Mobilna Android aplikacija za analizu energetske učinkovitosti računalnih sustava visokih performansi

Oršolić, Josip

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

MOBILNA ANDROID APLIKACIJA ZA ANALIZU ENERGETSKE UČINKOVITOSTI RAČUNALNIH SUSTAVA VISOKIH PERFORMANSI

Završni rad

Josip Oršolić

Osijek, 2021.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 20.09.2022.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime Pristupnika:	Josip Oršolić
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. Pristupnika, godina	R4254, 26.07.2018.
OIB Pristupnika:	72124550500
Mentor:	prof. dr. sc. Goran Martinović
Sumentor:	,
Sumentor iz tvrtke:	
Naslov završnog rada:	Mobilna Android aplikacija za analizu energetske učinkovitosti računalnih sustava visokih performansi
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak završnog rad:	U završnom radu potrebno je opisati i analizirati izazove računarstva visokih performansi s naglaskom na energetski osviješten načina rada, odnosno načela zelenog računarstva. Nadalje, treba istražiti i objasniti slojeve i pristupe za ostvarivanje energetski učinkovitog sustava, kao i mjerila vrjednovanja i metrike za izračun učinkovitosti računalnih sustava visokih performansi. S ciljem omogućavanja uvoza
Prijedlog ocjene završnog rada:	Vrlo dobar (4)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 2 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 1 bod/boda Jasnoća pismenog izražavanja: 1 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	20.09.2022.
Datum potvrde ocjene od strane Odbora:	21.09.2022.

Potvrda mentora o predaji konačne verzije rada:	*Mentor elektronički potpisao predaju konačne verzije.*
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 28.08.2023.

Ime i prezime studenta:	Josip Oršolić
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4254, 26.07.2018.
Turnitin podudaranje [%]:	5

Ovom izjavom izjavljujem da je rad pod nazivom**: Mobilna Android aplikacija za analizu energetske učinkovitosti računalnih sustava visokih performansi**

izrađen pod vodstvom mentora prof. dr. sc. Goran Martinović

i sumentora ,

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

Cilj ovog završnog rada je izrada mobilne aplikacije za analizu energetske učinkovitosti računalnih sustava visokih performansi . U aplikaciju se mogu unijeti parametri koji se tiču potrošnje energije i potreba računalnih sustava visokih performansi. Na temelju tih parametara, aplikacija treba izračunati metrike koje opisuju koliko točno je neki računalni sustav visokih performansi učinkovit. S tom namjerom u radu su opisane metrike koje određuju energetsku učinkovitost pojedinog računalnog sustava, ali i razloge zašto se koriste upravo te metrike i njihovi izračuni. Također, opisano je zašto je ušteda energije u podatkovnim središtima trend, te zašto će energetska učinkovitost postati još važnija u vremenima skupe energije. Mobilna Android aplikacije razvijena je u okolini Visual Studio Code, a korištene je programski jezik Dart u sklopu razvojnog okvira Flutter.

U drugom poglavlju završnog rada detaljno se govori o upotrebi računalnih sustava visokih performansi, načelima zelenog računarstva te potrošnji energije u podatkovnim središtima kao i o načinima za smanjenje potrošnje. U trećem poglavlju detaljno se opisuju metrike koje će biti korištene pri izračunima kao i dijagram toka rada mobilne aplikacije. Četvrto poglavlje bavi se opisivanjem programskih alata i tehnologija koje su se koristile prilikom izrade praktičnog dijela aplikacije, ali i opisivanjem programskog koda mobilne aplikacije. U petom poglavlju se prikazuje način korištenja mobilne aplikacije i rezultati ispitivanja za odgovarajuće ulazne podatke.

1.1. Zadatak završnog rada

U završnom radu potrebno je opisati i analizirati izazove računarstva visokih performansi s naglaskom na energetski osviješten načina rada, odnosno načela zelenog računarstva. Nadalje, treba istražiti i objasniti slojeve i pristupe za ostvarivanje energetski učinkovitog sustava, kao i mjerila vrjednovanja i metrike za izračun učinkovitosti računalnih sustava visokih performansi. S ciljem omogućavanja unosa ulaznih parametara, njihove pohrane u bazu podataka, procjene razine primijenjenih načela zelenog računarstva, izračuna učinkovitosti računalnih sustava visokih performansi i prikaza rezultata analize, potrebno je razraditi model i programsku arhitekturu mobilne aplikacije s bazom podataka. Pri izračunu učinkovitosti treba koristiti metrike PUE, DCiE, TGI, DWPE, a po potrebi i druge. Mobilnu Android aplikaciju za analizu energetske učinkovitosti računalnih sustava visokih performansi treba programski ostvariti u odgovarajućoj razvojnoj okolini korištenjem prikladnih programskih jezika i tehnologija, te je ispitati i analizirati za različite računalne sustave visokih performansi.

