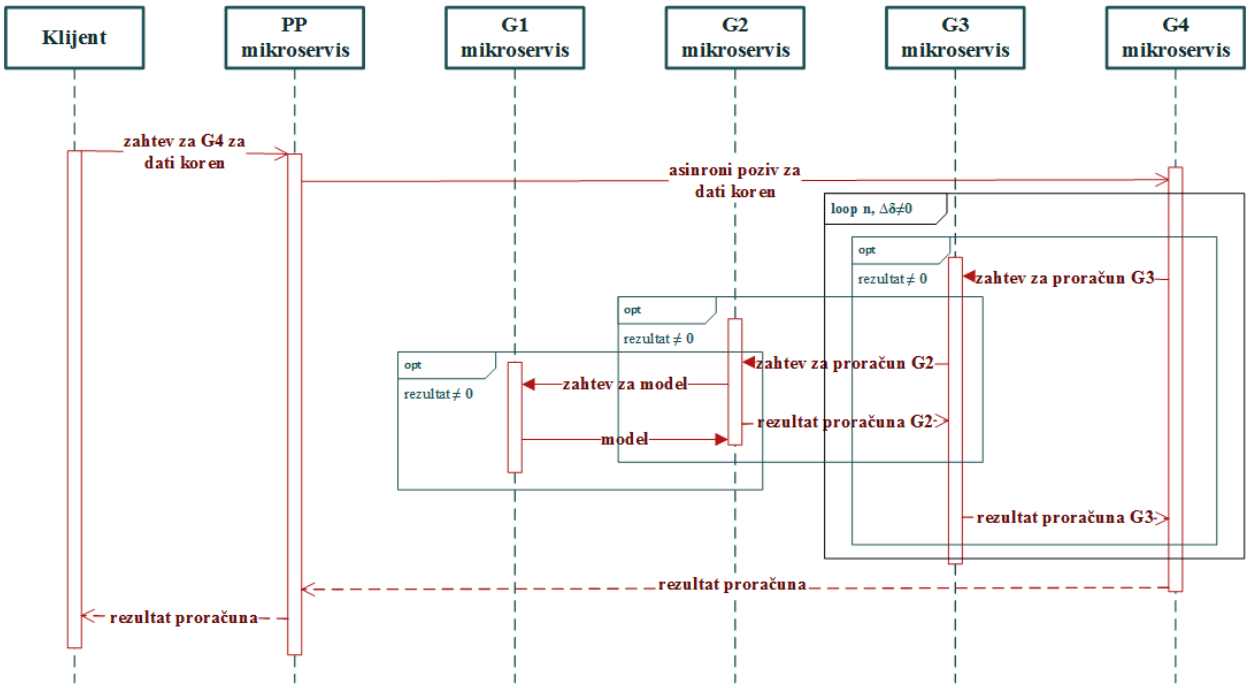
f. U G1 mikroservisu kreira se model na osnovu pristiglih korena koji je potreban za proračune G2, G3 i G4. Takav model se šalje G2 mikroservisu.

g. U G2 mikroservisu se vrši obrada proračuna pa se rezultat šalje nazad G3 mikroservisu.

h. U G3 mikroservisu nakon svih rezultata, G3 proračun može da se obradi i rezultat se šalje nazad G4 mikroservisu.

i. Nakon što su svi podaci na jednom mestu u G4 mikroservisu, G4 proračun može da se izvrši.

j. Nakon završetka G4 proračuna rezultat se šalje klijentu preko PP mikroservisa.



Slika 32 - Dijagram sekvence G4 mikroservisa

4. Četvrti slučaj je zahtev za proračun G5 (Slika 33) koji se može sastojati iz dvanaest koraka, u zavisnosti od toga da li mikroservisi imaju sačuvan rezultat za dati koren ili ne:

a. Klijent šalje asinhroni poziv PP mikroservisu za date korene elektroenergetske mreže za koje treba da se vrši proračun.

b. PP mikroservis vrši rutiranje poziva ka G5 mikroservisu.

c. G5 mikroservis, ako ima sačuvan rezultat za dati koren šalje se rezultat proračuna nazad klijentu i prelazi se na korak *l*, a ako nema obrađen rezultat za dati koren šalje zahtev G4 mikroservisu jer za G5 proračune su potrebni i G4, G3 i G2 proračuni i model elektroenergetske mreže.

d. G4 mikroservis, ako ima sačuvan rezultat za dati koren rezultat se vraća G5 mikroservisu i prelazi se na korak *k*, a ako nema šalje se zahtev G3 mikroservisu. Slanje zahteva G3 mikroservisu može da bude *n* puta u zavisnosti od delta greške proračuna utvrđena proračunom G3.

e. U G3 mikroservisu ako ima sačuvan rezultat za dati koren, rezultat se vraća G4 mikroservisu i prelazi se na korak *j*, a ako nema šalje se zahtev G2 mikroservisu.

f. G2 mikroservis, ako ima sačuvan rezultat za dati koren rezultat se vraća G3 mikroservisu i prelazi se na korak *h*, a ako nema šalje se zahtev G1 mikroservisu.

g. U G1 mikroservisu kreira se model na osnovu pristiglih korena koji je potreban za proračune G2, G3, G4 i G5. Takav model se šalje G2 mikroservisu.

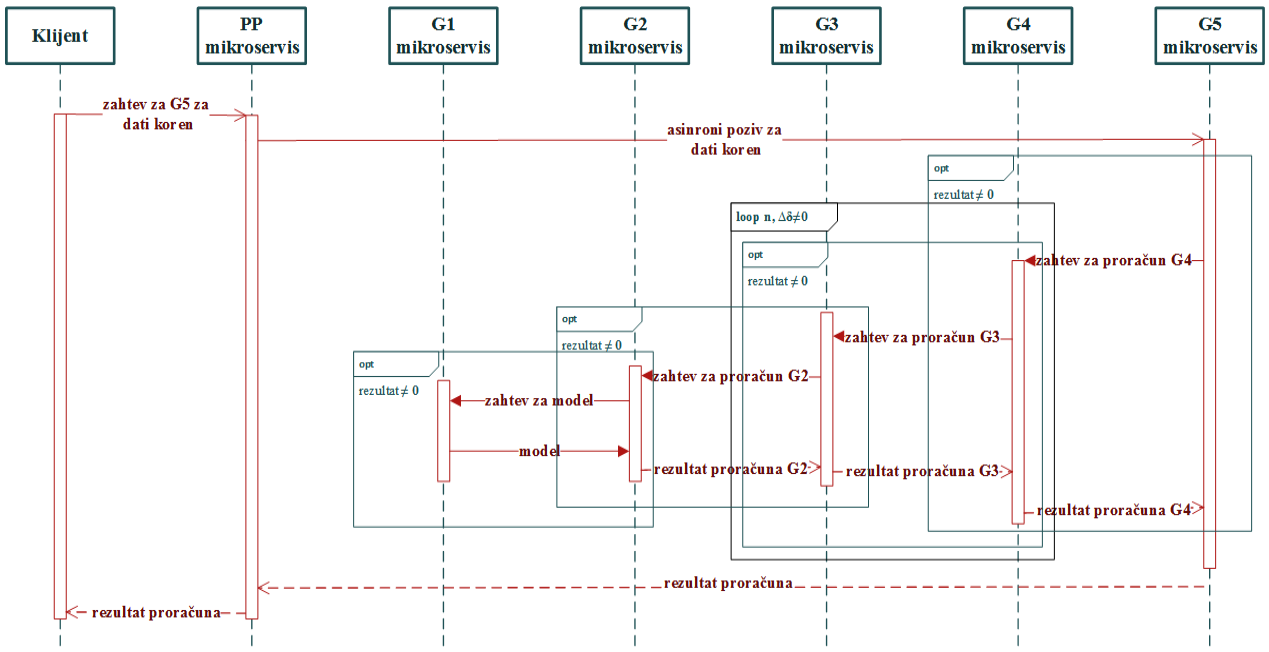
h. U G2 mikroservisu se vrši obrada proračuna pa se rezultat šalje nazad G3 mikroservisu.

i. U G3 mikroservisu nakon svih rezultata, G3 proračun može da se obradi i rezultat se šalje nazad G4 mikroservisu.

j. Isto važi i za G4 mikroservis, nakon obrade proračuna rezultati se šalje G5 mikroservisu.

k. Nakon što su svi podaci na jednom mestu u G5 mikroservisu, G5 proračun može da se izvrši.

l. Nakon završetka rezultat se šalje klijentu preko PP mikroservisa.



Slika 33 - Dijagram sekvence G5 mikroservisa

7.3 Transformacija proračuna na *Service Fabric* mikroservisno okruženje

Na osnovu predložene arhitekture (Slika 29), u ovom poglavlju je opisan način transformacije takve mikroservisne arhitekture na *Service Fabric*. Kao što je gore objašnjeno da svaki proračun pripada određenoj grupi (Slika 28), ali će na ovom mestu takav proračun iz svih grupa biti prezentovan kao jedan mikroservis koji ima primarnu i svoju repliku odnosno sekundarnu instancu na *Service Fabric*, što znači da ako dolazi do pada primarnog servisa njegov sekundarni će preuzeti posao. Uz oslonac na *Service Fabric*, svaki mikroservis odnosno proračun iz određene grupe je implementiran kao *stateless* i *stateful* kako bi zadovoljio kriterijume same platforme (opisano u poglavlju 5.1.1) [103].

Klijent u tabeli iznad (Tabela 1) ima oznaku NP, što znači da nije podržano. Jedino se mikroservisi transformišu na *Service Fabric*.

PP mikroservis ima oznaku u polju *stateless* (Tabela 2), što znači da je mikroservis koji ne čuva stanje. Kao što je i objašnjeno u poglavlju 5.1, to je mikroservis koji služi za prosleđivanje poruka do klijenta do G2-G5 mikroservisa.

MP mikroservis zbog njegove kompleksnosti, uloge i važnosti u datoj arhitekturi je modelovan kao *stateful* mikroservis (Tabela 2). Ovaj mikroservis je od kritičnijih u datoj arhitekturi i mora da čuva stanje, a to je sam model elektroenergetske mreže. Koristeći platformu *Service Fabric* postoji mehanizam za *stateful* mikroservise gde se podaci repliciraju na svoje ostale replike, što znači da nakon dodavanja nekog podatka u kolekciju, sama platforma vrši repliciranje na ostale replike datog mikroservisa. Ovo je dobro iz više razloga: sam mikroservis je kritičan jer ima u sebi ceo model elektroenergetske mreže i ako dolazi do rušenja jednog od aktivnih mikroservisa, automatski njegov sekundarni preuzima posao, a time i trenutni aktivni model elektroenergetske mreže. U zavisnosti od veličine modela postoji mogućnost particionisanja, gde je kriterijum podele na nezavisne delove odnosno korene elektroenergetske mreže kao što je opisano u radovima [104][105][106].

TA mikroservis je modelovan kao *stateful* mikroservis (Tabela 2), zato što je potrebno da se čuva stanje prethodno izračunate topološke analize na osnovu korena, kako bi bio brži algoritam ako tokom rada aplikacije zatreba topološka analiza za isti koren. Algoritam koji je iskorišćen za testiranje je opisan u radu [41], a to je algoritam matrične topološke analize. Implementirana je topološka analiza elektroenergetske mreže koja ima zadatak da odredi galvanski povezane celine (korene i ostrva), zatim da prilagodi podatke o povezanosti elemenata u mreži kako bi se omogućilo brzo i efikasno izvršenje elektroenergetskih proračuna i opsluži svaki elemenat mreže potrebnim informacijama kako bi klijent imao mogućnost uvida u trenutno stanje mreže. Topološka analiza se određuje za svaki koren posebno i pritom se obrađuju samo neažurni koreni, dok se ažurni koreni čuvaju interno u samom mikroservisu u *Reliable* kolekcijama opisane u poglavlju 5.1.2. Ovo je značajno ako postoji slučaj da se promeni status rasklopne opreme nekog polja ili se u model doda novi element mreže, tada se izvršava jedino ažuriranje topološke analize korena zahvaćenog promenom. Ovaj pristup značajno ubrzava proces ažuriranja grafa mreže, jer se izmene dešavaju samo u korenu gde one jedino i postoje. Koreni mreže se inicijalno izračunaju prilikom inicijalne topološke analize mreže, kao i prilikom drugih elektroenergetskih proračuna (npr. Volt/Var optimizacija). Rad topološke analize u velikim elektroenergetskim mrežama, gde postoje više miliona potrošača, predstavlja izuzetno kompleksan i zahtevan zadatak, ali pomenuti algoritam za obilazak grafa, je značajan sa strane

performansi jer koristi osobine retke matrice za memorisanje i operacije sa njenim elementima korišćena je tehnika retkih matrica, koja čuva nenulte elemente matrice i time omogućava da se prilikom obilaska grafa prođe kroz samo nenulte elemente jednog reda, tj. susede određenog čvora [38][107].

TS mikroservis deo G3 je modelovan kao *stateful* mikroservis (Tabela 2). Tokovi snaga se ponovo proračunavaju kada se izmeni stanje elektroenergetske mreže i usko su vezani sa topološkom analizom odnosno svaki proračun zavisi od topološke analize, pa samim tim treba da se pamti rezultat svakog izračunatog korena.

Putanja mikroservis deo G3 je modelovan kao *stateless* mikroservis (Tabela 2). Jednostavan proračun koji se dobija po potrebi radi analize grafa mreže, pa samim tim ne treba da pamti rezultat.

ES mikroservis deo G4 je modelovan kao *stateful* mikroservis (Tabela 2). ES proračun je vrlo važan i usko je vezan sa TA i TS, što znači da je mikroservis koji treba da pamti rezultat, odnosno proračun po korenu elektroenergetske mreže.

KS mikroservis deo G4 je modelovan kao *stateful* mikroservis (Tabela 2). KS proračun od koga zavisi i KPO i RZ pa je bitno da čuva rezultat proračuna radi bržeg proračuna kada dolazi do potrebe za KPO i RZ proračunima.

PMK mikroservis deo G4 je modelovan kao *stateless* mikroservis (Tabela 2). Ovaj proračun utiče na kvarove i na procenu kvara, i s obzirom da u sistemu ne može da se utiče na kvar i kakav će biti ovaj mikroservis ne treba da pamti rezultat.

RVP mikroservis deo G4 je modelovan kao *stateless* mikroservis (Tabela 2). RVP proračun se pokreće posle nekog kvara, da bi se uradila restauracija, a kao što je gore navedeno na kvarove se ne može uticati, ovaj mikroservis ne pamti rezultat.

REK mikroservis deo G4 je modelovan kao *stateful* mikroservis (Tabela 2). REK je značajan proračun koji daje rekonfiguraciju mreže na osnovu TA i TS proračuna, pa samim tim treba da pamti proračun svakog korena.

Prognoza mikroservis deo G4 je modelovan kao *stateless* mikoservis (Tabela 2). Prognoza je proračun koji zavisi i od spoljnih faktora, što znači da može doći do konstantne promene vrednosti, a i klijent traži po potrebi ovaj proračun, što implicira da ne treba da pamti rezultat.

SM mikroservis deo G5 je modelovan kao *stateful* mikroservis (Tabela 2). SM proračun zavisi od TA i ES i služi da bi se detektovala u kakvom je stanju elektroenergetska mreža, kako bi se neki složeniji proračun odradio, ovaj mikroservis pamti izračunati proračun.

Volt/Var mikroservis deo G5 je modelovan kao *stateful* mikroservis (Tabela 2). Volt/Var proračun je važan u elektroenergetskoj mreži, zavisi od TA, TS i ES, pa samim tim se svaki rezultat pamti.

KPO mikroservis deo G5 je modelovan kao *stateless* mikroservis (Tabela 2). KPO je proračun za koji nije potrebno da se pamti prethodno odrađeni rezultat.

RZ mikroservis deo G5 je modelovan kao *stateless* mikroservis (Tabela 2). RZ je proračun koji zavisi i od kvarova u elektroenergetskoj mreži na koji se ne može uticati, pa se u ovom mikroservisu ne čuva rezultat proračuna.

Tabela 2 - Odnos komponenti i tipovi mikroservisa (*stateless* ili *stateful*)

Komponenta	**Proračun**	***Tip servisa***
Klijent	-	*NP*
PP mikroservisi	-	*Stateless*
G1 mikroservisi	Model	*Stateful*
G2 mikroservisi	TA	*Stateful*
G3 mikroservisi	TS	*Stateful*
	Putanja	*Stateless*
G4 mikroservisi	ES	*Stateful*
	KS	*Stateful*
	PMK	*Stateless*
	RVP	*Stateless*
	REK	*Stateful*
	Prognoza	*Stateless*
G5 mikroservisi	SM	*Stateful*
	Volt/Var	*Stateful*
	KPO	*Stateless*
	RZ	*Stateless*

Kako bi se mikroservisi postavili na *Service Fabric* mora da se odrediti koliko čvorova treba da se konfiguriše i koliko primarnih instanci mikroservisa svakog tipa mikroservisa treba da se postavi. Da bi se to uradilo u tabeli ispod (Tabela 3) je data podela mikroservisa po kategoriji i svaka kategorija ima svoj index - parametar koji će se iskoristiti za izračunavanje broja postavljenih primarnih mikroservisa na čvorove. Kategorije su postavljene po važnosti i stepenu korišćenja proračuna u elektroenergetskoj mreži. Definisano je 5 kategorije i mikroservisi su podeljeni na sledeći način:

1. K1 sa indexom 1 sadrži TA mikroservis i TS mikroservis.
2. K2 sa indexom 2 sadrži ES mikroservis i KS mikroservis.
3. K3 sa indexom 3 sadrži SM mikroservis i Volt/Var mikroservis.
4. K4 sa indexom 4 sadrži MP mikroservis i REK mikroservis.
5. K5 sa indexom 5 sadrži PP mikroservis, Putanja mikroservis, PMK mikroservis, RVP mikroservis, KPO mikroservis i RZ mikroservis.

Tabela 3 - Odnos kategorije i mikroservisa

Kategorija	**K1**	**K2**	**K3**	**K4**	**K5**
Mikroservisi	TA mikroservis	ES mikroservis	SM mikroservis	MP mikroservis	PP mikroservis
	TS mikroservis	KS mikroservis	Volt/Var mikroservis	REK mikroservis	Putanja mikroservis
					PMK mikroservis
					RVP mikroservis
					Prognoza mikroservis
					KPO mikroservis
					RZ mikroservis

Za postavljanje aplikacije na *Service Fabric* najpre mora da se izračuna broj čvorova nad kojima će se aplikacija postaviti. Broj čvorova se računa po sledećoj formuli:

$$\mathbf{G} = \boldsymbol{max}\{(\boldsymbol{U_{mik}} - \boldsymbol{U_{kat}}), \boldsymbol{n_{min}}\} \quad (1)$$

gde su:

- G - broj čvorova,
- U_{mik} - ukupan broj svih mikroservisa,
- U_{kat} - ukupan broj svih kategorija, i
- n_{min} - minimalni broj čvorova. U ovim razmatanjima je postavljeno $n_{min} = 5$.

S obirom da je poznato kako se izračunava broj potrebnih čvorova (1), može da se izračuna broj primarnih mikroservisa za svaki proračun i to po sledećoj formuli:

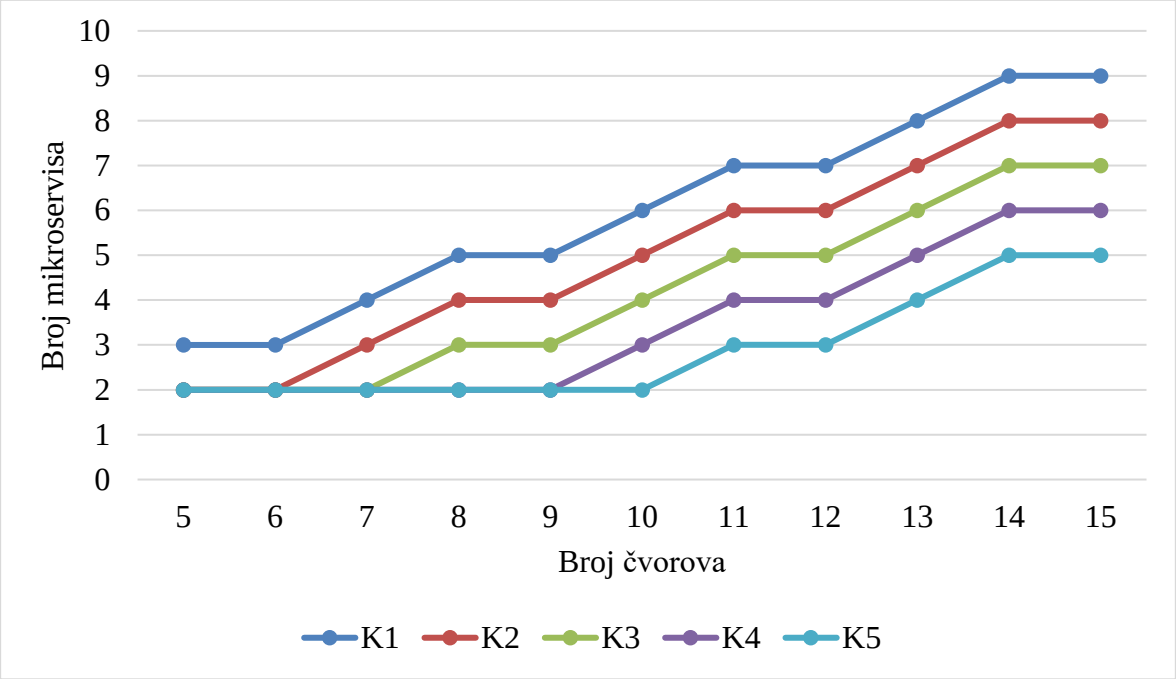
$$\boldsymbol{MIK} = \boldsymbol{max}\left\{(\boldsymbol{G} - \boldsymbol{i} - \boldsymbol{n_{min}} * \left(\frac{\boldsymbol{G}}{\boldsymbol{U_{kat}}}\right)), \boldsymbol{p_{min}}\right\} \quad (2)$$

gde su:

- MIK - ukupan broj jedne vrste primarnih mikroservisa,
- G - broj čvorova,
- i - index kategorije. Index kategorije je objašnjen u pasusu iznad,

- U_{kat} - ukupan broj svih kategorija,
- p_{min} - minimalan broj primarnih mikroservisa. U razmatranjima ove doktorske disertacije je $p_{min} = 2$, i
- n_{min} - minimalni broj čvorova. U razmatanjima ove doktorske disertacije je $n_{min} = 5$.

Na osnovu predložene formule za proračun primarnih mikroservisa (2) može se videti odnos čvorova (od 5 do 15) i kategorije mikroservisa (Slika 34).



Slika 34 - Odnos čvorova i kategorije

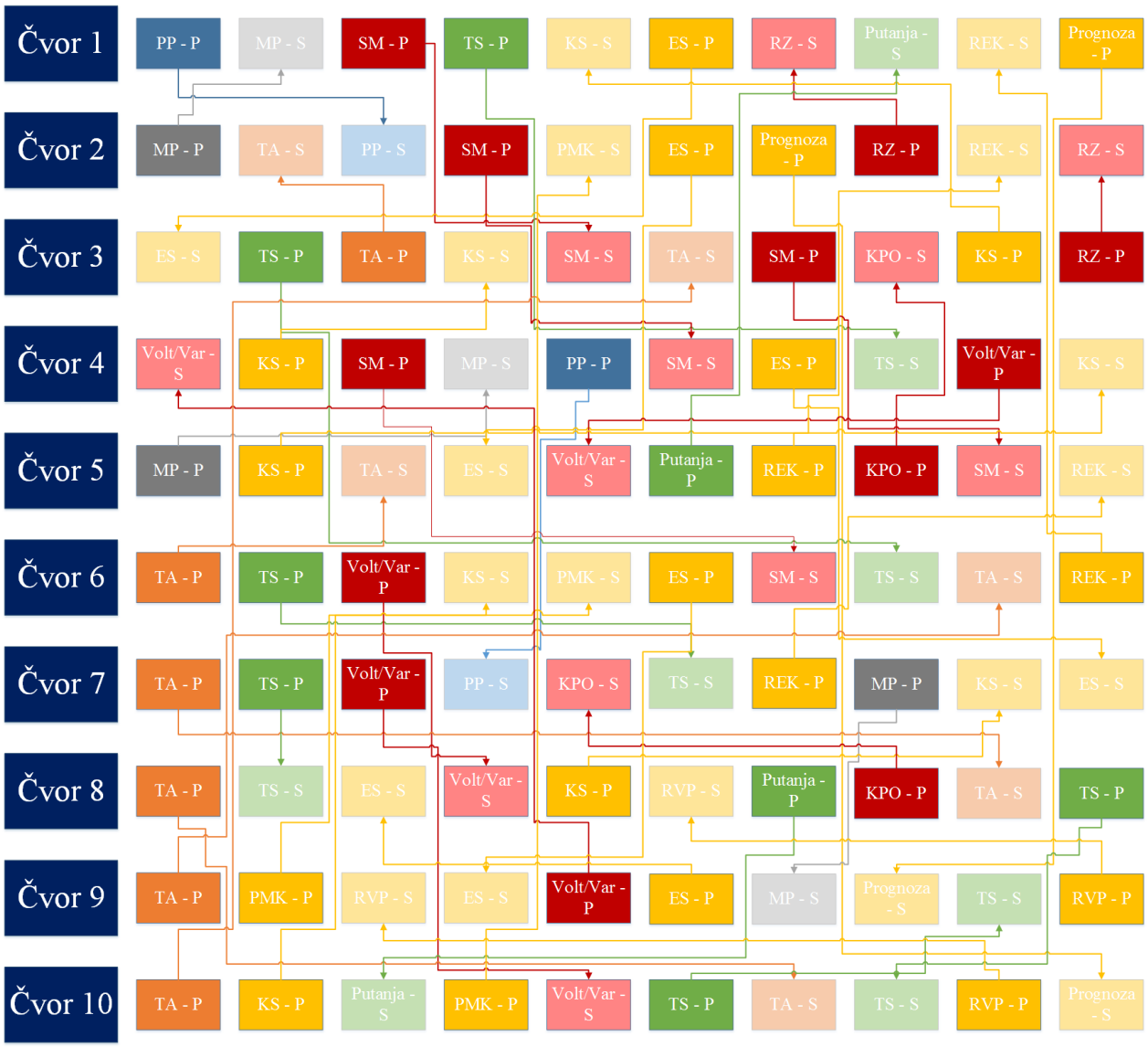
Ako uzmemo u obzir da će se na *Service Fabric* postaviti sve grupe mikroservisa, po formuli (1) se dobije da deset čvorova treba da bude konfigurisano ($U_{mik} = 15$ i $U_{kat} = 5$). Sami mikroservisi se skaliraju na tih deset čvorova (Slika 35) po datoj formuli iznad (2), gde su G = 10, parametar i je promenljiv za svaku kategoriju (vrednost 1 do 5), i $U_{kat} = 5$. Promenjive p_{min} i n_{min} su već definisane iznad, tj. $p_{min} = 2$ i $n_{min} = 5$. U tabeli ispod (Tabela 4) je prikazano broj primarnih servisa za svaku kategoriju mikroservisa. Za svaki primarni mikroservis biće postavljen i njegov sekundarni.

Tabela 4 - Broj primarnih mikroservisa za sve kategorije na deset čvorova

Kategorija	K1	K2	K3	K4	K5
Broj primarnih mikroservisa	6	5	4	3	2

Znajući broj čvorova i broj primarnih i sekundarnih mikroservisa na slici ispod (Slika 35) se mogu videti i zavisnosti primarnih i sekundarnih instanci na čvorovima kako bi se stvorila

jasnija slika o rasporedu sekundarnih mikroservisa iste instance i preuzimanju posla ukoliko dođe do otkaza odnosno pada nekog čvora. Takođe, mikroservisi po pripadnosti grupi su obojeni jednom bojom i to: PP mikroservisi plavom bojom, G1 mikroservisi sivom bojom, G2 mikroservisi narandžastom bojom, G3 mikroservisi zelenom bojom, G4 mikroservisi žutom bojom i G5 mikroservisi crvenom bojom, a i svaki primarni i sekundarni ima oznaku P (primarni) odnosno S (sekundarni).



Slika 35 - Skaliranje svih mikroservisa aplikacije na *Service Fabric* (boja mikroservisa odgovara grupi)

Mikroservisi su na sledeći način konfigurisani po čvorovima (Slika 35):

- Čvor 1 sadrži pet primarnih instanci mikroservisa i to: PP mikroservis, SM mikroservis, TS mikroservis, ES mikroservis i Prognoza mikroservis, kao i pet sekundarnih instanci mikroservisa: MP mikroservis, KS mikroservis, RZ mikroservis, Putanja mikroservis i REK mikroservis.

- Čvor 2 sadrži pet primarnih instanci mikroservisa i to: MP mikroservis, SM mikroservis, ES mikroservis, Prognoza mikroservis i RZ mikroservis, kao i pet

sekundarnih instanci mikroservisa: TA mikroservis, PP mikroservis, PMK mikroservis, REK mikroservis i RZ mikroservis.

- Čvor 3 sadrži pet primarnih instanci mikroservisa i to: TS mikroservis, TA mikroservis, SM mikroservis, KS mikroservis i RZ mikroservis, kao i pet sekundarnih instanci mikroservisa: ES mikroservis, KS mikroservis, SM mikroservis, TA mikroservis i KPO mikroservis.

- Čvor 4 sadrži pet primarnih instanci mikroservisa i to: KS mikroservis, SM mikroservis, PP mikroservis, ES mikroservis i Volt/Var mikroservis, kao i pet sekundarnih instanci mikroservisa: Volt/Var mikroservis, MP mikroservis, SM mikroservis, TS mikroservis i KS mikroservis.

- Čvor 5 sadrži pet primarnih instanci mikroservisa i to: MP mikroservis, KS mikroservis, Putanja mikroservis, REK mikroservis i KPO mikroservis, kao i pet sekundarnih instanci mikroservisa: TA mikroservis, ES mikroservis, Volt/Var mikroservis, SM mikroservis i REK mikroservis.

- Čvor 6 sadrži pet primarnih instanci mikroservisa i to: TA mikroservis, TS mikroservis, Volt/Var mikroservis, ES mikroservis i REK mikroservis, kao i pet sekundarnih instanci mikroservisa: KS mikroservis, PMK mikroservis, SM mikroservis, TS mikroservis i TA mikroservis.

- Čvor 7 sadrži pet primarnih instanci mikroservisa i to: TA mikroservis, TS mikroservis, Volt/Var mikroservis, REK mikroservis i MP mikroservis, kao i pet sekundarnih instanci mikroservisa: PP mikroservis, KPO mikroservis, TS mikroservis, KS mikroservis i ES mikroservis.

- Čvor 8 sadrži pet primarnih instanci mikroservisa i to: TA mikroservis, KS mikroservis, Putanja mikroservis, KPO mikroservis i TS mikroservis, kao i pet sekundarnih instanci mikroservisa: TS mikroservis, ES mikroservis, Volt/Var mikroservis, RVP mikroservis i TA mikroservis.

- Čvor 9 sadrži pet primarnih instanci mikroservisa i to: TA mikroservis, PMK mikroservis, Volt/Var mikroservis, ES mikroservis i RVP mikroservis, kao i pet sekundarnih instanci mikroservisa: RVP mikroservis, ES mikroservis, MP mikroservis, Prognoza mikroservis i TS mikroservis.

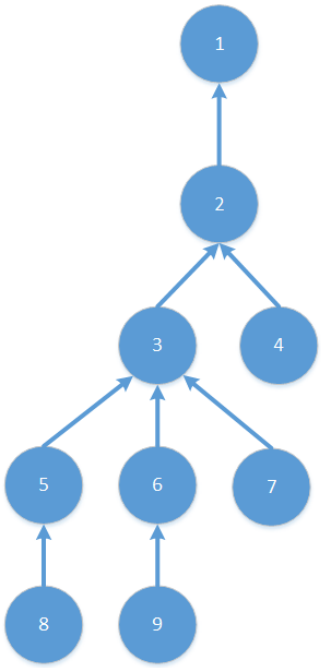
- Čvor 10 sadrži pet primarnih instanci mikroservisa i to: TA mikroservis, KS mikroservis, PMK mikroservis, TS mikroservis i RVP mikroservis, kao i pet sekundarnih instanci mikroservisa: Putanja mikroservis, Volt/Var mikroservis, TA mikroservis, TS mikroservis i Prognoza mikroservis.

7.4 Algoritam optimalnog izvršavanja proračuna

Uvođenjem algoritma optimalnog izvršavanja proračuna predložena arhitektura dobija na ubrzanju svih proračuna tako što se u postojećem stablu proračuna pokreću samo proračuni koji su neophodni za dobijanje rezultata trenutno traženog proračuna. Da bi se lakše objasnio

algoritam, na slici ispod (Slika 36) se može videti stablo svih proračuna, podeljenih po zavisnosti na osnovu slike iznad (Slika 27). Stablo sadrži sledeće čvorove:

- Čvor 1 sadrži model elektroenergetske mreže,
- Čvor 2 sadrži proračun topološke analize,
- Čvor 3 sadrži proračun tokove snaga,
- Čvor 4 sadrži proračun putanju,
- Čvor 5 sadrži proračun estimaciju stanja,
- Čvor 6 sadrži proračun kratki spojevi,
- Čvor 7 sadrži proračune: procena mesta kvara, restauracija velikog područja, rekonfiguracija i prognoza,
- Čvor 8 sadrži proračune: spremnost modela i Volt/Var optimizacija, i
- Čvor 9 sadrži proračune: kapacitet prekidača i osigurača i relejnu zaštitu.



Slika 36 - Graf zavisnosti proračuna

Algoritam će biti ilustrovan na par primera gde se pokreću proračuni i kako se čuvaju rezultati:

- Ukoliko klijent zatraži izvršavanje proračuna iz čvora 3 za neki koren, gledajući stablo (Slika 35) može se videti da za čvor 3 su potrebni proračuni i iz čvorova 1 i 2. Ako se u čvoru 3 ne nalazi rezultat korena koji je zatražen, zatraže se rezultati od čvora 2. U čvoru 2 se na identičan način ako ne nalazi proračun za dati koren, vrši

se obrada i prilikom izvršavanja zadatog korena čuva se rezultat za neko naredno izvršavanje. Takođe u čvoru 3 nakon obrade se takođe čuva proračun tokove snaga na osnovu korena.

- Ukoliko klijent zatraži izvršavanje proračuna iz čvora 5, pritom analizirajući stablo (Slika 35), može se zaključiti da su potrebni proračuni iz čvora 2 i 3. Ako se u samom čvoru 3 već nalazi rezultat korena koji je zatražen za čvor 5, rezultati se samo prosleđuju čvoru 5, ako ne, uzimaju se rezultati od čvora 2, pa se onda vrši obrada proračuna u čvoru 3 i rezultati se čuvaju. Nakon stizanja rezultata do čvora 5 vrši se obrada i rezultati ovog proračuna se isto čuvaju za neko naredno izvršavanje.

- Ukoliko klijent zatraži izvršavanje proračuna iz čvora 9, pritom analizirajući stablo sa slike iznad (Slika 35), može se zaključiti da su potrebni proračuni iz čvora 2, 3 i 6. Ako se u samom čvoru 6 već nalazi rezultat korena koji je zatražen za čvor 9, rezultati se samo prosleđuju čvoru 9, ako ne, uzimaju se rezultati od čvora 3 i po potrebi ako ne postoji rezultati u čvoru 3 zatraži i od čvora 2, i nakon stizanja svih rezultata vrši se obrada proračuna i u čvoru 6 i rezultati se čuvaju interno. Sada kada su svi rezultati na jednom mestu za proračun čvora 9, vrši se obrada i rezultati ovog proračuna se isto čuvaju za neko naredno izvršavanje.

Ovakav vid algoritma je značajan kad više klijenata istovremeno traži proračun za date korene. S obzirom da je ovo distribuiran sistem i kako je i prikazano na slici same arhitekture (Slika 29), u sistemu u svakom trenutku može biti aktivno više klijenata koji rade nešto nad elektroenergetskom mrežom. Zbog ovoga je značajno da se proračuni pamte po korenima, kako ne bi arhitekturu natrpali sa istim promenama za iste korene, ako već ti proračuni su bili obrađeni i izračunati.

Primena ovog algoritma je značajna i primenjiva kad u sistemu postoje više aktivnih klijenata. Svaki klijent ima određenu oblast nadležnosti, odnosno nadležnost nad korenima elektroenergetske mreže (eng. *Area of responsibility* - AOR). Kada klijent dobija tu nadležnost ima mogućnost promene opreme u elektroenergetskoj mreži, da zatraži određene proračune nad korenima koje su u nadležnosti, dok sa druge strane klijenti koji nemaju tu nadležnost mogu samo da vide stanje elektroenergetske mreže i sve njene promene. Ako više klijenata zatraži proračune nad istim korenima tada algoritam ima veću primenu jer samo jednom dolazi do proračuna nad jednim korenom. Ukoliko korisnici rade nad istim delovima sistema, onda su njihovi AOR-i disjunktni i oni mogu da traže proračune nad istim korenima, ali ne mogu da menjaju iste korene. U takvoj situaciji, npr. jedan korisnik može da menja statuse prekidača u nekom korenu (i njemu susednim korenima), ali više njih može da traži proračune nad tim korenima. Pored toga, kada jedan klijent izmeni stanje mreže za te korene, ostali klijenti nemaju to pravo, pa se resetuju rezultati svih proračuna za afektirane korene, s obzirom da se svi proračuni mikroservisa čuvaju po datim korenima, dok ne dolazi do promene proračuna topološke analize iz G2 mikroservisa. Tada se šalje zahtev svim mikroservisima iz ostalih ostalih grupa/čvorova radi anuliranja *Reliable* kolekcije sačuvanih proračuna za date korene. Najčešće su promene stanja mreže praćene pokretanjem osnovnih proračuna iz grupa G1-G3, čime se njihovi rezultati pripremaju za dalje korišćenje. Na taj način se dodatno povećava efikasnost sistema, a i zainteresovani klijenti dobijaju tačnije rezultate (u skladu sa trenutnom promenom stanja). Klijenti te promene dobija tako što se pretplaćuju na određene korene preko Publisher/Subscriber mehanizma [78][79].

7.5 Algoritam za dodavanje i uklanjanja proračuna iz sistema

Razvojem sistema može se ukazati potreba da se prošire njegove funkcionalnosti sa aspekta dodavanja nekih novih proračuna i uklanjanje postojećih proračuna, čime bi se upotpunile potrebe klijenata za praćenjem i upravljanjem sistemom. Novi proračuni mogu biti neka varijacija postojećih proračuna, ali se isto tako može pojaviti potpuno nova funkcionalnost koju bi trebalo ugraditi u postojeći sistem. Postojeći proračuni se nekad menjaju odnosno sistem nema potrebu za korišćenjem istih, pa je njihovo uklanjanje nešto što sistem treba da podržava. U takvim slučajevima treba predvideti akcije koje će omogućiti tu nadogradnju, bilo da je dodavanje ili uklanjanje. Ovakve akcije utiču na sam graf i dolazi do promene grafa zavisnosti proračuna.

Sa aspekta predloženog dizajna dodavanje novog proračuna, sa jedne strane podrazumeva određivanje grupe kojoj se dodeljuje proračun, a sa druge strane utiče na promenu grafa zavisnosti proračuna. Novi proračun može biti dodat u sistem u:

1. Postojeći čvor

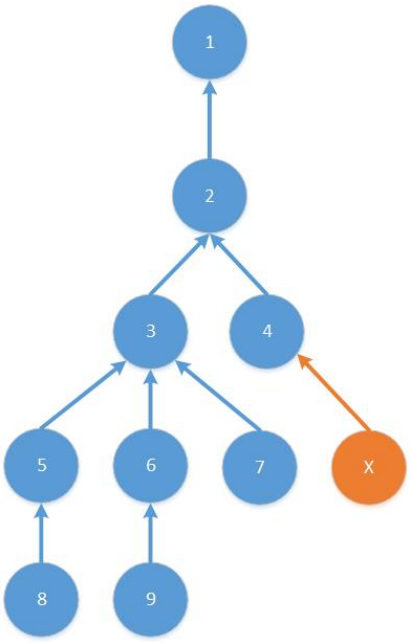
2. Novi čvor koja je list grafa, a zavisi od postojećeg čvora.

Ukoliko se u sistem uvodi novi proračun koji, po svojoj funkcionalnosti, pripada nekom čvoru onda se graf zavisnosti proračuna ne menja, već se samo evidentira dodavanje novog proračuna u postojeći čvor kome pripada.

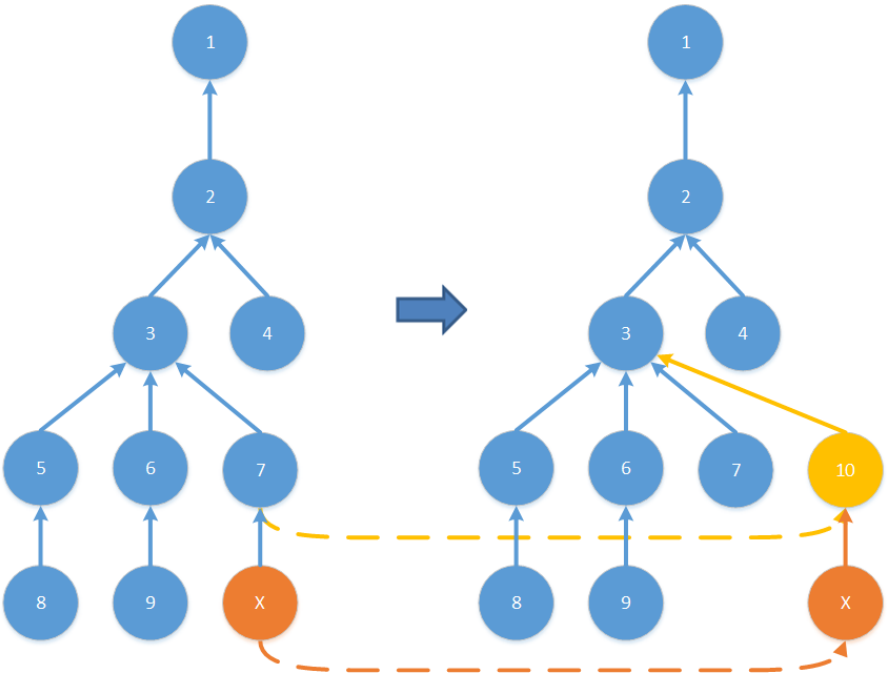
Ukoliko se u sistem uvodi novi proračun u već postojeći čvor, a zavisi od postojećeg čvora mogu da se razlikuju dve situacije:

1. Ako postojeći čvor P sadrži samo jedan proračun, a novi proračun koji se dodaje po svojoj funkcionalnosti zavisi od postojećeg proračuna koji se nalazi u tom čvoru P, u tom slučaju kreira se novi čvor N koji će sadržati novi proračun i direktno se spaja sa već postojećim čvorom P. Ovo može se videti na slici ispod (Slika 37), gde prethodno stanje grafa zavisnosti je prikazano na slici iznad (Slika 36), kad se novi proračun koji je deo čvora X dodaje u sistem a po svojoj funkcionalnosti zavisi od proračuna koji se nalazi u čvoru 4, ta dva čvora se povezuju i kreira se zavisnost.

2. Ako postojeći čvor P sadrži više proračuna, a novi proračun koji se dodaje po svojoj funkcionalnosti zavisi od postojećeg proračuna koji se nalazi u čvoru P, u tom slučaju dolazi do cepanja postojećeg čvora P na dva čvora, i to na čvor koji će sadržati samo proračun od koga zavisi novododati proračun i na čvor koji će sadržati preostale proračune. Ovo se može videti na slici ispod (Slika 38), ako se proračun X dodaje u sistem tako što po njegovoj funkcionalnosti treba da zavisi od proračuna koji se nalazi u čvoru P=7 (levi deo slike). S obzirom da čvor P=7 sadrži više proračuna, dolazi do cepanja čvora na dva dela, što znači da proračun od koga zavisi novi proračun X će formirati novi čvor koji će u ovom slučaju biti čvor 10, pa proračun X će zavisiti od njega (desni deo slike).



Slika 37 - Dodavanje novog proračunu kao nova grupa u graf zavisnosti



Slika 38 - Dodavanje novog proračuna u grafa zavisnosti

Za potrebe pseudokoda za dodavanje novog proračuna biće objašnjena jedna funkcija koja se koristi, a to je: *CreateNode*.

Pseudokod za funkciju *CreateNode* je prikazan ispod, gde su ulazni parametri ime proračuna *nF* i čvor v_1, a izlazni parametri izmenjeni čvor v_1 i novododati čvor v_2 koji zavisi od čvora v_1. Najpre se kreira novi čvor v_2 (linija 1), pa zatim se u novokreirani čvor v_2 dodaje novi proračun *nF* (linija 2). Zatim, se čvor v_1 dodaje kao dete čvoru v_2 (linija 3) i na kraju se v_2 dodaje kao roditelj čvor v_1 (linija 4).

function CreateNode (nF, v_1)

Input: nF - name of the new function

v_1 - depended node

Output: v_1 - new node connected to depended

v_2 - new depended node

1 v_2 <- create NEW node
2 ADD nF to Functions in v_2
3 ADD v_2 to Childs in v_1
4 ADD v_1 to Parent in v_2

Pseudokod za dodavanje novog proračuna u sistem za navedene slučajeve se može videti ispod (funkcija *AddNewFunction*).

function AddNewFunction (G, nF, dF)

Input: G = (V, E) - graph, V = {v_1, v_2,…,v_n}, E = {e(v_i,v_j)| $v_i \in V \wedge v_j \in V$ }

nF - name of the new function which should be added

dF - name of the function on which depends newly one

Output: G (V, E) - new graph

1 $v_d \leftarrow$ GET node with dF
2 **IF** (**COUNT** (Functions in v_d) = 1)
3 **IF** (**COUNT** (Childs of v_d) = 1)
4 $v_n \leftarrow$ GET Childs of v_d
5 ADD nF to Functions in v_n
6 **ELSE IF** (**COUNT** (Childs of v_d) $\neq$ 1)
7 $v_n \leftarrow$ CreateNode (nF, v_d)
8 **END IF**
9 **ELSE**
10 REMOVE dF from Functions in v_d
11 $v_p \leftarrow$ GET parent of v_d
12 $v_{dn} \leftarrow$ CreateNode (dF, v_p)
13 $v_n \leftarrow$ CreateNode (nF, v_{dn})
14 **END IF**

Ulazni parametri su graf *G*, naziv novog proračuna *nF* koji treba da se dodaje u stablu i naziv proračuna od koga će novi proračun biti zavisan *dF*. Prvo se iz grafa zavisnosti pronalazi

čvor v_d sa proračunom od koga je zavisan novi proračun (linija 1 u pseudokodu). Najpre se u prvom uslovu proverava da li pronađeni čvor v_d, iliti čvor od koga treba da zavisi novi proračun, ima samo jedan proračun u samom čvoru (linija 2). Ako je prvi uslov zadovoljen i ako čvor v_d ima jedan zavisan čvor (potomak) u stablu (linija 3), onda se pronalazi čvor v_n koji je zavisan od čvora v_d (linija 4) i vrši se dodavanje novog proračuna u postojeći čvor v_n (linija 5 u pseudokodu). U suprotnom ako čvor v_d nema zavisnih čvorova ili ima više od jednog čvora u dijagramu zavisnosti (linija 6), dolazi do pozivanja funkcija *CreateNode* (linija 7) koji kreira novi čvor, dodaje novi proračun *nF* u novom čvoru v_n i vrši se povezivanje v_n sa v_d čvorom. Na kraju, ako prvi uslov nije zadovoljen, što znači da postojeći čvor v_d ima više od jednog proračuna (linija 9), dolazi do brisanja proračuna iz postojećeg čvora (linija 10) i pronalaženja čvora v_p roditelja čvora v_d (linija 11). Zatim, poziva se funkcija *CreateNode* koji kreira novi čvor v_{dn}, dodaje novi proračun *dF* u novokreiranom čvoru v_{dn} i povezuje v_{dn} sa v_p čvorom (linija 12). Na kraju poziva se opet funkcija *CreateNode* koja kreira novi čvor v_n, dodaje se proračun *nF* tom čvoru v_n i povezuje se v_n sa v_{dn} čvorom (linija 13).