2. IZAZOVI ENERGETSKE UČINKOVITOSTI TE PROBLEM POTROŠNJE ENERGIJE

Kada se govori o računarstvu visokih performansi (engl. HPC – High Performance Computing) postoji više objašnjenja koja samo drugačije opisuju ono što rade takvi računalni sustavi. Jedna od definicija govori kako se računarstvo visokih performansi najčešće odnosi na nakupljanje računalne snage na način da se postignu mnogo veće performanse, nego što bi to slučaj bio kod običnih stolnih računala [1]. Druga definicija kaže da je računarstvo visokih performansi korištenje superračunala i tehnika paralelne obradbe za rješavanje složenih računalnih problema [2].

2.1. Mjesta upotrebe računarstva visokih performansi

Kao što je spomenuto, računarstvo visokih performansi se koristi kako bi se postigla veća računalna moć kojom bi se lakše mogle izračunati složene računske operacije koje , na primjer, srećemo prilikom nekog istraživanja gdje bi trebalo proširiti područje proučavanja (pokrajina → država→ regija→ globalna razina), kada bi trebalo unijeti dodatne podatke ili kada bi trebalo povećati model simulacije. Navedeni slučajevi mogu postati (i najčešće jesu) prezahtjevni za obična računala zato što je:

- Vrijeme izvođenja predugo
- Procesor ima nedovoljno veliki kapacitet.

Premda se u javnosti toliko ne govori o računarstvu visokih performansi ono je nezaobilazno u gotovo svakoj grani znanosti te većim organizacijama. Koristi se u:

- Laboratorijima – koristi se kako bi znanstvenici mogli jednostavnije pronaći izvore obnovljive energije, predvidjeti nadolazeće vrijeme, stvoriti nove elemente, pronaći cjepiva ili lijekove za bolesti ili viruse, poboljšati načine za otkrivanje raka, itd.
- Filmskoj industriji – koristi se za postizanje specijalnih efekata
- Naftnoj industriji – koristi se kako bi se što točnije izabralo mjesta za nove bušotine nafte ili kako bi se poboljšala proizvodnja iz postojećih bušotina
- Računarstvu – koristi se za poučavanje samovoznih automobila
- Industriji – veliki računalni sustavi osiguravaju da u zalihama postoji dovoljno dijelova kako ne bi došlo do zaustavljanja proizvodnje [3]