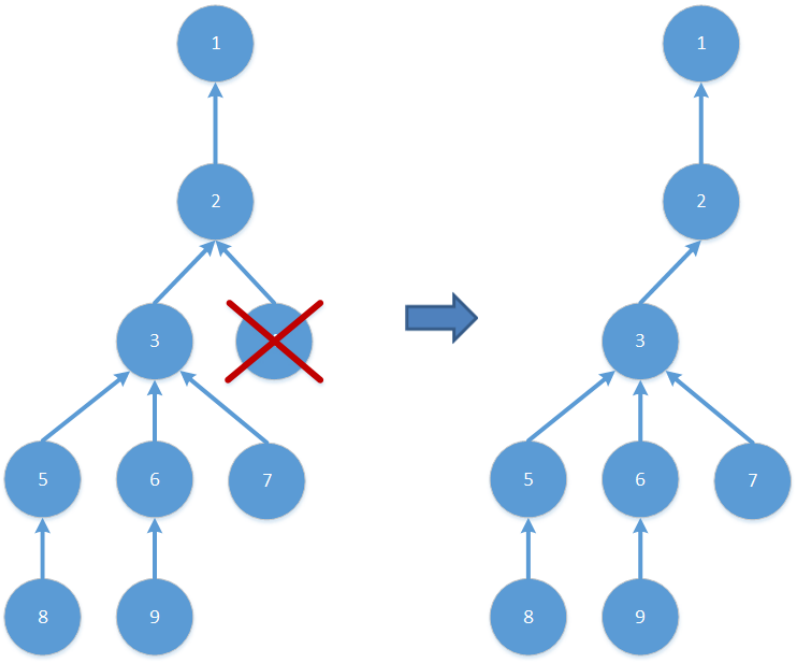
Predloženi dizajn podrazumeva i uklanjanje postojećeg proračuna iz čvora. Uklanjanje proračuna može se raditi samo u slučaju da ako od željenog proračuna ne zavisi nijedan proračun u grafu zavisnosti, u suprotnom uklanjanje neće biti dozvoljeno. Uklanjanje proračuna iz grafa zavisnosti može razlikovati dve situacije:

1. Ako je postojeći proračun u listi čvora.

2. Ako je proračun jedini u listi čvora.

Ukoliko dolazi do uklanjanje proračuna, a proračun se u čvoru nalazi u listi, odnosno ima više proračuna u čvoru, samo se iz postojećeg čvora obriše željeni proračun, a graf zavisnosti ostaje kakav je bio.

Na slici ispod (Slika 39) je prikazano uklanjanje proračuna iz čvora zavisnosti.



Slika 39 - Uklanjanje proračuna i čvora iz grafa zavisnosti

Ukoliko dolazi do uklanjanje proračuna iz određenog čvora, a proračun je jedini u listi i od čvora ne zavisi nijedan drugi proračun, onda dolazi do uklanjanje i proračuna, a i čvora (Slika 39). Na slici iznad (Slika 39) se može videti da se čvor 4 koji sadrži samo jedan proračun, a nema nijednog zavisnog čvora u listi grafa (levi deo slike), obriše se i sam čvor pa graf zavisnosti dobija novi oblik i nove zavisnosti (desni deo slike).

Pseudokod za uklanjanje postojećeg proračuna iz dijagrama se može videti ispod. Prvo se iz grafa zavisnosti pronalazi čvor v_r sa proračunom *rF* koji treba da se obriše (linija 1 u pseudokodu). Ako pronađeni čvor v_r nema zavisnih čvorova u grafu zavisnosti (linija 2), provera se još da li čvor v_r ima više od jednog proračuna (linija 3), dolazi do brisanja proračuna iz čvora v_r (linija 4). U suprotnom, ako čvor v_r ima samo jednog proračuna, vrši se brisanje čvora v_r (linija 5). Na kraju, ako ništa od ovog nije zadovoljeno, odnosno ako čvor v_r ima zavisnih čvorova u djagramu zavisnosti (linija 8), uklanjanje proračuna nije dozvoljeno (linija 9).

function RemoveFunction (G, rF)

Input: G = (V, E) - graph, V = {v_1, v_2,…,v_n}, E = {e(v_i,v_j)| $v_i \in V \wedge v_j \in V$ }

rF - name of the function which should be removed

Output: G (V, E) - new graph

```
1   v_r ← GET node with rF
2   IF (COUNT (Childs of v_r) = 0))
3       IF (COUNT (Functions in v_r) > 1)
4           REMOVE rF from Functions in v_r
5       ELSE
6           REMOVE v_r
7       END IF
8   ELSE
9           Not allowed to remove
10  END IF
```

7.6 Implementacija proračuna na *Service Fabric*

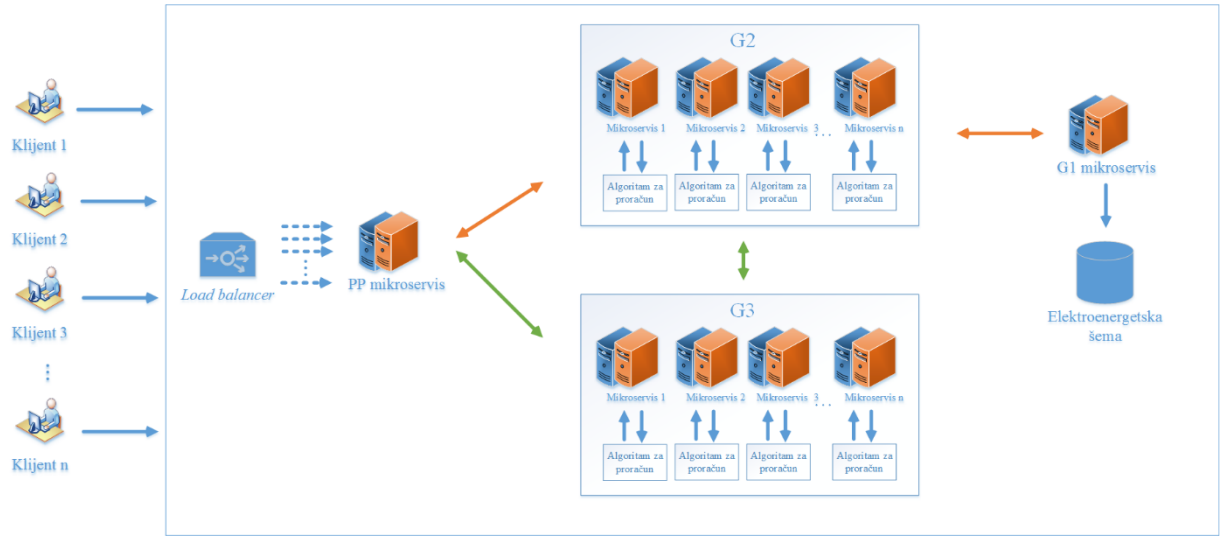
Na osnovu predložene arhitekture prikazane na slici iznad (Slika 29), data je arhitektura sistema sa proračunima u mikroservisnom okruženju, a to su proračuni iz Grupe 2 i 3 (Slika 40) koji će biti implementirani u mikroservisnom okruženju *Service Fabric*. Mikroservisi iz G2 i G3 koji će biti implementirani su: TA mikroservis i TS mikroservis. Oni su implementirani kao *stateful* mikroservisi (vidi Tabelu 2), kako bi kriterijumi same platforme bili ispunjeni (opisano u poglavlju 5.1.1) [103].

Ako uzmemo u obzir primenjenu arhitekturu sa gornje slike (Slika 40), na *Service Fabric* će biti postavljeno pet čvorova po formuli iznad (1), odnosno $U_{mik} = 4$ i $U_{kat} = 3$, tj. po formuli (1) daje pet čvorova, jer se uzima maksimum, a u ovom doktorskoj disertaciji je $n_{min} = 5$. Sami mikroservisi se skaliraju na tih pet čvorova (Slika 41) po datoj formuli iznad (2), gde su G = 5,

parametrar i je promenljiv za svaku kategoriju (vrednost 1 do 5), i $U_{kat} = 3$. Promenjive mik_{min} i n_{min} su već definisane iznad, tj. $mik_{min} = 2$ i $n_{min} = 5$. U tabeli ispod je prikazano broj primarnih servisa za svaku kategoriju mikroservisa (Tabela 5). Za svaki primarni mikroservis biće postavljen i njegov sekundarni.

Tabela 5 - Broj primarnih mikroservisa za kategoriju K1, K4 i K5 na pet čvorova

Kategorija	**K1**	**K4**	**K5**
Broj primarnih mikroservisa	3	2	2



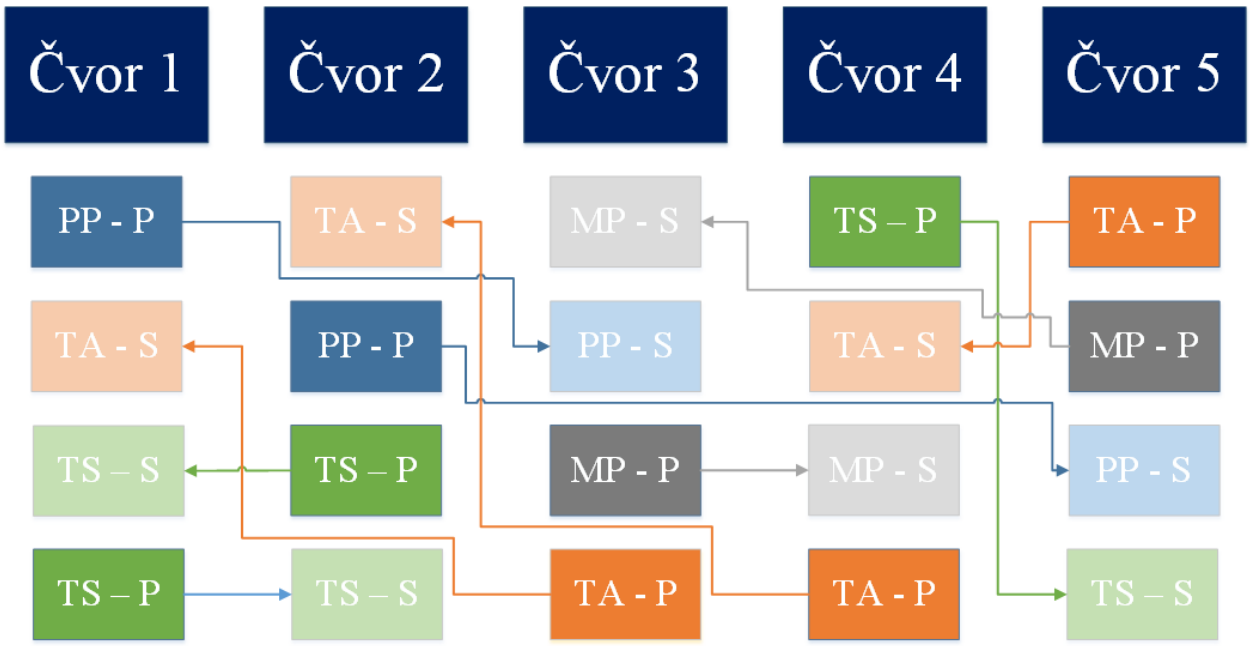
Slika 40 - Primenjena arhitektura proračuna za transformaciju

Prilikom postavljanja aplikacije na *Service Fabric* pet čvorova je konfigurisano, a sami mikroservisi se skaliraju na tih pet čvorova (Slika 41). Sa slike ispod (Slika 41) se mogu videti i zavisnosti primarnih i sekundarnih instanci na čvorovima kako bi se stekao utisak koji sekundarni mikroservis iste instance će preuzeti posao ako dolazi do otkaza odnosno pada nekog čvora. Mikroservisi su na sledeći način konfigurisani po čvorovima:

- Čvor 1 sadrži dve primarne instance mikroservisa i to: PP mikroservis i TS mikroservis, kao i dve sekundarne instance mikroservisa za: TA mikroservis i TS mikroservis.
- Čvor 2 sadrži dve primarne instance mikroservisa i to: PP mikroservis i TS mikroservis, kao i dve sekundarne instance mikroservisa za: TA mikroservis i TS mikroservis.
- Čvor 3 sadrži dve primarne instance mikroservisa i to: MP mikroservis i TA mikroservis, kao i dve sekundarne instance mikroservisa za: MP mikroservis i PP mikroservis.

- Čvor 4 sadrži dve primarne instance mikroservisa i to: TS mikroservis i TA mikroservis, kao i dve sekundarne instance mikroservisa za: TA mikroservis i MP mikroservis.

- Čvor 5 sadrži dve primarne instance mikroservisa i to: TA mikroservis i MP mikroservis, kao i dve sekundarne instance mikroservisa za: PP mikroservis i TS mikroservis.



Slika 41 - Skaliranje mikroservisa aplikacije na *Service Fabric*

Nakon podele svih instanci mikroservisa na svim čvorovima, predloženu arhitekturu za transformaciju proračuna se može transformisati na primenjenu arhitekturu na *Service Fabric*. Pregled svih primarnih i sekundarnih mikroservisa se može videti na slici ispod (Slika 42) i to sledeći servisi sa primarnim i sekundarnim instancama:

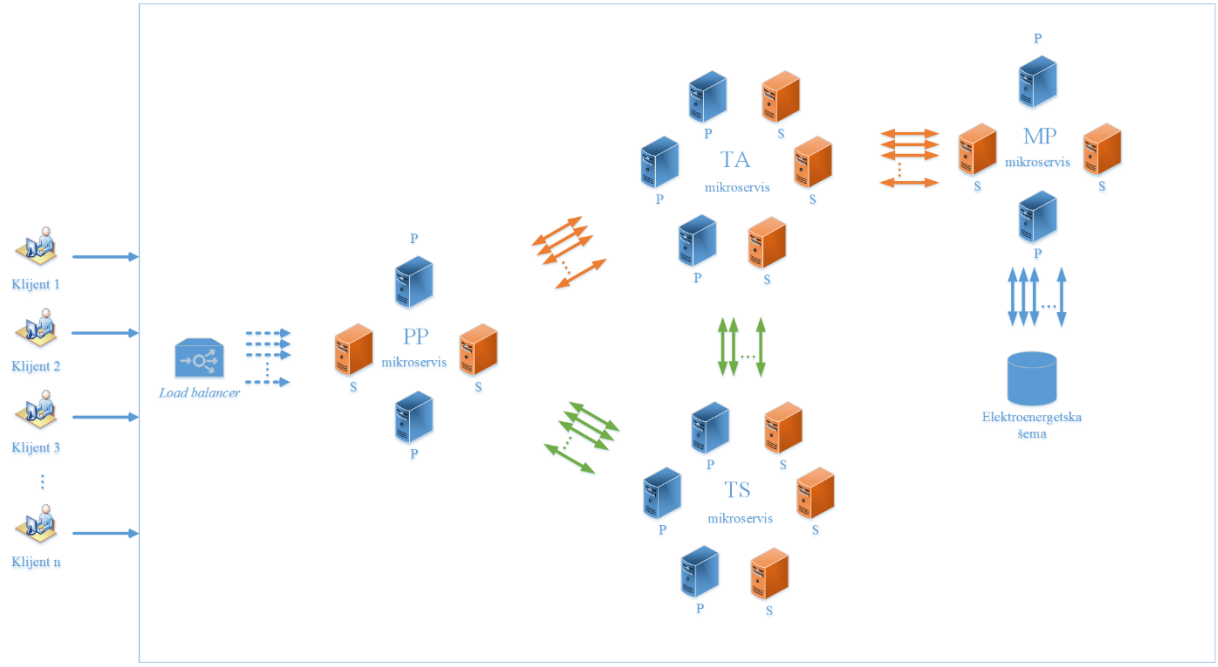
- PP mikroservis nije usko grlo ove same arhitekture iz ugla poslovne logike, pošto je servis bez poslovne logike ali jeste usko grlo aplikacije zbog rutianja poziva ka ostalim mikroservisa. U ovom slučaju PP mikroservis rutira poziv za svaki koren ponaosob na različitim primarnim instancama TA i TS mikroservisa. Ovaj mikroservis je postavljen na četiri čvora, na dva čvora primarni servis, isto važi za sekundarne, koje se skaliraju na preostale čvorove gde se ne nalaze primarne instance.

- MP mikroservis je mikroservis koji radi sa svojom bazom podataka i mikroservis se skalira na tri čvora, jer je mikroservis koji kreira model neophodan za proračune TA i TS. Svi njegovi sekundarni servisi se skaliraju na različitim čvorovima, pa ako nekad dođe do otkaza primarnog mikroservisa, tu je sekundarni gde se nastavlja započeti posao. MP mikroservis sa dve instance je dovoljan jer on ima poslovnu logiku kreiranja modela i podelu kojem će TA i TS mikroservisi da šalje zahtev.

- TA mikroservis je mikroservis tzv kritičnog odziva u samoj arhitekturi i skalira se na svakom pojedinačnom čvoru, pa samim tim primarna instanca TA mikroservisa

se skalira na tri čvora. Takođe, na tri čvora se nalaze i sekundarne instance koja služi tome, da kad na tom pojedinačnom čvoru dolazi do otkaza primarnog TA mikroservisa, sekundarni preuzima i nastavlja sa radom. Želeći da se dobija što bolje skaliranje proračuna, TA mikroservis je i zato postavljen na tri čvora, gde se vrši što veća paralelizacija na osnovu datog korena za proračun. Primera radi ako pristigne zahtev za proračun tri čvora, TA mikroservis će kreirati takav model, za svaki koren ponaosob i aktivirati svih tri instanci TA mikroservisa, gde će svaki vršiti proračun za jedan koren na osnovu pristiglog modela. Na ovaj način se vrši aktiviranje maksimalnih broja TA mikroservisa u zavisnosti od pristiglih zahteva od klijenta i dobija bolja fleksibilnost i ubrzanje što je jako značajno u velikim elektroenergetskim mrežama gde postoje više potrošača.

- TS mikroservis je mikroservis koji na identičan način kao i TA mikroservis primarni mikroservisi se skaliraju na tri čvora. Ovo se radi jer TS kao i TA je važan proračun u elektroenergetskoj mreži. Na ovaj način je obezbeđeno da uvek postoji bar jedan primarni mikroservis iz TS. Ako dolazi do pada jednog čvora, preostaje na ostalim čvorovima da bude uvek aktivan primarni mikroservis dok sekundarni ne preuzima posao pada primarne instance. Ovim se obezbeđuje potrebu svih klijenata da mogu dobiti tražene proračune, u nekom intervalu.



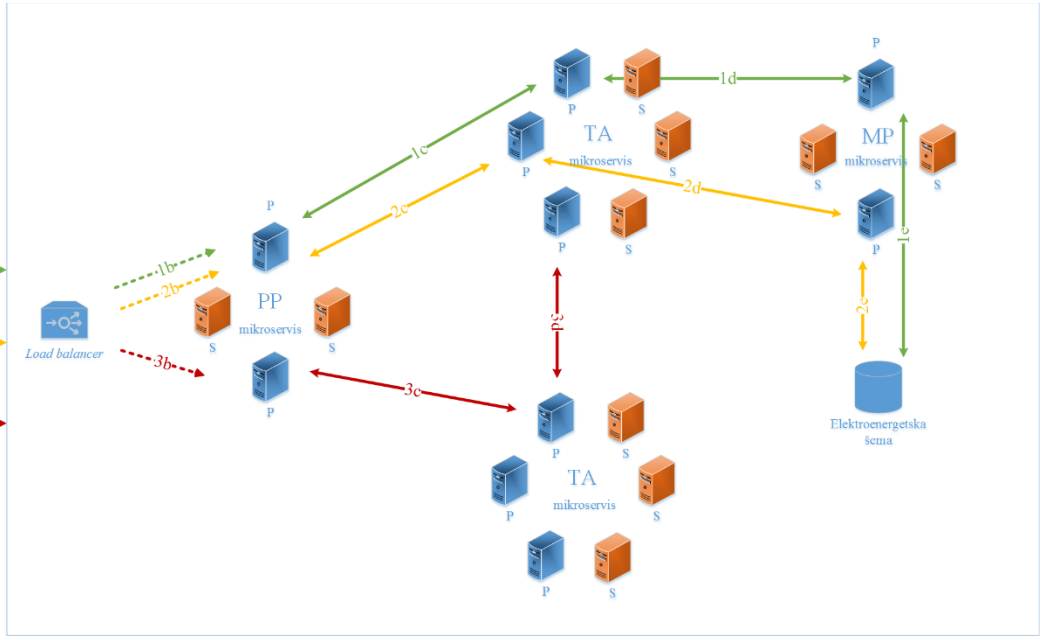
Slika 42 - Implementacija arhitekture na *Service Fabric*

Analizirana su 3 slučaja slanja zahteva klijenata (istovremeno) za date proračune TA i TS. Mikroservisna komunikacija u ovim slučajevima je realizovana korišćenjem WCF komunikacije, ali često se koristi u ovakvim sistemina i srednji sloj Publisher/Subscriber mehanizam [78][79] koji ima svoju internu bazu podataka [80]. S obzirom da se u ovom radu koristi platforma koji je *Service Fabric* i nudi *Reliable* kolekcije, pa samim tim je to iskorišćeno:

1. Klijent 1 šalje zahtev za topološku analizu za korene 1, 2 i 3.
2. Klijent 2 šalje zahtev za topološku analizu za korene 3 i 4.

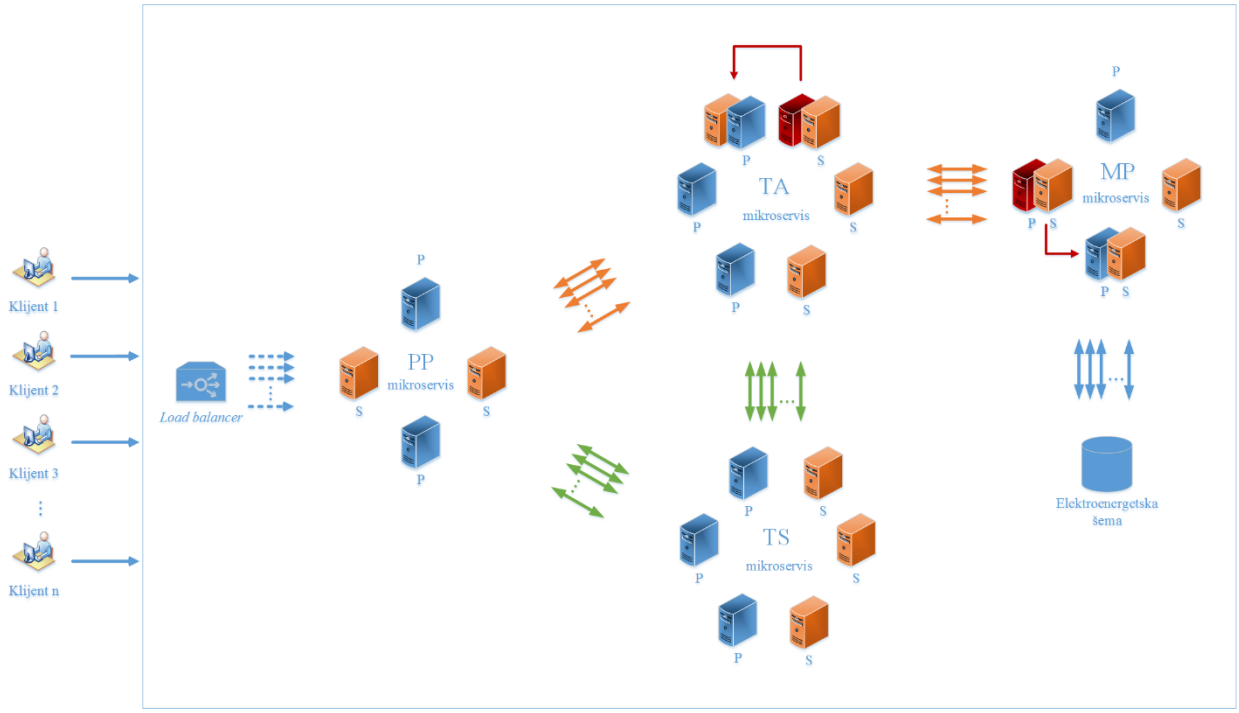
3. Klijent 3 šalje zahtev za tokove snaga za korene 1 i 2.

Klijent 1, klijent 2 i klijent 3 istovremeno šalju 3 zahteva za proračune TA i TS (Slika 43) (zahtev 1a, 1b i 1c). Sa slike iznad (Slika 43) se može videti da zahtev klijenta 1 i klijenta 2 (1b i 2b) *Load balancer* je rutirao na jednu primarnu instancu PP mikroservisa, dok zahtev klijenta 3 (3b) na drugu primarnu instancu PP mikroservisa. Od PP mikroservisa zahtevi (1c i 2c) se šalju na primarne instance TA mikroservisu, dok za zahtev 3c na primarnu instancu TS mikroservisu. TA mikroservis za TA proračun ima potrebu i za modelom elektroenergetske mreže pa se šalje poziv MP mikroservisu. Model se dalje šalje za klijenta 1 (zahtev 1d) na jednu primarnu instancu MP mikroservisa, za klijenta 2 na drugu primarnu instancu MP mikroservisa (zahtev 2d). U MP mikroservisu se vrši priprema modela tako što se čita iz baze elektroenergetska šema (zahtevi 1e za klijenta 1 i 2e za klijenta 2). Takav model se vraća nazad primarnim instancama TA mikroservisa radi TA proračuna. Ako uzmemo u obzir da zahtev klijenta 1 će se prvo obraditi proračun, u TA mikroservisu topološka analiza će odraditi proračun za korene 1, 2 i 3 i nakon obrade se interno čuva u *Reliable* kolekcijama. S obzirom da za klijenta 2 je potreban proračun topološke analize za korene 3 i 4, a za 3 je već odrađen za zahtev klijena 1, mikroservis neće vršiti ponovo proračun nego će se čitati iz *Reliable* kolekcije, dok za koren 4 će se odraditi i nakon obrade upisaće se interno u *Reliable* kolekciji. Svaka instanca istog mikroservisa, u ovom slučaju TA mikroservis ima svoju *Reliable* kolekciju ali nakon upisa u istu, replicira se i na primarne instance i na sekundarne instance. Ovo znači da svaka primarna instanca je svesna podataka u drugim primarnim instancama, tj. u ovom slučaju ako je primarna instanca upisala rezultat TA proračuna za korene 1, 2 i 3, druga primarna instanca kad bude trebala da obradi TA proračun za korene 3 i 4 biće svesna da u *Reliable* kolekciji se već nalazi proračun za koren 3 i na taj način se neće pokrenuti ponovo proračun. Sa druge strane, TS mikroservis prima poziv 3c, i za TS proračun je potreban i rezultat TA i model elektroenergetske mreže pa se šalje zahtev jednoj primarnoj instanci TA mikroservisa (zahtev 3d). TA mikroservis za date korene ima odrađenu topološku analizu, učita iz *Reliable* kolekcije i šalje nazad TS mikroservisu. U TS mikroservisu vrši se proračun tokove snaga za korene 1 i 2 i nakon obrade upisuje se interno u *Reliable* kolekciji.



Slika 43 - Primer slanja 3 klijenta za izvršavanje TA i TS proračuna

Na slici ispod (Slika 44) je prikazano primer otkaza jednog čvora (npr. čvora 3) u toku rada aplikacije na *Service Fabric*. U ovom čvoru se nalazi dve primarne instance mikroservisa i to: MP mikroservis i TA mikroservis, kao i dve sekundarne instance mikroservisa za: MP mikroservis i PP mikroservis. Primarne instance MP i TA mikroservisa se isključe (na slici obeležena instanca crvenom bojom), pa njihove sekundarne instance na čvoru 4 za MP mikroservis (Slika 41), a na čvoru 1 za TA mikroservis (Slika 41) će postati primarne instance (S instanca postaje P instanca), a primarne instance koje su se ugasile će, nakon revitalizacije čvora, postati sekundarne. S obzirom da su i G1, G2 i G3 *stateful* mikroservisi, svaki podatak iz *Reliable* kolekcije će biti aktivan i na sekundarnoj instanci sad kad je postala primarna.



Slika 44 - Pad čvora na *Service Fabric*

Na osnovu predložene arhitekture u mikroservisnom okruženju i njenog korišćenja u *Service Fabric* može se reći da je Hipoteza H2 zadovoljena. Predložena arhitektura ima primenu u DMS sistemu i na *cloud*. Korišćenjem *Service Fabric*-a dolazi do uštede resursa korišćenjem pomenute platforme od njenog postavljanja, do korišćenja samih mašina, skaliranja i particionisanje istih, praćenje sistema, povećanje i smanjenje resursa u zavisnosti od trenutne potrebe.

8. TESTIRANJE I REZULTATI

Od postavke hipoteze do samog eksperimenta koji će ih potvrditi, identifikovane su tri različite grupe i dve podgrupe scenarija sa istim ciljevima i istim testnim scenarijama, a to su:

1. Monolitni scenario - Testiranje tradicionalne monolitne DMS aplikacije sa TA i TS proračunima dato u poglavlju 6.2, i

2. Mikroservisni scenario I - Testiranje DMS sistema u mikroservisnom okruženju sa predlogom arhitekture i proračuna datom u poglavlju 7.6. U ovom scenariju je testiran slučaj ako u *Reliable* kolekcijama nisu sačuvani rezultati proračuna. Cilj ovog proračuna je da se utvrdi rad aplikacije u tako postavljenom slučaju i kako će se TA i TS mikroservisi skalirati.

3. Mikroservisni scenario II - Testiranje DMS sistema u mikroservisnom okruženju sa predlogom arhitekture i proračunom datom u poglavlju 7.6. U ovom scenariju je testiran slučaj ako u *Reliable* kolekcijama jesu sačuvani rezultati proračuna, što je ujedno i svrha nove arhitekture i primene na *Service Fabric*, da ako je neki proračun za dati koren već bio odrađen da se naredni put ne vrši obrada.

Obe grupe opisane iznad se mogu podeliti na još dve pogrupe:

1. Slanje istih korena (KI). Klijenti šalju istovremeno iste korene na izvršavanje proračuna.

2. Slanje različitih korena (KR). Klijenti istovremeno šalju uvek različite korene na izvršavanje proračuna.

Na osnovu grupa i podrupa mogu se izvršiti šest eksperimenata za zahteve topološke analize i tokova snaga odnosno G2 i G3 proračuna kroz celu arhitekturu. Svaki od eksperimenata izvršen je na različitim elektroenergetskim šemama koje su opisane u poglavlju 8.2. Eksperimenti su sledeći:

1. Eksperiment 1 (E1) - Dva klijenta šalju istovremeno zahtev za proračun za iste korene elektroenergetske mreže.

2. Eksperiment 2 (E2) - Dva klijenta šalju istovremeno zahtev za proračun za različite korene elektroenergetske mreže.

3. Eksperiment 3 (E3) - Deset klijenata šalju istovremeno zahtev za proračun za iste korene elektroenergetske mreže.

4. Eksperiment 4 (E4) - Deset klijenata šalju istovremeno zahtev za proračun za različite korene elektroenergetske mreže.

5. Eksperiment 5 (E5) - Dvadeset klijenata šalju istovremeno zahtev za iste korene elektroenergetske mreže.

6. Eksperiment 6 (E6) - Dvadeset klijenata šalju istovremeno zahtev za različite korene elektroenergetske mreže.

Sva merenja su izvršena kako bi se utvrdilo ponašanje sistema sa predloženom arhitekturom kroz sve servise/mikroservise i skaliranje mikroservisa za mikroservisnu grupu eksperimenata. Dobijena vremena su merena od trenutka kada klijent šalje zahtev za dati proračun do vraćanja zahteva sa rezultatom proračuna do samog klijenta. Bitno je napomenuti da je prilikom slanja zahteva za TA proračun odrađen samo G2 mikroservis, i po potrebi se zatraži od G1 mikroservisa model elektroenergetske mreže, ovo je objašnjeno u poglavlju 7.2 i prikazano na dijagramu sekvenci (Slika 30). Zatim, za TS proračun je odrađen i proračuni iz G3 i G2 mikroservisa i po potrebi se zatraži od G1 mikroservisa model elektroenergetske mreže, prikazano na dijagramu sekvenci (Slika 31).

8.1 Testno okruženje

Za testiranje su iskorišćene *Azure Dsv3-series* mašine i to sledeća veličina: *Standard_D16s_v3* za sve eksperimente i sve scenarije, kako bi dokazali hipotezu opisanu u poglavlju 6.4. Karakteristike *Azure* mašine su prikazane u tabeli ispod (Tabela 6) [108].

Tabela 6 - Opis testnog okruženja [108]

Komponenta	**Opis (*eng.)***
PP mikroservisi	*Processor: Intel Xeon Platinum 8272CL (Cascade Lake)*
MP mikroservisi	*vCPU: 16*
TA mikroservisi	*Memory:64GB*
TS mikroservisi	*Temp storage: 128GB*
	Mac data disks: 32
	Max cached and temp storage throughput: IOPS/MBps (cache size in GiB): 32000/256 (400)
	Max burst cached and temp storage throughput: IOPS/MBps: 32000/800
	Max uncached disk throughput: IOPS/MBps: 25600/384
	Max burst uncached disk throughput: IOPS/MBps: 32000/800
	Max NICs/Expected network bandwidth (Mbps): 8/8000
	Operating system: Windows 10 Pro, Version 20H2 – Gen1

8.2 Testne elektroenergetske mreže

Za korišćenje testne elektroenergetske šeme (mreže) za ove eksperimente je iskorišćen alat koji generiše elektroenergetsku mrežu na osnovu datih korena, s obzirom da se proračun topološke analize i tokove snaga obrađuje na osnovu korena. Sa tim u vezi testne šeme su podeljene na tri kategorije:

- Elektroenergetska mreža sa 20 korena,
- Elektroenergetska mreža sa 50 korena, i
- Elektroenergetska mreža sa 100 korena.

Na osnovu ovih kategorija mogu se sagledati potrebe elektrodistributvnih preduzeća koja se razlikuju po broju potrošača.

Jedan od ciljeva ove doktorske disertacije jeste da se pokaže primenjeno rešenje za različite tipove elektroenergetske mreže. U svakoj šemi date elektroenergetske mreže su simulirani razni uređaji i sami potrošači, kao i međusobne veze koje ih po CIM modelu povezuju u jednoj elektroenergetskoj mreži.

8.3 Eksperiment 1

U ovom eksperimentu E1 je simulirano da dva klijenta istovremeno koriste aplikaciju i zatraže TA i TS proračune za određene korene elektroenergetske mreže. Od strane klijenta je uvek poslat zahtev za proračun nad istim korenima elektroenergetske mreže. Isti eksperiment je ponovljen i za monolitni i mikroservisni scenario.

8.3.1 Prikaz rezultata za TA proračun

U tabeli ispod (Tabela 7) su prikazani rezultati za eksperiment (E1), kada dva klijenta u isto vreme šalju zahtev KI za TA proračun. U tabeli su dati rezultati prosečnog vremena izvršavanja eksperimenta, dobijeni njegovim ponavljanjem 100 puta, kao i standardna devijacija datih rezultata.

Tabela 7 - E1 - Rezultati slanja zahteva KI za TA proračun

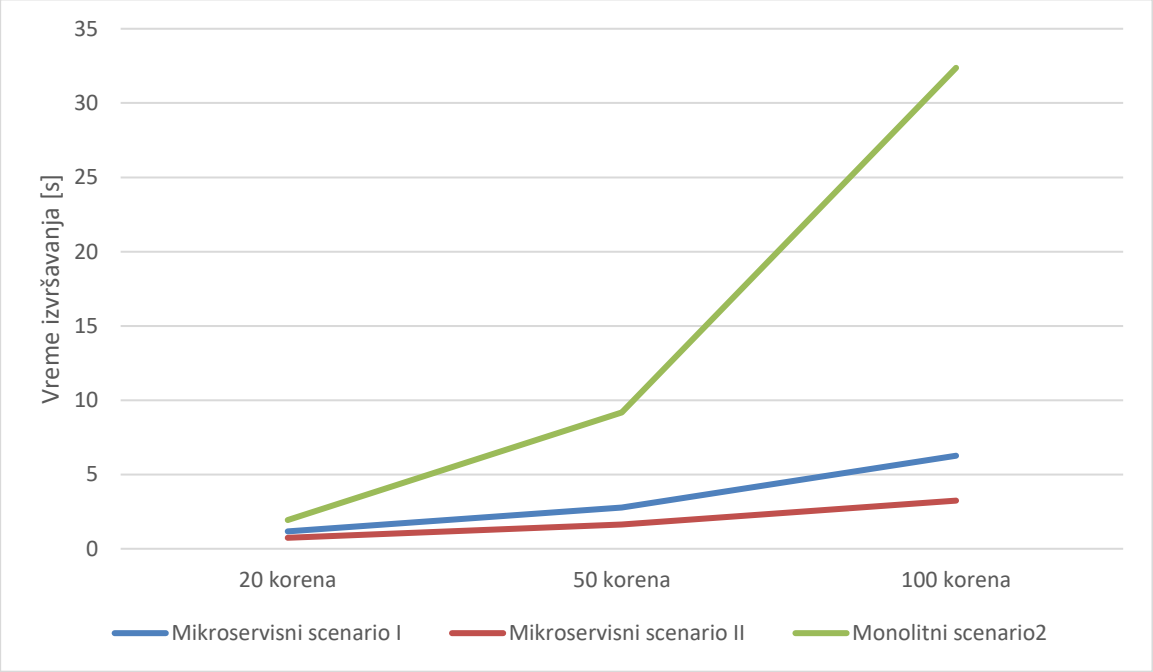
Broj klijenata: 2		**Broj korena**		
		20	**50**	**100**
Mikroservisni scenario I	Avg. [s]	1.170	2.770	6.262
	Dev. [s]	0.047	0.118	1.280
Mikroservisni scenario II	Avg. [s]	0.739	1.641	3.241
	Dev. [s]	0.059	0.317	0.403
Monolitni scenario	Avg. [s]	1.933	9.183	32.370
	Dev. [s]	0.157	0.292	0.602

Grafički prikaz može još preglednije prikazati dobijene rezultate (Slika 45). Na x osi su prikazani dati koreni elektroenergetske mreže, a na y osi je prikazano vreme izvršavanja za dati koren.

Sa povećanjem broja korena elektroenergetske mreže, približno eksponencijalno raste i vreme izvršavanja TA proračuna u testovima na monolitnoj arhitekturi, dok za mikroservisne scenarije linearno raste.

Upoređivanjem Mikroservisnog scenarija I i II, vidimo da u slučaju I traje duže i to za 20 korena 1.6 puta, za 50 korena 1.7 puta i za 100 korena 1.9 puta. Ovo ukazuje na opravdanost primene predložene arhitekture i njeno izvršavanje na *Service Fabric*. Slučajevi u aplikaciji će se najčešće koristiti iz scenarija II, jer ako se neki koren proračuna izračunava, to se čuva u *Reliable* kolekcijama u samom mikroservisu, a sledeći put kad je proračun za taj koren potreban isčitaće se iz *Reliable* kolekcije.

Ako uzmemo i da uporedimo npr slučaj za 20 korena, izvršavanje Monolitnog scenarija traje duže 1.6 puta u poređenju sa Mikroservisnim scenarijom I, dok sa Mikroservisnim scenariom II znatno duže i to 2.6 puta.



Slika 45 - E1 - Odnos rezultata slanja zahteva KI za TA proračun

8.3.2 Prikaz rezultata za TS proračun

Tabela 8 prikazuje rezultate vremena izvršavanja i standardnu devijaciju za E1, odnosno kada dva klijenta u isto vreme šalju zahtev KI za TS proračun.

Iz tabele ispod (Tabela 8) i sa grafika prikazanog na slici ispod (Slika 46) može se uočiti da sa povećanjem broja korena elektroenergetske mreže, približno eksponencijalno raste i vreme izvršavanja TS proračuna u monolitnom scenario, dok za mikroservisne scenarije linearno raste.

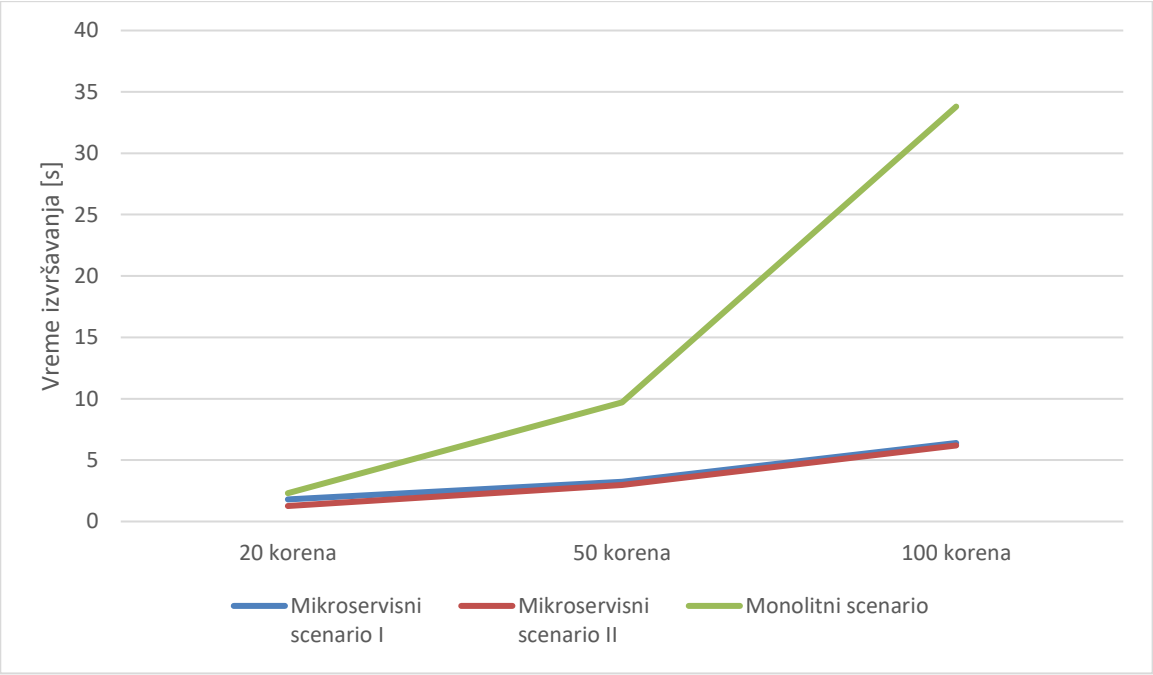
Analizirajući rezultate za Mikroservisni scenario I i II, može se zaključiti da Mikroservisni scenario I za 20 korena traje duže 1.4 puta, dok za 50 1.03 puta i 100 korena približno 1.08 puta,

ali i takva mala razlika je značajna za velike sisteme i evidentna je ušteda u proračunu ako se rezultati proračuna po korenu čuvaju u *Reliable* kolekcijama.

Tabela 8 - E1 - Rezultati slanja zahteva KI za TS proračun

Broj klijenata: 2		**Broj korena**		
		20	**50**	**100**
Mikroservisni scenario I	Avg. [s]	1.800	3.231	6.395
	Dev. [s]	0.916	0.479	0.583
Mikroservisni scenario II	Avg. [s]	1.262	2.976	6.193
	Dev. [s]	0.038	0.136	0.170
Monolitni scenario	Avg. [s]	2.305	9.707	33.801
	Dev. [s]	0.203	0.572	0.664

Ako se uporedi npr. slučaj za 20 korena, izvršavanje Monolitnog scenarija traje duže 1.3 puta u poređenju sa Mikroservisnim scenariom I, dok sa Mikroservisnim scenariom II duže i to 1.8 puta.



Slika 46 - E1 - Odnos rezultata slanja zahteva KI za TS proračun

8.4 Eksperiment 2

E2 simulira na identičan način situaciju kao i E1 samo za različite korene elektroenergetske mreže, odnosno kada dva klijenta istovremeno koriste aplikaciju i zatraže zahtev nad različitim korenima eleketroenergetske mreže za TA i TS proračune. Isti eksperiment je ponovljen i za monolitni i mikroservisni scenario.