2.2. Zeleno računarstvo

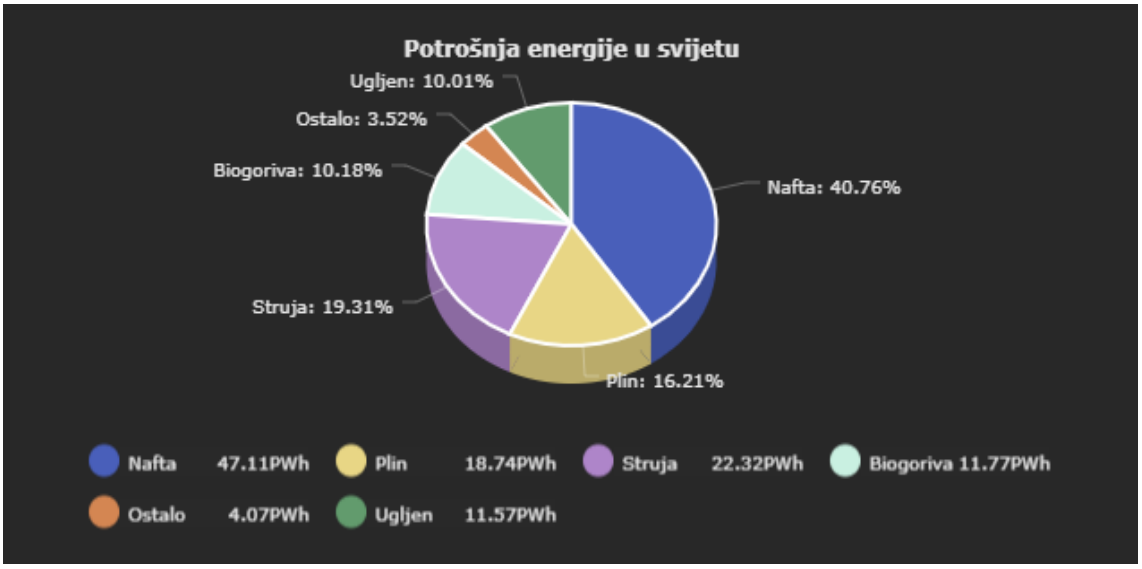
Zeleno računarstvo [4] se odnosi na korištenje računala te njihovih resursa na odgovoran način koji je pritom pogodan za okoliš. U širem smislu još se može definirati kao područje zaslužno za dizajn, izgradnju, proizvodnju te korištenje i odlaganje računala s ciljem smanjenja negativnog utjecaja na okoliš. Dakle, kao što je već navedeno, zeleno računarstvo može se podijeliti na četiri dijela, to jest, četiri principa koje je poželjno slijediti kako bi se postigla ekološki povoljna situacija:

1. Zelena upotreba - cilj je smanjenje potrošnje energije, ali se od računala zahtjeva da bude dugotrajan pri korištenju niske razine energije. Zbog toga su u današnja računala uvedena stanja poput „Stanja spavanja“ (engl. Sleep mode) te stanja u kojem se troši manje energije nego inače.
2. Zeleno odlaganje - odnosi se na ponovnu uporabu ili metode odlaganja računala. Pojedini dijelovi računala mogu se ponovno upotrijebiti čime se smanjuje elektronički otpad.
3. Zeleni dizajn - bavi se proizvodnjom računala ili računalnih sustava koja su i korisna i učinkovita, a pritom ne utječu negativno na okoliš.
4. Zelena proizvodnja - veliki nedostatak današnjih računala je taj što nisu lako razgradiva zbog čega se stvara enorman elektronički otpad. Treba ulagati sve više napora u proizvodnju računala koja će sadržavati biološke komponente i time otkloniti otrovne ili opasne supstance [4].

Nadalje, konačni ciljevi zelenog računarstva je da se postigne što bolja računalna učinkovitost uz smanjenje utjecaja na okoliš, smanjenje odlaganja elektroničkog otpada, smanjenje korištenja papira, optimizacija energije koja se pritom troši te korištenje energije stvorene preko obnovljivih izvora energije. Možda i ponajveći problem je odlaganje računalnog otpada. Prema [19], ukoliko bi se primijenila tehnika „Ponovno iskoristi, obnovi, recikliraj“ (engl. *Reuse, refurbish, recycle*) na način da se umjesto odbacivanja postojećih sustava odabere njihova zamjena s boljim i učinkovitijim dijelovima, a stari dijelovi mogli bi se iskoristiti u sustavima koji se mogu pokretati s istim tim dijelovima. Isto tako, prema [20], koristeći se načelom „upravljanja snagom“ na način da se računala namjeste da na način da se samostalne ugase ukoliko nakon određenog perioda nije bilo aktivnosti.

2.3. Potrošnja energije u svijetu

Kako bi se bolje i lakše razumio razlog zašto se područje računarstva sve više i više okreće ekološki prihvatljivim rješenjima, prvo se mora „baciti“ pogled na svjetsku potrošnju energije što se može učiniti pogledom na sliku 2.1. iz [5].



Sl. 2.1. *Svjetska potrošnja energije u 2018. godini prema [22]*

Nakon kratkog razmatranja grafa sa slike 2.1, može se lako zaključiti kako potrošnja struje u svijetu ima udio od čak skoro 20%. Ako se ništa ne bi napravilo povodom reduciranja elektroničkog otpada te smanjenja potrošnje energije, onda bi se mogao vidjeti velik porast emisija stakleničkih plinova. Moore-ov zakon nalaže kako se broj tranzistora na mikročipovima udvostručuje svake dvije godine te raste eksponencijalno [6]. Nadalje, povećava se i brzina razmjene podataka pa se tako proporcionalno povećava i potrošnja energije. U 2018. godini već je postojala vrlo raširena mreža elektronike i elektroničke i računalne opreme, a s obzirom da nas je svijet u protekle dvije godine bio zatečen pandemijom COVID-19 gdje je velika većina ljudi bila prisiljena ostati kod kuće, osjetio se i velik porast prodaje računala i popratne računalne opreme čijom prodajom se primijetila posljedica velikog povećanja potrošnje energije i ispuštanja CO_2.

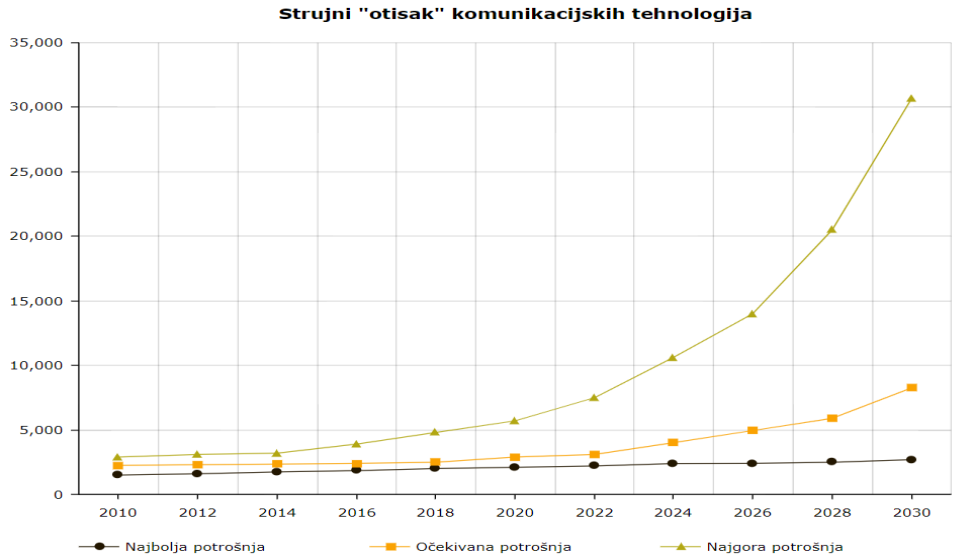
2.4. Podatkovna središta i potrošnja električne energije

Podatkovna središta (engl. data centers) sastoje se od jedne ili više zgrada ili posebnih mjesta u nekoj zgradi gdje se nalazi sva računalna oprema i gdje se izvršavaju sve operacije vezane za neku organizaciju. Potrebno je biti osviješten što znači da bi se trebalo paziti na potrošnju energije (slike 2.2 i 2.3. iz [7]) te ispuštanje CO_2 kao posljedicu korištenja sve tehnološke opreme. Danas se, nažalost,

ne može pronaći mnogo podataka o emisijama ugljičnog dioksida iz podatkovnih središta (iako prema [8] postoje neke procjene da su 2010. godine podatkovna središta trošila 1-1,5% svjetske potrošnje struje te ispuštala oko 2% ukupne emisije CO_2, dakle, ne može se točno odrediti koliko se ugljičnog dioksida zapravo ispušta u atmosferu.

Sl. 2.2. *Potrošnja struje u podatkovnim središtima izražena u TWh u periodu 2010.-2030. preuzeto iz [21]*

Jedino velike tvrtke kao što su Google, Amazon i Facebook javno objavljuju podatke iz kojih se da zaključiti kako se sve te organizacije kreću u smjeru što većeg smanjenja ispuštanja CO_2 [9]. Ove godine objavljen je podatak kako se Google bavi završavanjem prvog geotermalnog projekta takve vrste u kome će se do 2030. godine nastojati u potpunosti iskorijeniti emisije ugljičnog dioksida u atmosferu [10].