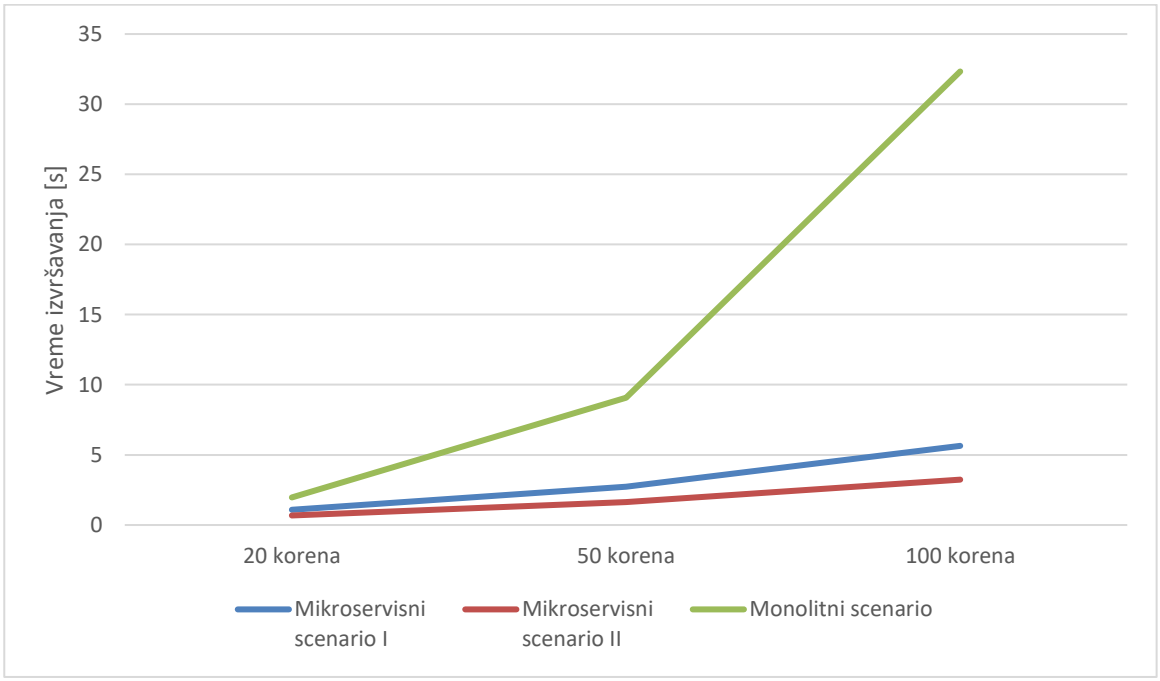
8.4.1 Prikaz rezultata za TA proračun

Rezultati - srednje vrednosti vremena izvršavanja kao i standardna devijacija za E2 tj. kada dva klijenta u isto vreme šalju zahtev KR za TA proračun su prikazani u tabeli ispod (Tabela 9). Svi rezultati koji su ponovljeni 100 puta, kako bi se utvrdilo tačnost istih.

Tabela 9 - E2 - Rezultati slanja zahteva KR za TA proračun

Broj klijenata: 2		**Broj korena**		
		20	**50**	**100**
Mikroservisni scenario I	Avg. [s]	1.088	2.723	5.642
	Dev. [s]	0.071	0.146	0.221
Mikroservisni scenario II	Avg. [s]	0.681	1.631	3.235
	Dev. [s]	0.061	0.054	0.151
Monolitni scenario	Avg. [s]	1.965	9.081	32.330
	Dev. [s]	0.080	0.224	0.526

Na slici ispod (Slika 47) je grafički prikazano odnos Mikroservisnog scenarija I i II i Monolitnog scenarija. Sa povećanjem broja korena elektroenergetske mreže, približno eksponencijalno raste i vreme izvršavanja TA proračuna u monolitnom scenariju, dok za mikroservisne scenarije linearno raste.



Slika 47 - E2 - Odnos rezultata slanja zahteva KR za TA proračun

Iz tabele iznad (Tabela 9) se može zaključiti da Mikroservisni scenario I traje duže od Mikroservisnog scenarija II i to: za 20 korena 1.6 puta, dok za 50 korena 1.7 puta i za 100 korena

1.75 puta, što u je svim slučajevima evidentna razlika i klijenti će biti svesni brzog odziva TA proračuna. U slučaju testova za npr. 50 korena, izvršavanje Monolitnog scenarija traje duže 3.3 puta u poređenju sa Mikroservisnim scenarijom I, dok sa Mikroservisnim scenarijom II duže i to 5.6 puta.

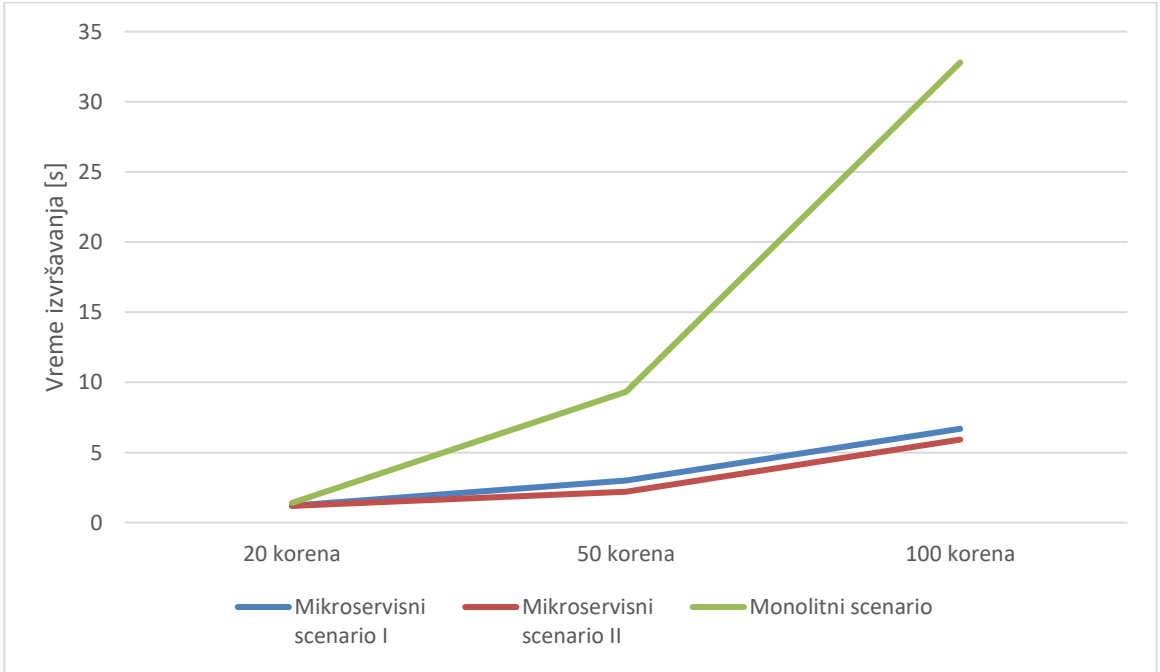
8.4.2 Prikaz rezultata za TS proračun

Tabela 10 prikazuje rezultate srednje vrednosti izvršavanja kao i standardnu devijaciju za E2, tj. kada dva klijenta u isto vreme šalju KR zahtev za TS proračun. Svi rezultati su ponovljeni 100 puta, kako bi se utvrdila tačnost.

Tabela 10 - E2 - Rezultati slanja zahteva KR za TS proračun

Broj klijenata: 2		**Broj korena**		
		20	**50**	**100**
Mikroservisni scenario I	Avg. [s]	1.210	3.014	6.688
	Dev. [s]	0.071	0.143	0.102
Mikroservisni scenario II	Avg. [s]	1.195	2.198	5.921
	Dev. [s]	0.044	0.088	0.448
Monolitni scenario	Avg. [s]	1.407	9.321	32.800
	Dev. [s]	0.175	0.465	0.400

Na slici ispod (Slika 48) je grafički prikazan odnos Mikroservisnog scenarija I i II i Monolitnog scenarija.



Slika 48 - E2 - Odnos rezultata slanja zahteva KR za TS proračun

Vreme izvršavanja Monolitnog scenarija eksponecijalno raste, dok mikroservisni linearno raste, sa povećanjem broja korena po scenarijima.

Analizirajući rezultate za oba mikroservisna scenarija, za 20 korena je Mikroservisni scenario II brži 1.01 puta, za 50 korena 1.4 puta i za 100 korena 1.13 puta od Mikroservisnog scenarija I, što je dokazano da čuvanje u *Reliable* kolekcijama znatno pomaže, da se proračun ne obradi ponovo.

Analizom grafika (Slika 48) i tabele (Tabela 10) npr. testove za 50 korena, vreme izvršavanja Monolitnog scenarija traje duže 3.1 puta u poređenju sa Mikroservisnim scenarijom I, dok sa Mikroservisnim scenarijom II duže i to 4.2 puta.

8.5 Eksperiment 3

E3 simulira slučaj kada deset klijenata istovremeno koriste aplikaciju i zatraže TA i TS proračune za iste korene elektroenergetske mreže. Isti eksperiment je ponovljen i za monolitni i mikroservisni scenario.

8.5.1 Prikaz rezultata za TA proračun

Tabela 11 prikazuje rezultate za E3, odnosno vreme izvršavanja kao i standardnu devijaciju kada deset klijenata u isto vreme šalju KI zahtev za TA proračun. Rezultati predstavljaju prosečne vrednosti dobijene za 100 puta ponovljene testove.

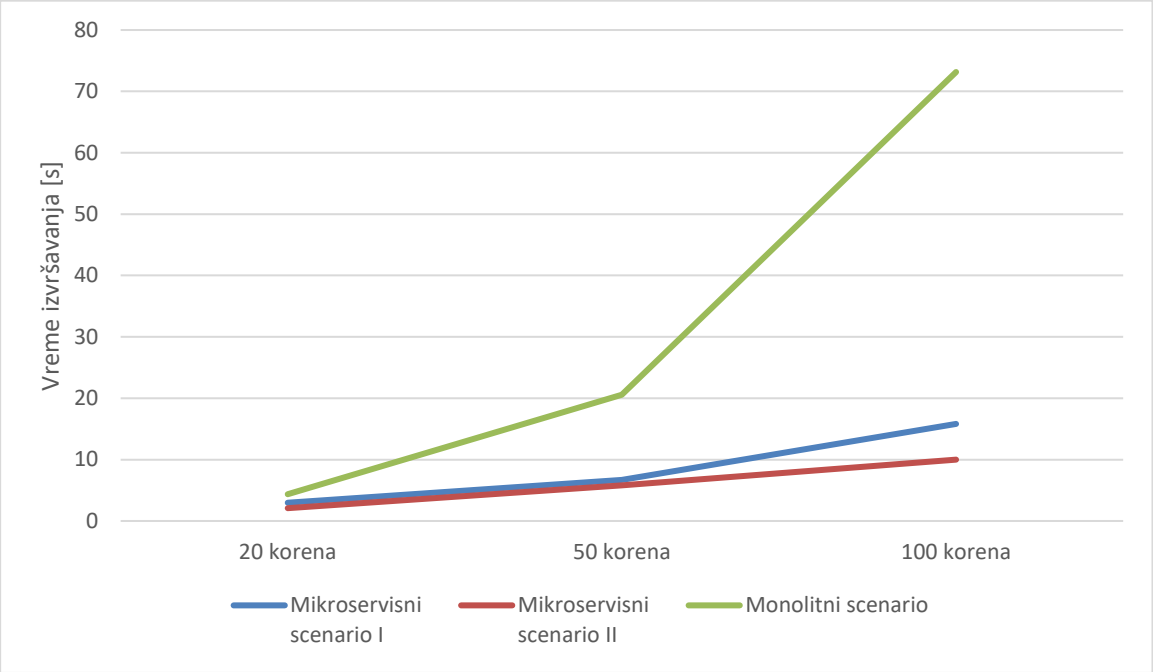
Tabela 11 - E3 - Rezultati slanja zahteva KI za TA proračun

Broj klijenata: 10		**Broj korena**		
		20	**50**	**100**
Mikroservisni scenario I	Avg. [s]	2.975	6.672	18.314
	Dev. [s]	0.510	1.574	2.002
Mikroservisni scenario II	Avg. [s]	2.088	5.803	11.205
	Dev. [s]	0.275	1.579	2.528
Monolitni scenario	Avg. [s]	4.365	20.551	73.151
	Dev. [s]	0.693	1.531	3.919

Grafički prikaz rezultata je predstavljen na slici ispod (Slika 49). Vreme izvršavanja mikroservisnog scenarija eksponencijalno raste, dok kod mikroservisnih scenarija linearno raste sa povećanjem korena elektroenergetske mreže.

Analizom rezultata iz tabele iznad (Tabela 11) za Mikroservisni scenario I i II, može se zaključiti da Mikroservisni scenario I traje duže i to za: 20 korena 1.4 puta, za 50 korena 1.15 puta i za 100 korena 1.6 puta, što je u svim slučajevima značajna razlika, i klijenti će biti svesni brzog odziva TA proračuna.

Posmatranjem dobijenih rezultata za 100 korena, može se utvrditi da izvršavanje Monolitnog scenarija traje duže 4 puta u poređenju sa Mikroservisnim scenariom I, dok sa Mikroservisnim scenariom II duže i to 6.53 puta.



Slika 49 - E3 - Odnos rezultata slanja zahteva KI za TA proračun

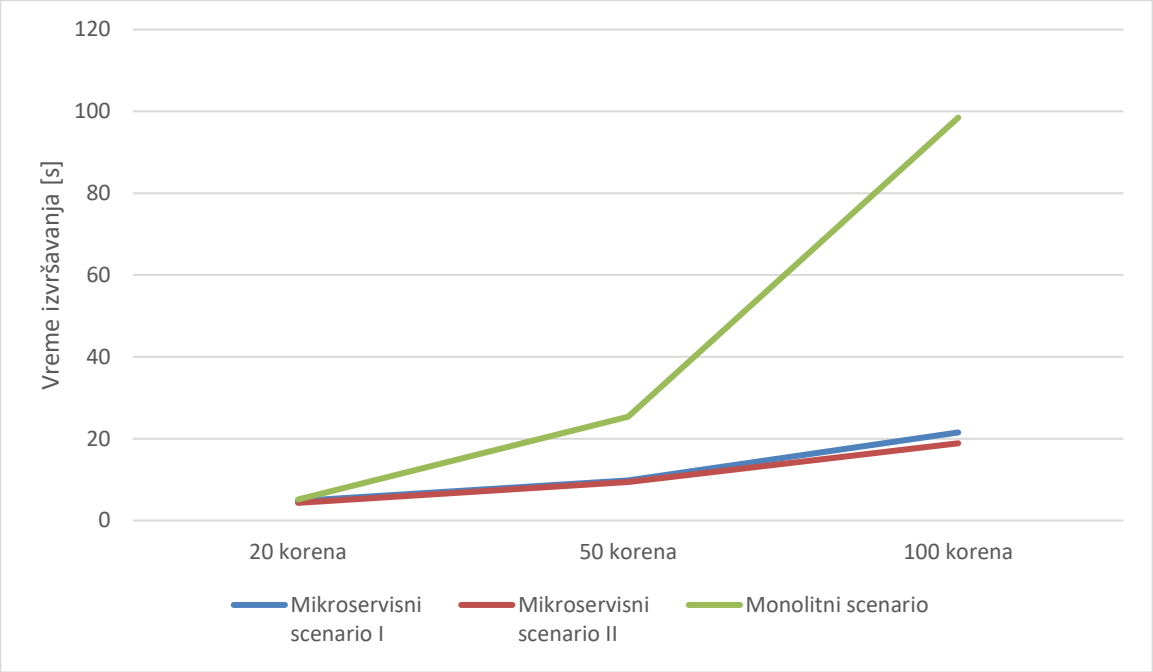
8.5.1 Prikaz rezultata za TS proračun

Tabela 12 prikazuje rezultate za E3, kada deset klijenata u isto vreme šalju zahtev KI za TS proračun. U tabeli su prikazani rezultati prosečne vrednosti vremena izvršavanja i standardne devijacije za 100 puta ponovljene eksperimente.

Tabela 12 - E3 - Rezultati slanja zahteva KI za TS proračun

Broj klijenata: 10		**Broj korena**		
		20	**50**	**100**
Mikroservisni scenario I	Avg. [s]	4.785	9.790	21.529
	Dev. [s]	0.403	0.893	2.372
Mikroservisni scenario II	Avg. [s]	4.325	9.448	18.884
	Dev. [s]	0.618	2.090	3.902
Monolitni scenario	Avg. [s]	5.142	25.361	98.471
	Dev. [s]	0.191	9.005	10.703

Na slici ispod (Slika 50) je prikazan odnos rezultata za sva tri scenarija po korenima. Na x osi je prikazan broj korena elektroenergetske mreže, a na y osi je prikazano vreme izvršavanja za dati koren. Evidentno je da za Monolitni scenario vreme izvršavanja eksponencijalno raste, a za Mikroservisna scenarija linearno raste.



Slika 50 - E3 - Odnos rezultata slanja zahteva KI za TS proračun

Analizom Mikroservisnih scenarija I i II se može zaključiti da Mikroservisni scenario I traje duže od Mikroservisnog scenarija II i to za: 20 korena 1.1 puta, za 50 korena 1.04 puta i za 100 korena 1.14 puta. Razlika je mala i približno ista, i klijenti će biti svesni brzog odziva TS proračuna.

Daljom analizom npr. za slučaj 100 korena, izvršavanje Monolitnog scenarija traje duže 4.6 puta u poređenju sa Mikroservisnim scenariom I, dok sa Mikroservisnim scenariom II duže i to 5.2 puta.

8.6 Eksperiment 4

E4 simulira istu situaciju kao i E3 samo za različite korene elektroenergetske mreže, odnosno kada deset klijenata istovremeno koriste aplikaciju i zatraže zahtev za TA i TS proračune za različite korene elektroenergetske mreže. Isti eksperiment je ponovljen i za monolitni i mikroservisni scenario.

8.6.1 Prikaz rezultata za TA proračun

Rezultati srednje vreme izvršavanja kao i devijacija za rezultate ponovljene 100 puta su prikazane u tabeli ispod (Tabela 13) za E4, kada deset klijenata u isto vreme šalju zahtev KR za TA proračun.

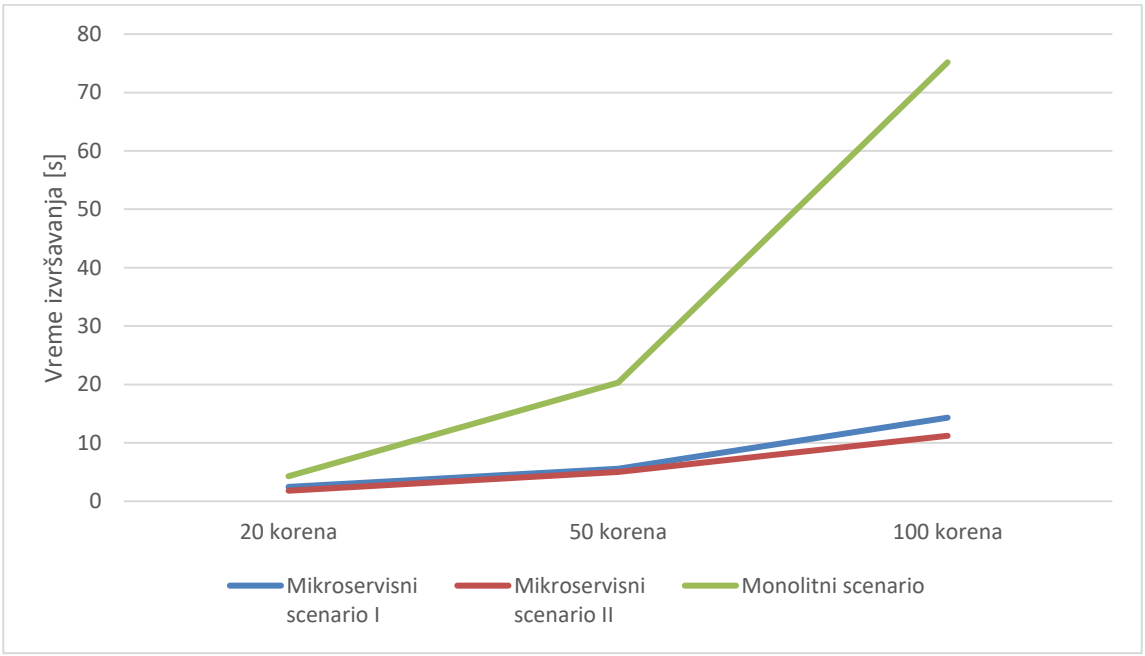
Grafički prikaz rezultata je dat na slici ispod (Slika 51), što pokazuje da za monolitni scenario vreme izvršavanja eksponencijalno raste, dok za mikroservisna scenarija linearno raste sa porastom broja korena elektroenergetske mreže.

Tabela 13 - E4 - Rezultati slanja zahteva KR za TA proračun

Broj klijenata: 10		**Broj korena**		
		20	**50**	**100**
Mikroservisni scenario I	Avg. [s]	2.456	5.577	15.817
	Dev. [s]	0.579	1.559	4.787
Mikroservisni scenario II	Avg. [s]	1.805	5.017	9.998
	Dev. [s]	0.269	1.461	2.096
Monolitni scenario	Avg. [s]	4.280	20.309	75.172
	Dev. [s]	0.645	1.864	5.310

Razmatrajući rezultate iz tabele iznad (Tabela 13) i grafika sa slike (Slika 51) za Mikroservisni scenario I i II, Mikroservisni scenario I traje duže i to za: 20 korena 1.36 puta, dok za 50 korena 1.1 puta i za 100 korena 1.6 puta. Razlika nije mala ni zanemarljiva i korišćenjem aplikacije klijenti će biti svesni ovog ubrzanja.

U slučaju testova za 50 korena, izvršavanje Monolitnog scenarija traje duže 3.6 puta u poređenju sa Mikroservisnim scenariom I, dok sa Mikroservisnim scenariom II duže i to 4.04 puta.



Slika 51 - E4 - Odnos rezultata slanja zahteva KR za TA proračun

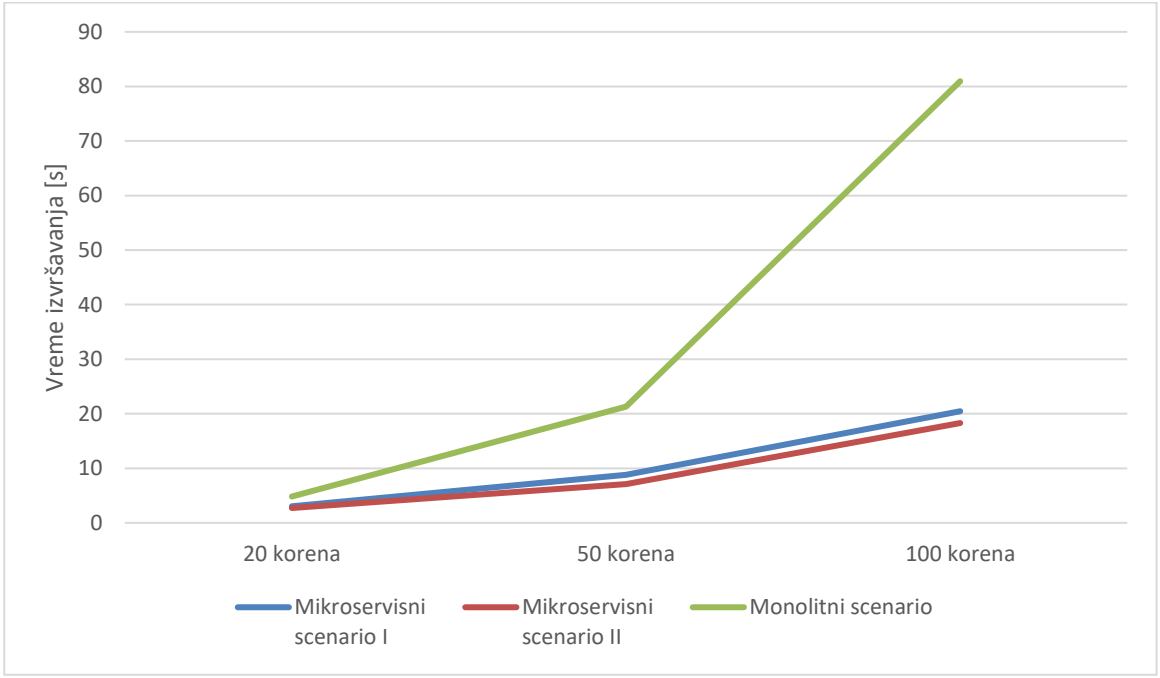
8.6.1 Prikaz rezultata za TS proračun

U tabeli (Tabela 14) su prikazani rezultati za eksperiment 4, kada deset klijenata u isto vreme šalju KR zahtev za TS proračun. U tabeli je prikazano prosečno vreme izvršavanja i standradna devijacija za 100 ponovljenih testova.