Sl. 2.3. *Strujni „otisak“ komunikacijskih tehnologija izražen u TWh u periodu 2010.-2030. preuzeto iz [21]*

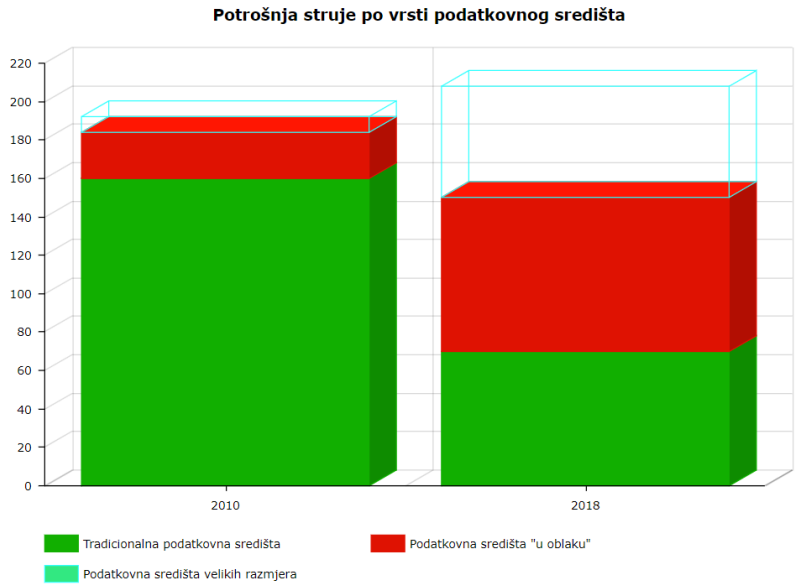
2.5. Energetska učinkovitost i ušteda energije u podatkovnim središtima

Energetska učinkovitost direktno je proporcionalna izlazu, a obrnuto proporcionalna ulazu energije u sustav što se vidi u jednadžbi 2-1.

$$\eta = \frac{Wizlaz}{Wulaz} \times 100\% \qquad (2\text{-}1)$$

Postoji još jedan način za definiranje energetske učinkovitosti koji kaže da je ona predstavlja korištenje što manje energije za obavljanje istog zadatka čime bi se smanjio energetski otpad, smanjilo ispuštanje ugljičnog dioksida, smanjile cijene za kućanstva ili bi se pak smanjila potreba za uvozom energije.

Slikom 2.4. prikazuje se potrošnja struje po podatkovnim središtima. Prikazana je potrošnja energije u podatkovnim središtima 2010. te 2018. godine. Odmah se primijeti dvostruko smanjenje potrošnje u tradicionalnim podatkovnim središtima, dok se za podatkovna središta u oblaku prikazuje trostruko povećanje u odnosu na vremenski period između 2010. te 2018. godine.



Sl. 2.4. *Potrošnja struje po vrsti podatkovnog središta izražena u TWh 2010. te 2018. godine*

Kad se govori o podatkovnim središtima (prema [12]), onda se tu mogu naći brojni načini kako smanjiti potrošnju energije, odnosno što učiniti kako bi se napravile određene uštede:

1. *Ujediniti slabo korištene poslužitelje*

   Većina poslužitelja ne radi ni blizu svog kapaciteta. Ujedinjenjem i uklanjanjem nepotrebnih sklopovlja može se uštediti ne samo na energiji nego i na ostalim izdacima. Uklanjanjem jednog poslužitelja može se uštedjeti oko 3 000 kn u energiji, 3 000 kn u licencama za operativni sustav i oko 10 000 kn troškova održavanja sklopovlja godišnje.

2. *Provesti učinkovite mjere za pohranu podataka*

   Dostupni su razni alati i tehnologije koji pomažu smanjiti količinu podataka koja se pohranjuje i koje mogu učinkovitije pohranjivati ono što je potrebno. Npr., program za uklanjanje duplikata može smanjiti količinu podataka pohranjenih u mnogim organizacijama za više od 95%.

3. *Smanjenje gubitaka energije iz distribucijskih jedinica (jedinice podatkovnih protokola)*

   Gubitak energije može se smanjiti korištenjem „pametnih“ i učinkovitijih jedinica podatkovnih protokola koji nadgledaju potrošnju energije. Jedinice podatkovnih protokola visoke učinkovitosti 2-3 % su učinkovitije od konvencionalnih jedinica.

4. *Smanjenje gubitaka energije iz sustava neprekidnog napajanja*

Energetski učinkoviti sustavi smanjuju električne gubitke i mogu raditi na "eko-način". Ispravno učitavanje takvih sustava također može uštedjeti energiju. Pokretanje sustava u ekološkom načinu rada može smanjiti troškove energije središta podataka za čak 2 %.