Tabela 14 - E4 - Rezultati slanja zahteva KR za TS proračun

Broj klijenata: 10		**Broj korena**		
		20	**50**	**100**
Mikroservisni scenario I	Avg. [s]	3.038	8.826	20.458
	Dev. [s]	0.614	1.558	4.139
Mikroservisni scenario II	Avg. [s]	2.724	7.125	18.295
	Dev. [s]	0.669	1.915	4.055
Monolitni scenario	Avg. [s]	4.828	21.317	80.948
	Dev. [s]	1.822	8.110	10.275

Na slici ispod (Slika 52) je grafički prikazan odnos Mikroservisnog scenarija I i II koji linearno raste i Monolitnog scenarija koji eksponencijalno raste, kako se koreni elektroenergetske mreže povećavaju.



Slika 52 - E4 - Odnos rezultata slanja zahteva KR za TS proračun

Analizirajući rezultate za oba mikroservisna scenarija, za 20 korena je Mikroservisni scenario II brži 1.11 puta, za 50 korena 1.23 puta i za 100 korena 1.12 puta, što je dokazano da čuvanje u *Reliable* kolekcijama omogućuje efikasniji proračun.

Za slučaj 50 korena, izvršavanje monolitnog scenarija traje duže 2.4 puta u poređenju sa Mikroservisnim scenariom I, dok sa Mikroservisnim scenariom II duže i to 3 puta.

8.7 Eksperiment 5

U ovom eksperimentu E5 su simulirani scenariji kada dvadeset klijenata istovremeno koriste aplikaciju i zatraže TA i TS proračun za iste korene elektroenergetske mreže. Isti eksperiment je ponovljen i za monolitni i mikroservisni scenario.

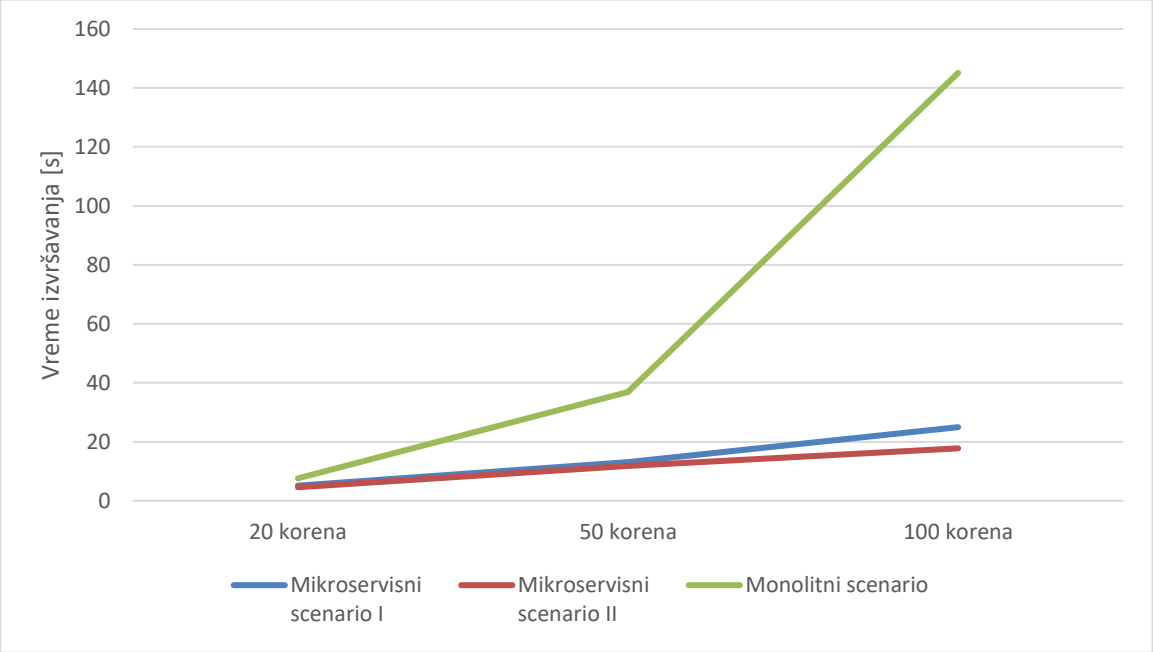
8.7.1 Prikaz rezultata za TA proračun

Srednja vrednost vremena izvršavanja kao i standardna devijacija rezultata za E5, kada dvadeset klijenata u isto vreme šalju zahtev KI za TA proračun su prikazani u tabeli ispod (Tabela 15). Svi testovi su ponovljeni 100 puta.

Tabela 15 - E5 - Rezultati slanja zahteva KI za TA proračun

Broj klijenata: 20		**Broj korena**		
		20	**50**	**100**
Mikroservisni scenario I	Avg. [s]	5.136	13.136	26.394
	Dev. [s]	1.430	3.704	6.376
Mikroservisni scenario II	Avg. [s]	4.595	11.842	18.084
	Dev. [s]	1.131	2.706	5.956
Monolitni scenario	Avg. [s]	7.614	36.866	145.108
	Dev. [s]	1.550	5.849	5.627

Slika 53 prikazuje grafički odnos Mikroservisnog scenarija I i II i Monolitnog scenarija.



Slika 53 - E5 - Odnos rezultata slanja zahteva KI za TA proračun

Povećanjem broja korena elektroenergetske mreže, približno eksponencijalno raste i vreme izvršavanja i TA proračuna za Monolitni scenario, dok za Mikroservisne scenarije linearno raste. Mikroservisni scenario I u odnosu na Mikroservisni scenario II traje duže za 20 korena 1.12 puta, dok za 50 korena 1.1 puta i za 100 korena 1.4 puta. Za slučaj 50 korena, izvršavanje Monolitnog scenarija traje duže 2.8 puta u poređenju sa Mikroservisnim scenariom I, dok sa Mikroservisnim scenariom II duže i to 3.11 puta.

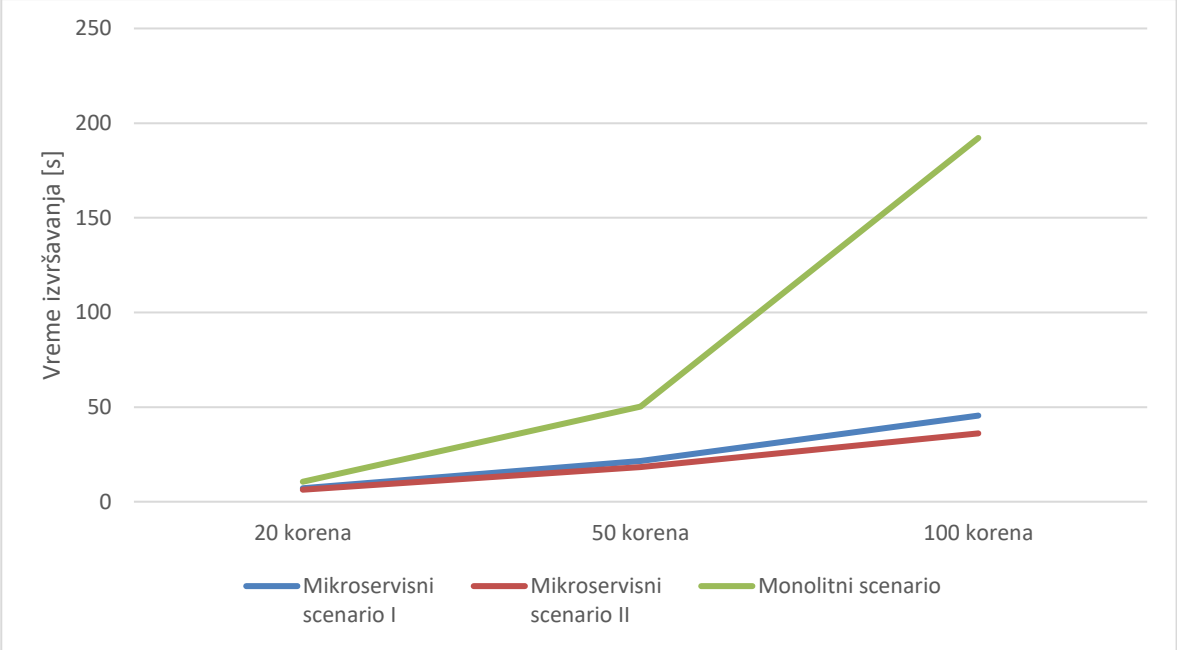
8.7.1 Prikaz rezultata za TS proračun

U tabeli ispod (Tabela 16) su prikazani rezultati srednje vrednosti vreme izvršavanja kao i standardna devijacija za 100 ponovlljenih testova za E5, kada dvadeset klijenata u isto vreme šalju KI zahtev za TS proračun.

Tabela 16 - E5 - Rezultati slanja zahteva KI za TS proračun

Broj klijenata: 20		**Broj korena**		
		20	**50**	**100**
Mikroservisni scenario I	Avg. [s]	7.121	21.463	45.501
	Dev. [s]	1.075	5.249	8.308
Mikroservisni scenario II	Avg. [s]	6.352	18.337	36.088
	Dev. [s]	1.720	6.026	9.744
Monolitni scenario	Avg. [s]	10.559	50.215	192.160
	Dev. [s]	5.179	20.345	64.688

Na slici ispod (Slika 54) i grafički su prikazani dobijeni rezultati za sva tri scenarija.



Slika 54 - E5 - Odnos rezultata slanja zahteva KI za TS proračun

Iz grafika (Slika 54) može zaključiti da vreme izvršavanja linearno raste za Mikroservisna scenarija, a za Monolitnog scenarija eksponencijalno raste kako se broj korena elektroenergetske mreže povećavaju.

Analizom rezultata i poređenjem Mikroservisnih scenarija I i II, može se utvrditi da Mikroservisni scenario I traje duže i to za: 20 korena 1.12 puta, 50 korena 1.17 puta i 100 korena 1.26 puta.

Ako se uporede rezultati za npr. 50 korena za Monolitnog i Mikroservisna scenarija može se zaključiti da vreme izvršavanja Monolitnog scenarija traje duže 2.34 puta u poređenju sa Mikroservisnim scenariom I, dok sa Mikroservisnim scenariom II duže i to 2.73 puta.

8.8 Eksperiment 6

E6 je eksperiment koji na sličan način kao i E5 simulira scenarije samo što za rad aplikacije, dvadeset klijenata uvek šalju zahtev za različite korene elektroenergetske mreže za TA i TS proračune. Ovi testovi su ponovljeni i za monolitni i za mikroservisni scenario.

8.8.1 Prikaz rezultata za TA proračun

Vreme izvršavanja kao i standardna devijacija za testove ponovljene 100 puta su prikazani u tabeli ispod (Tabela 17) za slučaj kada dvadeset klijenata u isto vreme šalju KR zahtev za TA proračun.

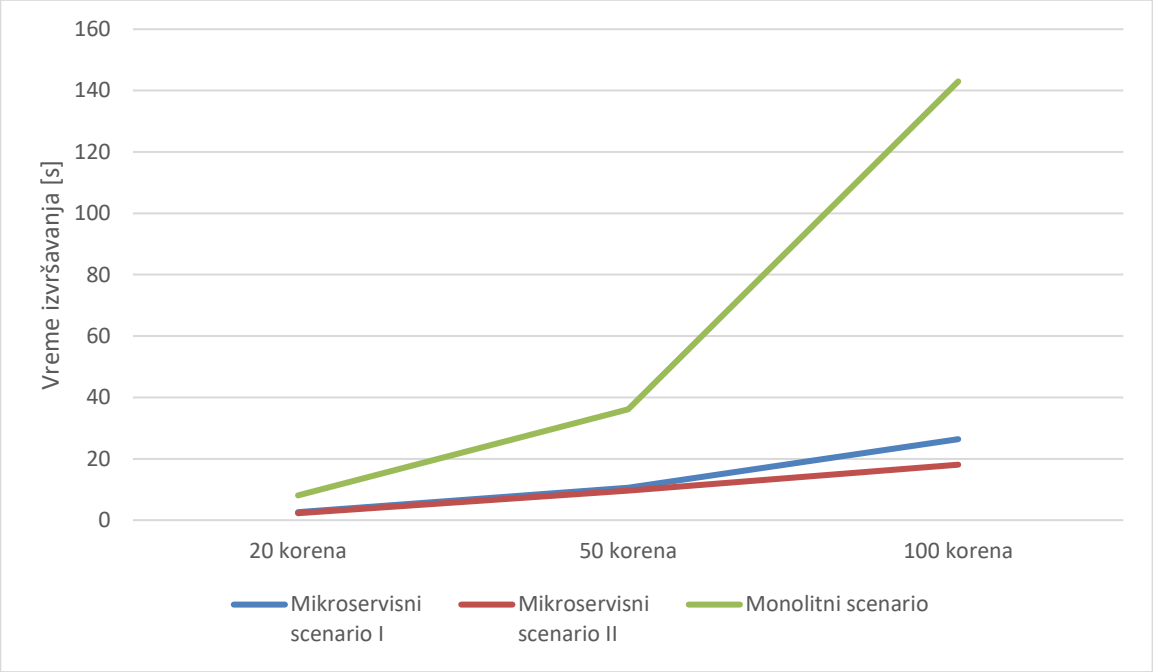
Tabela 17 - E6 - Rezultati slanja zahteva KR za TA proračun

Broj klijenata: 20		**Broj korena**		
		20	**50**	**100**
Mikroservisni scenario I	Avg. [s]	2.612	10.602	24.969
	Dev. [s]	0.883	3.058	9.175
Mikroservisni scenario II	Avg. [s]	2.275	9.600	17.782
	Dev. [s]	0.600	3.204	6.075
Monolitni scenario	Avg. [s]	8.076	36.130	142.965
	Dev. [s]	1.586	5.630	8.710

Grafički prikazani rezultati za sva tri scenarija (Slika 55) ukazuju da vreme izvršavanja Mikroservisnih scenarija linearno raste, dok za Monolitnog scenarija eksponencijalno raste sa povećanjem broja korena elektroenergetske mreže.

Analizom tabele (Tabela 17) i grafika (Slika 55) mogu se porediti rezultati za Monolitni scenario I i II. Monolitni scenario I traje duže za 20 korena 1.14 puta, dok za 50 korena 1.10 puta i za 100 korena 1.4 puta, što u svim slučajevima je razlika značajna, i klijenti će biti svesni brzog odziva TA proračuna.

U slučaju npr. 100 korena, izvršavanje Monolitnog scenarija traje duže 5.7 puta u poređenju sa Mikroservnisnim scenariom I, dok sa Mikroservisnim scenariom II duže i to 8.03 puta.



Slika 55 - E6 - Odnos rezultata slanja zahteva KR za TA proračun

8.8.2 Prikaz rezultata za TS proračun

Srednje vrednosti vremena izvršavanja i standardne devijacije rezultata testova E6 koji su ponovljeni 100 puta su prikazani u tabeli ispod (Tabela 18). Slučaj koji je simuliran je kada dvadeset klijenata u isto vreme šalju KR zahtev za TS proračun.

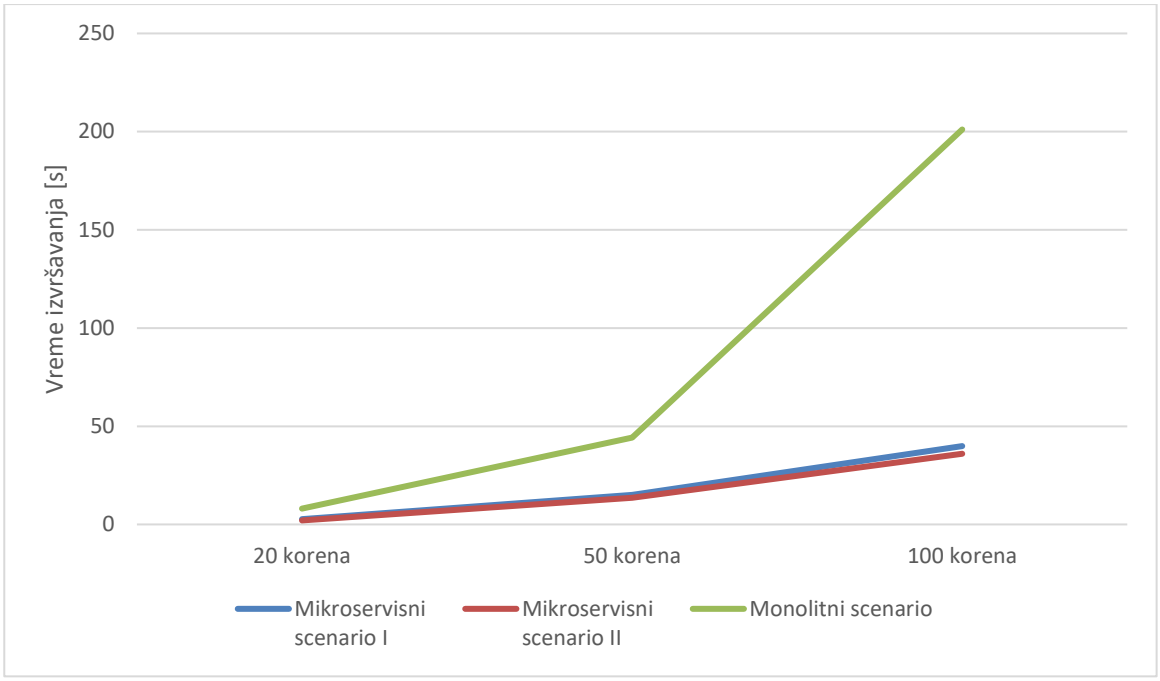
Tabela 18 - E6 - Rezultati slanja zahteva KR za TS proračun

Broj klijenata: 20		**Broj korena**		
		20	**50**	**100**
Mikroservisni scenario I	Avg. [s]	2.693	15.124	39.886
	Dev. [s]	0.621	3.734	9.814
Mikroservisni scenario II	Avg. [s]	2.043	13.499	35.989
	Dev. [s]	0.745	3.468	6.378
Monolitni scenario	Avg. [s]	8.064	44.293	201.095
	Dev. [s]	3.927	7.064	12.549

Grafički prikazan odnos Mikroservisnog scenarija I i II koji linearno rastu i Monolitnog scenarija koji eksponencijalno raste, kako se brojevi korena elektroenergetske mreže povećavaju je dat na slici ispod (Slika 56).

Analizom rezultata Mikroservisnih scenarija iz tabele i grafika, može se uočiti odnos da Mikroservisni scenario I traje duže od Mikroservisnog scenarija II i to za: 20 korena 1.3 puta, za 5 korena 1.12 puta, dok za 100 korena 1.11 puta.

Daljom analizom, ako se uporedi npr slučaj za 100 korena, izvršavanje Monolitnog scenarija traje duže 5.04 puta u poređenju sa Mikroservisnim scenariom I, dok sa Mikroservisnim scenariom II duže i to 5.6 puta.



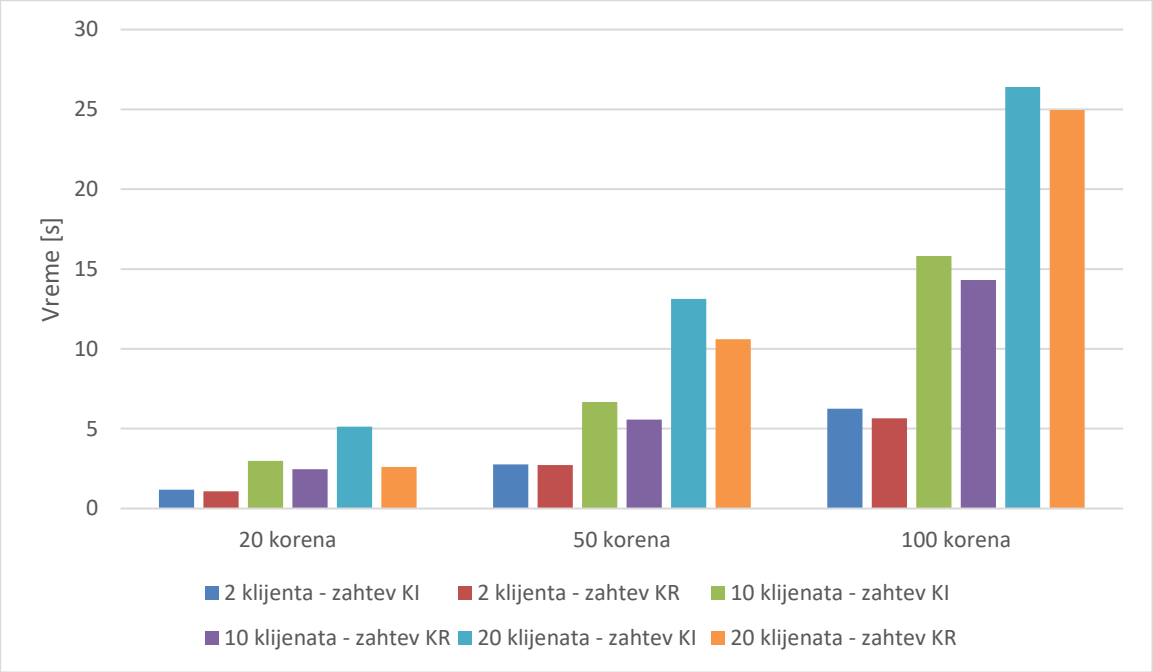
Slika 56 - E6 - Odnos rezultata slanja zahteva KR za TS proračun

8.9 Upoređenje rezultata eksperimenata za TA proračun

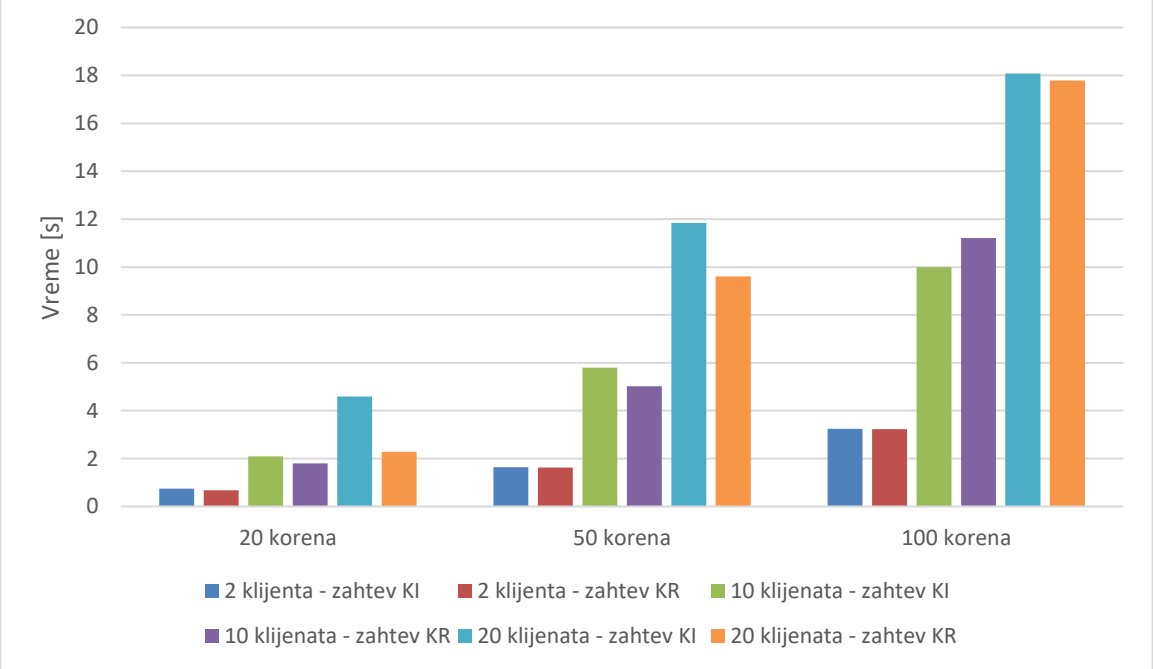
Na osnovu rezultata za Mikroservisni scenario I i II (Slika 57 i Slika 58) može se primetiti da kako broj klijenata raste, izvršavanje TA proračuna traje duže. Naravno, za veće elektroenergetske mreže, sa većim brojem korena, izvršavanje zahteva za TA proračunom kroz celu arhitekturu traje duže nego za elektroenergetske mreže sa manjim brojem korena. Takođe, može se zaključiti da za određenu elektroenergetsku mrežu povećavanje broja klijenata dovodi do porasta srednje vrednosti vremena izvršavanja.

Upoređivanjem rezultata za Mikroservisni scenario I za 10 i 20 klijenata, 100 korena za zahtev KI, može se zaključiti da vremena izvršavanja TA proračuna za 20 klijenata su veći za 40% nego za 10 klijenata (Slika 57). Takođe, za iste klijente ali zahtev KR, obrada zahteva za 20 klijenata je veća za 42% nego za 10 klijenata (Slika 57).

Pored toga, ako se uporede rezultati za Mikroservisni scenario II za 10 i 20 klijenata, za zahtev KI odnosno slanja 100 istih korena sa klijenata, može se zaključiti da su performanse za 20 klijenata za 44% lošije nego za 10 klijenata (Slika 58). U slučaju slanja zahteva KR, može se zaključiti da su vremena obrade zahteva za 20 klijenata veći za 37% nego za 10 klijenata (Slika 58).



Slika 57 - Ukupni rezultati TA proračuna za Mikroservisni scenario I



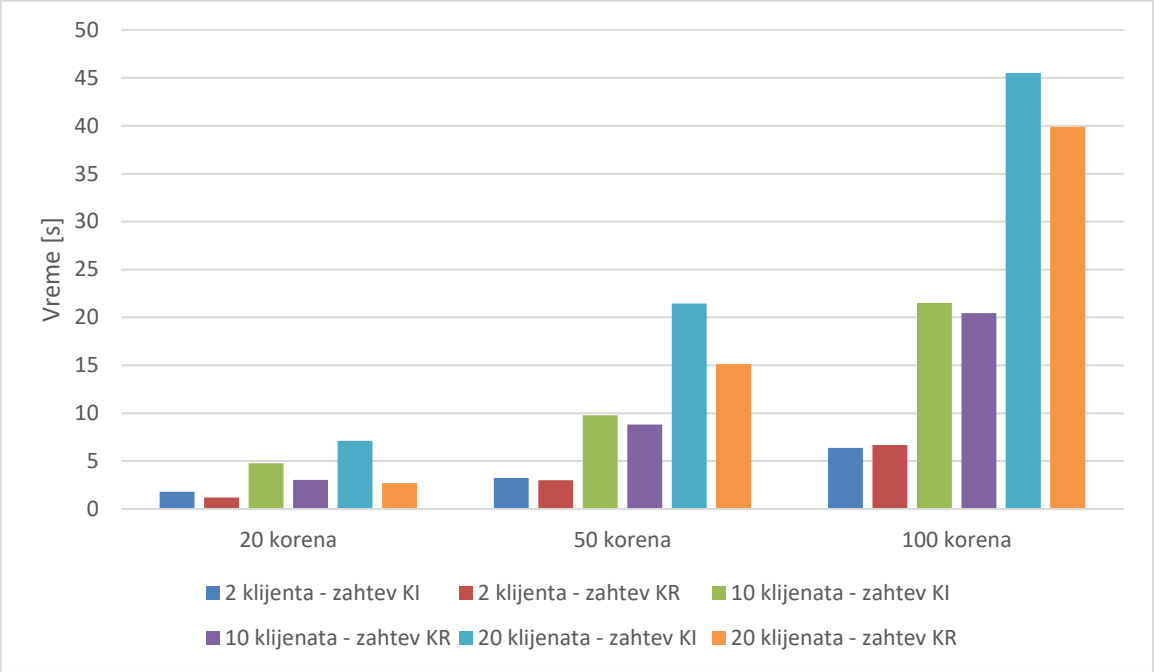
Slika 58 - Ukupni rezultati TA proračuna za Mikroservisni scenario II

Analizom rezultata može se zaključiti da je ovakvo vreme izvršavanja očekivano zbog samih scenarija testa, imajući u vidu da se za 20 klijenata kreira 20 niti, koje izvršavaju zahtev TA proračuna kroz celu arhitekturu, dok se u slučaju 10 klijenata kreira 10 niti. Sve ove niti se šalju na tri instance TA mikroservisa, koje su postavljane na različitim čvorovima na *Service Fabric*. Povećanjem broja klijenata koji simuliraju datu aplikaciju, povećava se i vreme izvršavanja zahteva TA proračuna elektroenergetske mreže.

Daljom analizom E1-E6, za slučaj slanja zahteva KI i KR za TA proračun i poređenjem Monolitnog i Mikroservisnog scenarija, može se zaključiti da Mikroservisni scenariji imaju bolje performanse čime je hipoteza H3 dokazana.

8.10 Upoređenje rezultata eksperimenata za TS proračun

Analizirajući rezultate za Mikroservisni scenario I i II za TS proračun (Slika 59 i Slika 60) može se zaključiti da kako broj klijenata raste, izvršavanje TS proračuna traje duže. Ako se testiranje vrši na elektroenergetskoj mreži sa većim brojem korena, izvršavanje zahteva za TS proračun traje duže nego za elektroenergetske mreže sa manjim brojem korena. Takođe, povećanje broja klijenata dovodi do porasta vremena izvršavanja zahteva.



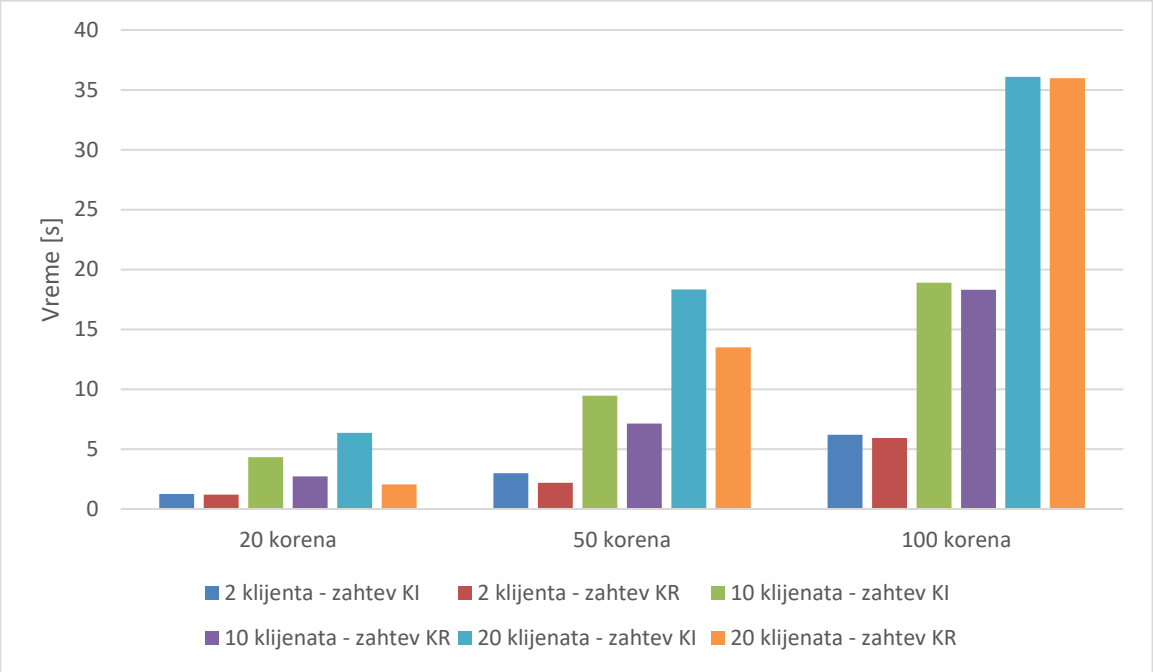
Slika 59 - Ukupni rezultati TS proračuna za Mikroservisni scenario I

Upoređivanjem rezultata za Mikroservisni scenario I za 10, 20 klijenata i 100 korena za KI zahtev, može se zaključiti da su vremena obrade zahteva za 20 klijenata veća za 53% nego za 10 klijenata (Slika 59). Uz to, rezultat za iste klijente i iste korene, ali slanjem zahteva KR, može se zaključiti da su performanse za 20 klijenata za 49% lošije nego za 10 klijenata (Slika 59).

Ukoliko se uporede dobijeni rezultati za Mikroservisni scenario II za 10, 20 klijenata i 100 korena za slučaj slanja zahteva KI za TA proračune, može se zaključiti da su performanse za 20 klijenata za 48% lošije nego za 10 klijenata (Slika 60). Takođe, rezultate dobijeni za slanje zahteva KR za TA proračune, pokazuju da su vremena izvršavanja za 20 klijenata veća za 49% nego za 10 klijenata (Slika 60).

Dobijena vremena izvršavanja su i očekivana zbog samih scenarija testa, s obzirom da se za 20 klijenata kreira 20 niti, koje izvršavaju zahtev TS proračuna kroz celu arhitekturu, dok se u slučaju 10 klijenata kreira 10 niti. Ove niti se šalju na sve tri instance TS mikroservisa, koje su postavljane na različitim čvorovima na *Service Fabric*. Povećanjem broja klijenata koji simuliraju datu aplikaciju, povećava se i vreme izvršavanja zahteva TS proračuna.

Analizom E1-E6, za slučaj slanje zahteva KI i KR za TS proračun i poređenjem Monolitnog i Mikroservisnog scenarija, može se zaključiti da Mikroservisni scenariji daju bolje rezultate vremena izvršavanja TS proračuna čime je hipoteza H3 dokazana.



Slika 60 - Ukupni rezultati TS proračuna za Mikroservisni scenario II

9. ZAKLJUČAK

Električna energija je jedan od najznačajnih oblika energije današnjice. Potreba za optimalnim korišćenjem električne energije, kroz proces nadgledanja i upravljanja, raste iz dana u dan. U tu svrhu, u poslednje dve decenije je intenzivirano korišćenje DMS sistema, softvera za upravljanje i nadgledanje distribucije električne energije u elektroenergetskim mrežama. Tradicionalni DMS sistemi su zasnovani na monolitnoj arhitekturi, gde su servisi usko povezani i svaka izmena funkcionalnosti u jednom servisu utiče i na druge servise, čak i kada se ti drugi servisi nepromenjeni. Ovo znači da su servisi međusobno zavisni i uvek moraju da se korisnicima isporuče zajedno, kao i da detalji implementacije jednog servisa nisu skriveni od drugog servisa. Takođe, kod takvog rešenja se u svakom servisu koristi ista tehnologija i nema mogućnosti korišćenja raznovrsnosti tehnologija. Drugi problem arhitekture tradicionalnog DMS sistema i rešenja jeste potreba za njegovom elastičnošću i skaliranjem. Monolitni servisi se skaliraju zajedno i nema mogućnosti skaliranja samo pojedinih servisa. Elastičnost je skoro nemoguća u monolitnoj arhitekturi, jer kao što je gore navedeno svi servisi su međusobno zavisni pa ako dolazi do pada jednog servisa, dolazi do pada cele aplikacije. Još jedan od ozbiljnih problema takvog tradicionalnog sistema jeste hardver. Dokazano je da se potreba za hardverom menja prilikom korišćenja aplikacije. U jednom trenutku je npr. potrebno više hardvera, dok u drugom trenutku znatno manje. Ovo dovodi nekad i do neiskorišćenog resursa sa kojim se kompanije (korisnici) susreću. Primera radi, ako je za neku aplikaciju potrebno deset servera, i u toku rada se aplikacija manje koristi, odnosno ima manje klijenata koji koriste aplikaciju, broj servera gde se aplikacija postavlja se smanjuje. Kompanije taj hardver ne koriste, odnosno nemaju potrebe za tim i on se posmatra kao neiskorišćeni resurs.

Istraživanje u okviru ove disertacije se baziralo na gore navedenim problemima sa kojima se susreću tradicionalni DMS sistemi i njihovi analitički proračuni elektroenergetske mreže. Na početku istraživanja formulisane su hipoteze koje su i dokazane kroz celokupan proces istraživanja. Problemi koji se javljaju u okviru monolitne aplikacije, a odnose se na: postavljanje aplikacije, skrivanja detalja implementacije svakog servisa, raznovrsnost tehnologije, elastičnost, skaliranje kao i potreba za korišćenjem resursa, rešava *cloud* korišćenjem mikroservisne arhitekture. *Cloud* pruža mogućnost da se koristi samo resurs koji je u tom trenutku neophodan, što ne dovodi do neiskorišćenog resursa.

Sve ovo dovelo je do arhitekture koja je predložena i prikazana u ovom istraživanju, a zasniva se na mikroservisima za analitičke elektroenergetske proračune DMS sistema. Arhitektura je tako napravljena da podrži elektroenergetske proračune po grupama, tj. proračuni su podeljeni po grupama i postoji mogućnost za dodavanje novog proračuna i brisanje postojećeg proračuna iz stabla arhitekture. Ovakva arhitektura otvara mogućnost njenog korišćenja i za druge elektroenergetske proračune koje se pojavljuju kao novi zahtevi, jer kako je i rečeno DMS sistem raste iz dana u dan sa potrebama klijenata i tržista. Tako predložena *cloud* bazirana

arhitektura se može koristi i primeniti na bilo koju mikroservisnu platformu i na taj način zadovoljiti potrebe samih klijenata.

Za primenu i testiranje predložene arhitekture je iskorišćena platforma *Service Fabric,* kao i dva proračuna iz dve grupe (G2-topološka analiza i G3-tokovi snaga). Ova dva proračuna su izabrana kao dva najznačajnija, odnosno, najčešće korišćena proračuna elektroenergetske mreže, jer od ovih proračuna zavisi većina ostalih analitičkih elektroenergetskih proračuna, pa i sam rad DMS sistema bez ova dva proračuna nije moguć. Takođe, u istraživanju su date i matematičke formule za proračunavanje broja čvorova i broja primarnih i sekundarnih instanci svakog tipa mikroservisa, u zavisnosti od broja eletroenergetskih proračuna koji su potrebni za postavljanje u rad aplikacije.

Rezultati svih eksperimenata i svih scenarija ukazuju na to da primena arhitekture u mikroservisnom okruženju daje bolje rezultate sa aspekta vremena izvršavanja elektroenergetskih proračuna u odnosu na monolitni scenario. Prilikom slanja zahteva za proračunom topološke analize rezultati mikroservisnog scenarija su bolji od 1.4 puta do skoro 10 puta u odnosu na monolitni scenario, a prilikom slanja zahteva za proračunom tokova snaga rezultati mikroservisnog scenarija su bolji od 1.07 do 6 puta u odnosu na monolitni scenario. Na ovaj način, primene *cloud*-a i mikroservisnog okruženja pored mnogobrojnih prednosti doprinosi i poboljšanju performansi rada aplikacije i dodatno ubrzava rad aplikacije sa elektroenergetskim proračunima, što dovodi do efikasnijeg rada klijenata. Ovo se postiže tako što se rezultat proračuna za svaki koren elektroenergetske mreže čuva interno u svakom mikroservisu.

Pravci daljih istraživanja trebalo bi da obuhvate implementaciju i drugih elektroenergetskih proračuna i da se na taj način utvrdi ponašanje i performanse sistema sa predloženom arhitekturom. Dodatno, primenu predložene arhitekture treba istestirati i postaviti na druge mikroservisne platforme kao što su: Amazon, Google, IBM i drugi. Dobijene rezultate uporediti sa vremenima izvršavanja na platformi *Service Fabric* i sa vremenima izvršavanja monolitne aplikacije, kao i analizirati koja su rešenja bolja. Sa druge strane, jedna takva predložena arhitektura u mikroservisnom *cloud* okruženju sa akademske strane zahteva istraživanja i sa bezbednosnih aspekata.

LITERATURA

[1] M. Cancila, D. Toombs, A. D. Waite, and K. Elias, Gartner: 2017 Planning Guide for Cloud Computing, 2016.

[2] N. Alshuqayran, N. Ali, R. Evans, A Systematic Mapping Study in Microservice Architecture, IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA), Macau, China, 2016.

[3] S. Stoja, S. Vukmirovic, N. Dalcekovic, D. Capko, B. Jelacic, Accelerating performance in critical topology analysis of distribution management system process by switching from monolithic to microservice, Revue Roumaine des Sciences Techniques. Ser. Electrotechnique et Energetique, vol. 63, no 3, pp. 338-343, 2018.

[4] S. Stoja, Distribuirana topološke analize modela podataka elektroenergetske mreže u mikroservisnom okruženju, Specijalistički rad, Fakultet tehničkih nauka, 2015.

[5] J. Stubbs, W. Moreira, R. Dooley, Distributed Systems of Microservices Using Docker and Serfnode, International Workshop on Science Gateways (IWSG), Budapest, Hungary, 2015.

[6] C. Anderson, Docker [Software engineering], IEEE Software, vol. 32, no. 3, pp. 102-c3, 2015.

[7] D. Jaramillo, D. V Nguyen, R. Smart, Leveraging microservices architecture by using Docker technology, SoutheastCon, Norfolk, USA, 2016.

[8] X. Zheng, J. Jiang, Y. Zhang, Y. Deng, M. Fu, T. Zheng, X. Liu, SmartVM: A Multi-Layer Microservice-Based Platform for Deploying SaaS, IEEE International Symposium

on Parallel and Distributed Processing with Applications and EEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC), 2017.

[9] M. Villamizar, O. Garces, H. Castro, M. Verano, L. Salamanca, R. Casallas, S. Gil, Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud, in: Com-puting Colombian Conference (10CCC), 2015 10th, 2015, pp. 583–590. doi:10.1109/ColumbianCC.2015.7333476.

[10] M. Villamizar, L. Ochoa, H. Castro, M. Verano Merino, R. Casallas, C. Valencia, M. Lang, L. Salamanca, S. Gil, A. Zambrano, O. Garcés, Infrastructure Cost Comparison of Running Web Applications in the Cloud Using AWS Lambda and Monolithic and Microservice Architectures, 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), Bogotá, Colombia, pp. 179-182, 2016.

[11] G. Kecskemeti, A. Cs. Marosi, A. Kertesz, The ENTICE approach to decompose monolithic services into microservices, International Conference on High Performance Computing & Simulation (HPCS), Innsbruck, Austria, 2016.

[12] S. Sharma, N. Uniyal, B. Tola, Y. Jiang, On Monolithic and Microservice Deployment of Network Functions, IEEE Conference on Network Softwarization (NetSoft), Paris, France, pp. 387-395, 2019.

[13] M. Rahman, J. Gao, A Reusable Automated Acceptance Testing Architecture for Microservices in Behavior-Driven Development, IEEE Symposium on Service-Oriented System Engineering (SOSE), San Francisco Bay, USA, 2015.

[14] A. Balalaie, A. Heydarnoori, P. Jamshidi, Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture, IEE Software, vol. 33, no. 3, pp. 42-52, 2016.

[15] E. Braun, T. Schlachter, C. Düpmeier, K. Stucky, W. Suess, A Generic Microservice Architecture for Environmental Data Management, Environmental Software Systems. Computer Science for Environmental Protection pp 383-394, 2017.

[16] N. H. Do, T. Van Do, X. Thi Tran, L. Farkas, Cs. Rotter, A scalable routing mechanism for stateful microservices, 20th Conference on Innovations in Clouds, Internet and Networks (ICIN), Paris France, 2017.

[17] S. Newman, Building Microservices Designing Fine-Grained Systems, O'Reilly, 2015.

[18] Microservices, [Online], Available: http://martinfowler.com/articles/microservices.html, [Accessed: 30-Oct-20].

[19] N. Dalcekovic, S. Vukmirovic, S. Stoja, N. Milosevic, Enabling the IoT paradigm through multi-tenancy supported by scalable data acquisition layer, Ann. des Telecommun. Telecommun., vol. 72, no. 1–2, pp. 71–78, 2016.

[20] N. Dalčeković, Platforma za transformaciju softverskih rešenja pametnih elektroenergetskih mreža na cloud bazirani višeorganizacijski SaaS, doktorska disertacija, Univerzitet u Novom Sadu, 2019.

[21] G. Buchgeher, M. Winterer, R. Weinreich, J. Luger, R. Wingelhofer, and M. Aistleitner: Microservices in a Small Development Organization - An Industrial Experience Report, 11th European Conference on Software Architecture (ECSA 2017), Canterbury, UK, September 11-17, 2017, doi: 10.1007/978-3-319-65831-5_15.

[22] G. Kecskemeti, A. Kertesz, A. C. Marosi, Towards a Methodology to Form Microservices from Monolithic Ones, Euro-Par 2016: Parallel Processing Workshops pp 284-295, 2016.

[23] G. Mazlami, J. Cito, Ph. Leitner, Extraction of Microservices from Monolithic Software Architectures, IEEE International Conference on Web Services (ICWS), Honolulu, HI, USA, 2017.

[24] V. Singh, S. K Peddoju, Container-based Microservice Architecture for Cloud Applications, International Conference on Computing, Communication and Automation, 2017.

[25] Y. Yu, H. Silveira, M. Sundaram, A Microservice Based Reference Architecture Model in the Context of Enterprise Architecture, IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), China, 2016.

[26] J. C Olivares-Rojas, E. Reyes-Archundia, J.A. Gutiérrez-Gnecchi, J.W. González-Murueta, J. Cerda-Jacobo, A multi-tier architecture for data analytics in smart metering systems, Simulation Modelling Practice and Theory, vol. 122, 2020.

[27] M. Forcan, M. Maksimović, Cloud-fog-based approach for smart grid monitoring, Simulation Modelling Practice and Theory, vol. 101, 2019.

[28] Lyu, H. Wei, X. Bai, C. Lian, Microservice-Based Architecture for an Energy Management System, IEEE Systems Journal, vol.14, no.4, pp. 5061 - 5072, 2020.

[29] B. Jelacic, I. Lendak, S. Stoja, M. Stanojevic, D. Rosic, Security Risk Assessment-based Cloud Migration Methodology for Smart Grid OT Services, Acta Polytechnica Hungarica, vol. 17, no. 5, pp. 113-134, 2020.

[30] G.T. Heydt, The Next Generation of Power Distribution Systems, IEEE Trans. Smart Grid, vol.1, no.3, pp. 225-235, 2010.

[31] V. Krsman, Specijalizovani algoritmi za detekciju, identifikaciju i estimaciju loših podataka u elektrodistributivnim mrežama, doktorska disertacija, Univerzitet u Novom Sadu, 2017.

[32] D.S. Popović, Power applications: A cherry on the top of the DMS cake, DA/DSM DistribuTECH Europe, Specialist Track 3, Session 3, Paper 2, Vienna, Austria, 2000.

[33] W.R. Cassel, Distribution Management Systems: Functions and Payback, IEEE Transactions on Power Systems, vol. 8, no. 3, pp. 796-801, 1993.

[34] V. C. Strezoski, Osnovi elektroenergetike, FTN Izdavaštvo, Novi Sad, 2014.

[35] M. Ćalović, A. Sarić, Osnovi analize elektroenergetskih mreža i sistema, Akademska Misao, Beograd, 2004.