5. *Upravljanje protokom zraka da bi se osigurala učinkovitost hlađenja.*

   Jeftine uvodnice, difuzori i slijepe ploče mogu spriječiti miješanje hladnog zraka s vrućim ispušnim zrakom. Jedan veliki podatkovni centar godišnje uštedi oko 2290140,00 kn zahvaljujući jeftinim mjerama upravljanja protokom zraka.

6. *Raspored vrućih/hladnih prolaza*

   Redovi poslužitelja trebaju biti orijentirani tako da su prednji dijelovi poslužitelja okrenuti jedni prema drugima. Uz to, stražnja strana nosača poslužitelja također bi trebala biti okrenuta jedna prema drugoj, čineći vruće (ispušne) i hladne prolaze. Procjenjuje se smanjenje potrošnje energije ventilatora od 20% do 25%.

7. *Korištenjem ograničenja (pregrada)*

   Zavjese ili ploče od pleksiglasa mogu spriječiti miješanje hladnog zraka s vrućim zrakom koji dolazi od poslužitelja, smanjujući ukupne troškove hlađenja. U podatkovnim središtima a s uređenim prolazima toplo / hladno, sustavi za zadržavanje mogu smanjiti troškove energije za 5% do 10%.

8. *Instaliranjem hlađenja u redove poslužitelja*

   Ovi sustavi približavaju hladni zrak poslužiteljima. Mogu koristiti 3 puta manje energije u postolju poslužitelja visoke gustoće od konvencionalnih rashladnih sustava.

9. *Podešavanjem vlažnosti*

   Općenito govoreći, IT oprema može tolerirati velik raspon vlage. Ako morate dodati vlagu, upotrijebite energetski učinkovite tehnologije kao što su maglice i ultrazvučne jedinice.

10. *Korištenjem uređaja za vodu*

    Upotrebom rashladnog tornja umjesto mehaničkog hladnjaka za opskrbu rashlađenom vodom za hlađenje podatkovnog centra. Smanjenje troškove rashlađene vode do 70%.

11. *Korištenjem uređaja za zrak*

    Rashlađivanjem podatkovnog centra (gotovo) besplatno koristeći zrak izvana, ako to vremenske prilike dopuštaju. NetApp-ov podatkovni centar smanjio je potrošnju energije rashladnog sustava za gotovo 90 %.

12. *Upotreba senzora kako bi se uskladili kapacitet hlađenja i protok zraka s IT opterećenjima*

Upravljanje infrastrukturom podatkovnih centara je usklađivanje funkcija IT-a i zgrada, tako da se energija, oprema i prostor koriste što učinkovitije. Postoje informacije koje će mogu omogućiti da se "prilagodi veličina" infrastrukture i smanje troškove energije za čak 30 %.

13. *Usporedbom energetske učinkovitosti podatkovnog centra*

    Usporedbom objekta s podatkovnim središtima u cijeloj zemlji. Dovoljno visoka ocjena i može se zaraditi oznaka podatkovnog centra s energetskim certifikatom.

14. *Odabirom održivog kolokacijskog centra*

    Odabir objekta za kolokaciju s naglaskom na energetsku učinkovitost.

15. *Osiguranjem treninga o svijesti o energetskoj učinkovitosti*

    Energetska učinkovitost rijetko je dio obrazovanja IT stručnjaka. Uvođenje osnovnih koncepata energetske učinkovitosti u obrazovanje može navesti na razmišljanje o načinima štednje.

2.6. Ukratko o metrikama

U ranijem dijelu ovog poglavlja opisana su mjesta korištenja velikih računalnih sustava, opisivalo se za koju svrhu se koriste, ali i uzroke velike ili povećane potrošnje u podatkovnim središtima. Dodatno, dane su preporuke kako bi se moglo uštediti na potrošnji energije te povećati samu energetsku učinkovitost. U poglavlju 3, detaljno će biti predstavljene metrike PUE, DCiE, TGI i DWPE. Svaka od ovih metrika vrši svoj izračun za energetsku učinkovitost. Kako su izrazi za računanje metrika različiti, tako i rezultati variraju pa prema tome nije isto ako je izračun za PUE 1.7, a za DCiE 1.7%. Same po sebi prve tri metrike daju okvirnu sliku o potrošnji energije u podatkovnim središtima. Kad se kaže „daju okvirnu sliku“ to znači da one nisu potpune te da ne uzimaju neke druge parametre u obzir poput ukupne potrošnje energije na rad rashladne opreme kao što je to slučaj kod DWPE metrike.

2.7. Idejno rješenje mobilne aplikacije

Mobilna Android aplikacija treba biti iskorištena za računanje parametara koje će se unijeti od strane korisnika kako bi se dobio rezultat metrika PUE, DCE, TGI i DWPE. Kada su metrike izračunate, trebaju se prikazati na narednom zaslonu zajedno s rezultatima prethodnih izračuna te navigacija mora biti ostvarena kroz sve ekrane aplikacije. Cilj aplikacije je omogućiti korisniku pregled svih metrika, njihove formule te parametara koje formule sadrže, kao i prikaz izračuna nakon unesenih vrijednosti parametara. Metrike i model aplikacije bit će detaljnije opisane u trećem poglavlju.

3. PRIJEDLOG MODELA MOBILNE APLIKACIJE

3.1. Parametri za računanje energetske učinkovitosti po metrikama

Opisane metrike ukratko objašnjene u poglavlju 2 dobivaju se izračunom parametara koji su opisani u ovom poglavlju završnog rada. Iznimno je važno paziti na sve čimbenike koji utječu na ukupnu energetsku potrošnju, a naročito u posljednje vrijeme kad su cijene struje „skočile u nebesa“, a tek se očekuju novi valovi poskupljenja struje i energenata. Ono što se može napraviti je da se pažljivo pristupa načinu izgradnje podatkovnog središta od korištenja novije tehnologije pa sve do razmještaja opreme te hlađenja podatkovnog središta. Ukoliko, na primjer, raspored vrućih ili hladnih prolaza nije ispravan, doći će do povećane potrošnje uslijed zagrijavanja, a kao što smo to spomenuli u poglavlju 2.2 redovi poslužitelja trebaju biti pozicionirani na način da su prednji dijelovi poslužitelja okrenuti jedna prema drugima.

Naravno, ako podatkovno središte nema ugrađen sustav hlađenja u toplim/hladnim prolazima, onda to svakako troši više struje jer se server više zagrijava, a njegov interni sustav hlađenja, ne može dovoljno ohladiti poslužitelj. Zato se pribjegava ugradnji sustava klimatizacije kako bi se poslužiteljske sobe bolje i korisnije ohladile.

3.2. Metrike PUE i DCiE

3.2.1. Udruga *„The Green Grid“*

The Green Grid je neprofitna organizacija osnovana 2007. godine sa stajalištem da je energetska učinkovitost u podatkovnom centru jedno od najvažnijih pitanja s kojima se suočavaju dobavljači tehnologije i njihovi kupci. Green Grid je globalno udruženje posvećeno unapređivanju energetske učinkovitosti u podatkovnim centrima i poslovnim računalnim ekosustavima. Podatkovni centri su se bitno promijenili razvojem IT-a, njihov broj se drastično povećao kao odgovor na sve veće poslovne zahtjeve. To je rezultiralo velikim brojem visokoenergetskih centara. Od 2011. godine procijenjeno je da IT oprema čini 2% svjetskih emisija ugljika. Kao rezultat toga, upravitelji podatkovnih centara žele razvijati strategije kako bi podatkovni centri postali što učinkovitiji. Green Grid nastoji postati vodeća globalna organizacija u izradi energetski učinkovitijih podatkovnih centara i poslovnih računalnih sustava, zbog čega su osmislili dvije metrike PUE (engl. Power Usage Effectiveness) i DCiE (engl. Data Center Infrastructure Efficiency) koje su danas prihvaćene u svijetu [11].