[36] L. Strezoski, Proračun kompleksnih kratkih spojeva neuravnoteženih distributivnih mreža sa distribuiranim energetskim resursima, doktorska disertacija, Univerzitet u Novom Sadu, 2017.

[37] V. Strezoski, D. Popović, D. Bekut, N. Katić, G. Švenda, Z. Gorečan, J. Dujić, Osnovne energetske funkcije za analizu, upravljanje i planiranje pogona srednjenaponskih distributivnih mreža, IV skup Trendovi Razvoja: „Nove tehnologije u elektrodistribuciji“, Kopaonik, 1998.

[38] D. Čapko, Optimalna podela velikih modela podataka u okviru nadzorno-upravljačkih elektroenergetskih sistema, doktorska disertacija, Univerzitet u Novom Sadu, 2012.

[39] I. Dzafic, S. Henselmeyer, N. Lecek, T. Schwietzke, D. Ablakovic, Object Oriented Topology Tracing for Large Scale Three Phase Distribution Networks, 3rd IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe), Berlin, 2012.

[40] S. Pandit, S.A. Soman, S.A. Khaparde, Object-oriented network topology processor [power system automation], IEEE Computer applications in Power, vol. 14, no. 2, pp. 42-46, 2001.

[41] T. Kovac, D. Bekut, A. Saric, Topological analysis of unbalanced distribution networks with single-phase switching equipment and temporary elements, International Transaction on Electrical Energy Systems, vol. 27, no. 12, e2455, 2017.

[42] B. Jelacic, S. Vukmirovic, S. Stoja, A software infrastructure for distributed data models in Cloud, 21st Telecommunications Forum, Belgrade, Serbia, 2013.

[43] I. V. Nemoianu, R. M. Ciuceanu, Characterization of non-linear threephase unbalanced circuits power flow supplied with symmetrical voltages, Rev. Roum. Sci. Techn. – Électrotechn. et Énerg., vol. 60, no. 4, pp. 355–365, 2015.

[44] P. Vidović, A. Sarić, A novel correlated intervals-based algorithm for distribution power flow calculation, International Journal of Electrical Power & Energy Systems, no. 90, pp. 245-255, 2017.

[45] V. C. Strezoski, P. Vidović, Power flow for general mixed distribution networks, International Transactions on Electrical Energy Systems, vol. 25, no. 10, pp. 2455–2471, 2014.

[46] D. Popović, D. Bekut, V. Treskanica, Specijalizovani DMS algoritmi; DMS Grupa, Novi Sad, 2004.

[47] V. C. Strezoski, P. Vidović, Power flow for general mixed distribution networks International Transactions on Electrical Energy Systemsvol. 25, pp. 2455–2471, 2014.

[48] G. Švenda, V. Strezoski, S. Kanjuh, Real-life distribution state estimation integrated in the distribution management system, International Transactions on Electrical Energy Systems, vol. 27, no.5, pp.1-16, 2016.

[49] D. Axel, W. Salter, What is SCADA, International Conference on Accelerator and Large Experimental Physics Control Systems, Trieste, Italy, 1999.

[50] A. Zidan, M. Khairalla, A. M. Abdrabou, T. Khalifa, K. Shaban, A. Abdrabou, R. El Shatshat, A. M. Gaouda, Fault Detection, Isolation, and Service Restoration in Distribution Systems: State-of-the-Art and Future Trends, IEEE Transactions on Smart Grid, vol. 8, no. 5, pp. 2170 – 2185, 2017.

[51] C. H. Lin; H. J. Chuang; C. S. Chen; C. S. Li; C. Y. Ho, Fault detection, isolation and restoration using a multiagent-based Distribution Automation System, 4th IEEE Conference on Industrial Electronics and Applications, 2009.

[52] I. Roytelman, V. Landenberger, Real-time Distribution System Analysis - Integral Part of DMS, IEEE/PES Power Systems Conference and Exposition, USA, 2009.

[53] N. Kovački, Operativno planiranje rekonfiguracije distributivnih mreža primenom višekriterijumske optimizacije, doktorska disertacija, Univerzitet u Novom Sadu, 2017.

[54] J. Gama, P. Pereira Rodrigues, Stream-Based Electricity Load Forecast, Knowledge Discovery in Databases: PKDD 2007, pp 446-453, Lecture Notes in Computer Science, vol 4702. Springer, Berlin, Heidelberg.

[55] S. R. Khuntia, J. L. Rueda, M. A. M. M. van der Meijden, Forecasting the load of electrical power systems in mid- and long-term horizons: a review, IET Generation, Transmission & Distribution, vol. 10, no. 16, pp. 3971 – 3977, 2016.

[56] T. Honga, S. Fan, Probabilistic electric load forecasting: A tutorial review, International Journal of Forecasting, vol. 32, no. 3, pp. 914-938, 2016.

[57] J. W. Taylor, An evaluation of methods for very short-term load forecasting using minute-by-minute British data, International Journal of Forecasting, vol. 24, no. 4, pp. 645-658, 2008.

[58] K. Liu, S. Subbarayan, R.R. Shoults, M.T. Manry, C. Kwan, F.I. Lewis, J. Naccarino, Comparison of very short-term load forecasting techniques, IEEE Transactions on Power Systems, vol. 11, no. 2, pp. 877 - 882, 1996.

[59] G. Gross, F.D. Galiana, Short-term load forecasting, Proceedings of the IEEE, vol. 75, no.12, pp. 1558-1572, 1987.

[60] J. Y. Fan, J. D. McDonald, A real-time implementation of short-term load forecasting for distribution power systems, IEEE Transactions on Power Systems, vol. 9, no. 2, pp. 988-994, 1994.

[61] N. Amjady, F. Keynia, Mid-term load forecasting of power systems by a new prediction method, Energy Conversion and Management, vol. 49, no. 10, pp. 2678-2687, 2008.

[62] S. R. Khuntia, J. L. Rueda, M. A. M. M. van der Meijden, Forecasting the load of electrical power systems in mid- and long-term horizons: a review, IET Generation, Transmission & Distribution, vol. 10, no. 16, pp. 3971 – 3977, 2016.

[63] H. Daneshi; M. Shahidehpour; A. L. Choobbar, Long-term load forecasting in electricity market, IEEE International Conference on Electro/Information Technology, USA, 2008.

[64] M.S. Kandil; S.M. El-Debeiky; N.E. Hasanien, Long-term load forecasting for fast developing utility using a knowledge-based expert system, IEEE Transactions on Power Systems, vol. 17, no. 2, pp. 491 – 496, 2002.

[65] G. Cochenour, R. Ochoa, V. Rajsekar, Distribution network model readiness for Advanced Distribution Management Systems, IEEE PES T&D Conference and Exposition, Chicago, USA, 2014.

[66] J. Carden, D. Popović, Closed-Loop Volt\/Var Optimization: Addressing Peak Load Reduction, IEEE Power and Energy Magazine, vol. 16, no. 2, pp. 67-78, 2018.

[67] V. Strezoski, D. Popović, D. Bekut, G. Švenda, DMS - Basis for increasing of green distributed generation penetration in distributed networks, Thermal Science, vol. 16, no. 1, pp. S189-S203, 2012.

[68] J. Thönes, Microservices, IEEE Software, vol. 32, no. 1, pp. 116-116, 2015.

[69] M. Kalske, N. M¨akitalo, T. Mikkonen, Challenges When Moving from Monolith to Microservice Architecture, Current Trends in Web Engineering, Springer, pp.32-47, 2018.

[70] P. D. Francesco, I. Malavolta, and P. Lago, Research on Architecting Microservices: Trends, Focus, and Potential for Industrial Adoption, IEEE International Conference on Software Architecture. IEEE, pp. 21–30, Apr. 2017

[71] M. Mazzara, K. Khanda, R. Mustafin, V. Rivera, L. Safina, A. Sillitti, Microservices Science and Engineering, Proceedings of 5th International Conference in Software Engineering for Defence Applications, pp 11-20, 2016.

[72] O. Zimmermann, Microservices tenets, Computer Science - Research and Development, Volume 32, no. 3–4, pp 301–310, 2017.

[73] S. Newman, Monolith to Microservices: Evolutionary Patterns to Transform Your Monolith, O'Reilly, 2019.

[74] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, P. Merson, R. Nord, J. Stafford, Documenting Software Architectures: Views and Beyond (2nd Edition), Boston: Addison-Wesley, 2010.

[75] Microservice Architecture Patterns and Best Practices, [Online], Available: http://microservices.io/ , [Accessed: 2019-25-11].

[76] M. Richards, Software Architecture Patterns, O'Reilly, 2015.

[77] B. Mayer, R. Weinreich, An Approach to Extract the Architecture of Microservice-Based Software Systems, IEEE Symposium on Service-Oriented System Engineering (SOSE), Germany, 2018.

[78] S. Stoja, S. Vukmirovic, B. Jelacic, Publisher/Subscriber Implementation in Cloud Environment, P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2013 Eighth International Conference, Compiegne, France, 2013.

[79] S. Stoja, Implementacija *Publisher/Subscriber* komunikacije u *cloud* okruženju, Zb. Rad. Fak. Teh. Nauk., vol. 12, pp. 2486-2489, 2012.

[80] S. Stoja, S. Vukmirovic, B. Jelacic, D. Capko, N. Dalcekovic, Architecture of Real-Time Database in Cloud Environment for Distributed Systems, Artificial Intelligence, Modelling and Simulation (AIMS), 2014 2nd International Conference, Madrid, Spain, 2014.

[81] A. Brogi, A. Canciani, D. Neri, L. Rinaldi, J. Soldani, Towards a Reference Dataset of Microservice-Based Applications, Software Engineering and Formal Methods, pp.219-229, 2018.

[82] D. Trihinas, A. Tryfonos, M. D. Dikaiakos, G. Pallis, DevOps as a Service: Pushing the Boundaries of Microservice Adoption, IEEE Internet Computing, vol. 22, no. 3, pp. 65 – 71, 2018.

[83] B. Simon, C. Millard, I. Walden. Contracts for clouds: Comparison and analysis of the terms and conditions of cloud computing services, International Journal of Law and Information Technology vol. 19 no.3, pp 187-223, 2011

[84] S. Nagaprasad, A. VinayaBabu, K. Madhukar, D. Marlene G Verghese, V. Mallaiah and A. Sreelatha, Reviewing some platforms in cloud computing, International Journal of Engineering and Technology, vol. 2, no.5, pp. 348-353, 2010.

[85] Distributed Internet Computing for IT and Scientific Research, Dikaiakos, M.D. Univer. Of Cyprus, 2009.

[86] B. James, The enterprise cloud, O'Reilly, 2015.

[87] B. Briggs, E. Kassner, Enterprise cloud strategy, Redmont: Microsoft Press, 2016.

[88] W. Y. Chang, H. Abu-Amara, J. F. Sanford, Transforming enterprise cloud services, Springer, 2010.

[89] H. Bai, Programming Microsoft Azure Service fabric, Microsoft Press, 2016.

[90] Microsoft Azure Service Fabric, [Online], Available: https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-overview , [Accessed: 30-Oct-20].

[91] Introduction to Service Fabric Reliable Actors, [Online], Available: https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-reliable-actors-introduction, [Accessed: 5-Apr-21].

[92] Introduction to Reliable Collections in Azure Service Fabric stateful services, [Online], Available: https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-reliable-services-reliable-collections [Accessed: 5-Apr-21].

[93] J. Murty, Programming Amazon Web Services: S3, EC2, SQS, FPS, and SimpleDB, O'Reilly, 2008.

[94] AWS Lambda, [Online], Available: https://aws.amazon.com/lambda/, [Accessed: 30-Oct-20].

[95] AWS Fargate, [Online], Available: https://aws.amazon.com/fargate/, [Accessed: 30-Oct-20].

[96] D. Vohra, Amazon Fargate Quick Start Guide, Packt, 2018.

[97] R. C. Dugan et al., Distribution System Analysis to Support the Smart Grid, IEEE PES Gen. Meet., pp. 1–8, 2010.

[98] M. Kezunovic, J. D. McCalley, T. J. Overbye, Smart Grids and Beyond: Achieving the Full Potential of Electricity Systems, Proceedings of the IEEE, vol. 100, Special Centennial Issue, pp. 1329–1341, 2012.

[99] S. Vukmirovic, A. Erdeljan, I. Lendak, D. Capko, Extension of the Common Information Model with Virtual Meter, Electronics and Electrical Engineering, Kaunas: Technologija, vol. 107, no. 1, pp. 59-64, 2011.

[100] D. S. Popovic, E. Varga, Z. Perlic, Extension of the Common Information Model with a Catalog of Topologies, IEEE Transactions on Power Systems, vol. 22, no. 2, pp. 770-777, 2007.

[101] S. Vukmirovic, A. Erdeljan, I. Lendak, D. Capko, Unifying the common information model (CIM), Rev. Roum. Sci. Techn. – Électrotechn. et Énerg., vol. 57, no. 3, pp. 301–310, 2012.

[102] IEC/TR 61968-11 Application integration at electric utilities - System interfaces for distribution management - Part 11: Common information model (CIM) extensions for distribution, IEC, Edition 1.0, 2010.

[103] E. Price, G. Aroraa, Hands-On Microservices with C# 8 and .NET Core 3: Refactor your monolith architecture into microservices using Azure, 3rd Edition, Packt, 2020.

[104] D. Čapko, A. Erdeljan, S. Vukmirović, I. Lendak, A Hybrid Genetic Algorithm for Partitioning of Data Model in Distribution Management Systems, Information technology and control, vol. 40, no. 4, 2011.

[105] D. Čapko, A. Erdeljan, M. Popović, G. Švenda, An Optimal Initial Partitioning of Large Data Model in Utility Management Systems, Advances in Electrical and Computer Engineering, vol. 11, no. 4, 2011.

[106] M. Korica, Particionisanje modela podataka distributivne mreže u Cloud okruženju, Zb. Rad. Fak. Teh. Nauk., vol. 14, 2015.

[107] V. Strezoski, Analiza elektroenergetskih sistema, skripta, Fakultet tehničkih nauka, Novi Sad, 2010.

[108] Dv3 and Dsv3-series, [Online], Available: https://docs.microsoft.com/en-us/azure/virtual-machines/dv3-dsv3-series, [Accessed: 11-May-21].

BIOGRAFIJA

Kandidat Sebastijan Stoja je rođen 1988. godine u Kikindi. Završio je Srednju Elektrotehničku i Građevinsku školu "Nikola Tesla", smer Elektrotehničar računara u Zrenjaninu, 2007. godine. Fakultet Tehničkih Nauka u Novom Sadu je upisao 2007. godine da bi diplomski rad pod nazivom "Softverski alat za administraciju DMS *SmartCache* baze podataka" odbranio 2012. godine na departmanu za Računarstvo i automatiku, Fakulteta tehničkih nauka u Novom Sadu, sa prosečnom ocenom 9.14. Iste godine je upisao master studije na Fakultetu Tehničkih Nauka u Novom Sadu. Diplomski-master rad pod nazivom "Implementacija *Publisher/Subscriber* komunikacije u *cloud* okruženju" odbranio je 2012. godine na departmanu za Računarstvo i automatiku, Fakulteta tehničkih nauka u Novom Sadu sa prosečnom ocenom 9.9. Iste godine je upisao specijalističke akademske studije. Specijalistički rad pod nazivom "Distribuirana topološke analize modela podataka elektroenergetske mreže u mikroservisnom okruženju" odbranio je 2015. Godine na departmanu za Energetika, elektronika i telekomunikacije sa prosečnom ocenom 10.0.

Od 2012. godine je zaposlen u kompaniji "Schneider Electric DMS NS", dok je u istoj kompaniji imao angažman kao stipendista dve godine. Počeo je kao razvojni inženjer, potom kao vlasnik proivoda za SaaS, zatim vlasnik proizvoda za sisteme za upravljanje radnim nalozima strujne industrije da bi 2019. godine počeo da radi kao menadžer proizvoda za pomenute sisteme. 2020. godine prelazi kao menadžer proizvoda za proizvoda gasne industrije.

Angažovan je i na Fakultetu tehničkih nauka. U zvanje asistenta-mastera je izabran 2016. godine. Trenutno drži vežbe iz dva predmeta "Cloud Computing u elektroenergetskim sistemima" i "Cloud računarstvo u infrastrukturnim sistemima" sa nepunim radnim vremenom. Godine 2015. upisao je doktorske studije na studijskom programu Energetika, elektronika i telekomunikacije, na Fakultetu tehničkih nauka u Novom Sadu. Položio je sve ispite predviđene programom sa prosečnom ocenom 10.00. Koautor je tri naučna rada objavljena u međunarodnim časopisima, jedan u nacionalnom časopisu kao i na sedam rada objavljenih u saopštenjima sa međunarodnih skupova.

Oženjen je, ima dvoje dece i živi u Novom Sadu. Od stranih jezika govori rumunski, engleski i španski.

Овај Образац чини саставни део докторске дисертације, односно докторског уметничког пројекта који се брани на Универзитету у Новом Саду. Попуњен Образац укоричити иза текста докторске дисертације, односно докторског уметничког пројекта.

План третмана података

Назив пројекта/истраживања
Архитектура софтверског система за електроенергетске прорачуне заснована на микросервисима
Назив институције/институција у оквиру којих се спроводи истраживање
а) **Универзитет у Новом Саду, Факултет Техничких Наука** б) в)
Назив програма у оквиру ког се реализује истраживање
Истраживање се реализује у оквиру израде докторске дисертације на студијском програму Енергетика, Електроника и телекомуникације.
1. Опис података
1.1 Врста студије *Укратко описати тип студије у оквиру које се подаци прикупљају* **Докторска дисертација** _____ 1.2 Врсте података а) **квантитативни** б) **квалитативни** 1.3. Начин прикупљања података а) анкете, упитници, тестови

б) клиничке процене, медицински записи, електронски здравствени записи

в) генотипови: навести врсту _____

г) административни подаци: навести врсту _____

д) узорци ткива: навести врсту_____

ђ) снимци, фотографије: навести врсту_____

е) текст, навести врсту _____

ж) мапа, навести врсту _____

з) остало: описати **рачунарски експерименти**

1.3 Формат података, употребљене скале, количина података

1.3.1 Употребљени софтвер и формат датотеке:

а) Excel фајл, датотека _____

b) SPSS фајл, датотека _____

c) PDF фајл, датотека _____

d) Текст фајл, датотека _____

e) JPG фајл, датотека _____

f) Остало, датотека **.xml**

1.3.2. Број записа (код квантитативних података)

а) број варијабли **велики број**

б) број мерења (испитаника, процена, снимака и сл.) **велики број**

1.3.3. Поновљена мерења

а) **да**

б) не

Уколико је одговор да, одговорити на следећа питања:

а) временски размак између поновљених мера је **променљив**

б) варијабле које се више пута мере односе се на **време извршавања**

в) нове верзије фајлова који садрже поновљена мерења су именоване као _____

Напомене: _____

Да ли формати и софтвер омогућавају дељење и дугорочну валидност података?

а) ***Да***

б) Не

Ако је одговор не, образложити _____

_____

2. Прикупљање података

2.1 Методологија за прикупљање/генерисање података

2.1.1. У оквиру ког истраживачког нацрта су подаци прикупљени?

а) експеримент, навести тип **рачунарски експеримент**

б) корелационо истраживање, навести тип _____

ц) анализа текста, навести тип **Анализа доступне литературе**

д) остало, навести шта _____

2.1.2 Навести врсте мерних инструмената или стандарде података специфичних за одређену научну дисциплину (ако постоје).

_____

_____

2.2 Квалитет података и стандарди

2.2.1. Третман недостајућих података

а) Да ли матрица садржи недостајуће податке? Да **Не**

Ако је одговор да, одговорити на следећа питања:

а) Колики је број недостајућих података? _____

б) Да ли се кориснику матрице препоручује замена недостајућих података? Да Не

в) Ако је одговор да, навести сугестије за третман замене недостајућих података

_____

2.2.2. На који начин је контролисан квалитет података? Описати

Квалитет података је контролисан поређењем експерименталних и теоријских података

Квалитет података је контролисан поређењем експерименталних и теоријских података

2.2.3. На који начин је извршена контрола уноса података у матрицу?

Контрола уноса података је изведена на бази експертног знања

_____

3. Третман података и пратећа документација

3.1. Третман и чување података

3.1.1. Подаци ће бити депоновани у **Универзитет у Новом Саду** *репозиторијум.*

3.1.2. URL адреса https://www.cris.uns.ac.rs/searchDissertations.jsf

3.1.3. DOI _____

3.1.4. Да ли ће подаци бити у отвореном приступу?

а) ***Да***

б) *Да, али после ембарга који ће трајати до _____*

в) *Не*

Ако је одговор не, навести разлог _____

3.1.5. Подаци неће бити депоновани у репозиторијум, али ће бити чувани.

Образложење

_____

_____

3.2 Метаподаци и документација података

3.2.1. Који стандард за метаподатке ће бити примењен? **Стандард који примењује Репозиторијум докторских дисертација Универзитета у Новом Саду**

3.2.1. Навести метаподатке на основу којих су подаци депоновани у репозиторијум.

_____

_____

Ако је потребно, навести методе које се користе за преузимање података, аналитичке и процедуралне информације, њихово кодирање, детаљне описе варијабли, записа итд.

_____

_____

_____

_____

3.3 Стратегија и стандарди за чување података

3.3.1. До ког периода ће подаци бити чувани у репозиторијуму? _____

3.3.2. Да ли ће подаци бити депоновани под шифром? **Да** Не

3.3.3. Да ли ће шифра бити доступна одређеном кругу истраживача? **Да** Не

3.3.4. Да ли се подаци морају уклонити из отвореног приступа после извесног времена?

Да **Не**

Образложити

_____

_____

4. Безбедност података и заштита поверљивих информација

Овај одељак МОРА бити попуњен ако ваши подаци укључују личне податке који се односе на учеснике у истраживању. За друга истраживања треба такође размотрити заштиту и сигурност података.

4.1 Формални стандарди за сигурност информација/података

Истраживачи који спроводе испитивања с људима морају да се придржавају Закона о заштити података о личности (*https://www.paragraf.rs/propisi/zakon_o_zastiti_podataka_o_licnosti.html*) и одговарајућег институционалног кодекса о академском интегритету.

4.1.2. Да ли је истраживање одобрено од стране етичке комисије? Да **Не**

Ако је одговор Да, навести датум и назив етичке комисије која је одобрила истраживање

_____

4.1.2. Да ли подаци укључују личне податке учесника у истраживању? Да **Не**

Ако је одговор да, наведите на који начин сте осигурали поверљивост и сигурност информација везаних за испитанике:

а) Подаци нису у отвореном приступу

б) Подаци су анонимизирани

ц) Остало, навести шта

_____

_____

5. Доступност података

5.1. Подаци ће бити

а) ***јавно доступни***

б) доступни само уском кругу истраживача у одређеној научној области

ц) затворени

Ако су подаци доступни само уском кругу истраживача, навести под којим условима могу да их користе:

_____

_____

Ако су подаци доступни само уском кругу истраживача, навести на који начин могу приступити подацима: _____

_____

5.4. Навести лиценцу под којом ће прикупљени подаци бити архивирани.

ауторство - некомерцијално

6. Улоге и одговорност

6.1. Навести име и презиме и мејл адресу власника (аутора) података

Себастијан Стоја, sebastijan.stoja@uns.ac.rs

6.2. Навести име и презиме и мејл адресу особе која одржава матрицу с подацима

Себастијан Стоја, sebastijan.stoja@uns.ac.rs

6.3. Навести име и презиме и мејл адресу особе која омогућује приступ подацима другим истраживачима

Себастијан Стоја, sebastijan.stoja@uns.ac.rs