3.2.2. Opis korištenih metrika i formule

Kao što je već ranije spomenuto, udruga *„The Green Grid“* osmislila je dvije metrike kako bi se bolje pratila i računala energetska učinkovitost podatkovnih središta:

- PUE – Učinkovitost korištenja energije (*engl. Power Usage Effectiveness*)
- DCiE - Učinkovitost infrastrukture podatkovnog središta (*engl. Data Center Infrastructure Efficiency*)

Ove dvije metrike nam omogućavaju procjenu energetske učinkovitosti nekog podatkovnog središta. Obje metrike su izuzetno korisne, no ipak je samo PUE metrika našla veliku publiku. Navedene metrike koriste iste parametre u svojim izračunima. Ukupna potrošnja električne energije u jednom podatkovnom središtu definira se kao energija koja se koristi samo za rad podatkovnog središta (ovo nije loše naglasiti jer postoje zgrade u kojima je podatkovno središte samo jedno od potrošača ukupne energije). Konačnu brojku ukupne potrošnje energije čine baterije, UPS-ovi, chilleri ili ostala rashladna oprema, generatori struje, rasvjeta, itd. Ukupna potrošnja električne energije na rad računalne opreme u podatkovnom središtu obuhvaća opremu koja se koristi za upravljanje, procesuiranje ili skladištenje podataka u podatkovnom središtu, ali isto tako i KVM „switch“-eve (omogućuju upravljanje s više računala odjednom preko jedne tipkovnice, miša i monitora) i računala koja se koriste za upravljanje i kontrolu podatkovnog središta.

PUE (*engl. Power Usage Effectiveness*) metrika [12] jednaka je omjeru ukupne potrošnje električne energije u jednom podatkovnom središtu i ukupne potrošnje električne energije na rad računalne opreme u podatkovnom središtu.

$$PUE = \frac{E_{UK}}{E_{IT}} \quad (2\text{-}2)$$

gdje je:

E_{UK} - ukupna potrošnja električne energije u jednom podatkovnom središtu
E_{IT} – ukupna potrošnja električne energije na rad računalne opreme u podatkovnom središtu

DCiE (*engl. Data Center Infrastructure Efficiency*) metrika [12] računa se na način da se uzme vrijednost PUE metrike i napravi recipročni izraz što se na kraju pomnoži sa sto jer se ova metrika izražava u postotcima. Dakle, DCiE je zapravo omjer ukupne potrošnje električne energije na rad

računalne opreme u podatkovnom središtu i ukupne potrošnje električne energije u jednom podatkovnom središtu.

$$DCiE = \frac{1}{PUE} = \frac{E_{IT}}{E_{UK}} * 100\% \quad (2\text{-}3)$$

No, koji su to iznosi, to jest vrijednosti ovih metrika koji se smatraju prihvatljivima? Kada se govori o vrijednostima za PUE, one mogu biti u rasponu od 1.0 pa do beskonačno, s tim da bi vrijednost od 1.0 bila idealna vrijednost sa 100 posto učinkovitosti (vrijednost PUE od 1.0 bi značila da je iznos snage potrebne za rad podatkovnog središta 1 puta veći od one snage koju daje računalna oprema).

Preliminarna istraživanja su pokazala kako većina podatkovnih središta ima PUE vrijednost od 3.0 ili više, ali i da bi s pravim dizajnom i rasporedom navedena vrijednost mogla pasti na 1.6. Slijedi jedan kratki primjer izračuna PUE i DCE. Ako imamo podatkovno središte koje ima ukupnu potrošnju električne energije od 100,000 kW od kojih se 80,000 kW koristi za potrošnju električne energije na rad informatičke opreme, to znači da će PUE imati vrijednost od 1.25, a DCE vrijednost 0.8, to jest 80 posto jer se rezultat DCE množi sa 100.

Ove metrike imaju prednosti i mane. Kada se govori o prednosti, ova metrika ima samo 2 parametra što izuzetno olakšava izračun, međutim, negativna strana je što rezultati mogu stvoriti krivu sliku o učinkovitosti podatkovnog središta. Isključivanjem rashladnog sustava dobivamo smanjenje potrošnje električne energije, ali zauzvrat dobijemo povećanu potrošnju cijelog sustava zato što se gašenjem klima uređaja oprema više zagrijava jer rashladni sustavi koji postoje u samim komponentama više energije potroše na okretanje kako bi prikladno rashladile komponente).

3.3. Metrika The Green Index

3.3.1. Nastanak, opis metrike i formule

Prije upotrebe metrike TGI, koristila se LINPACK metrika visokih performansi prilikom rangiranja za liste *„Green500“* i *„TOP500“*. Ta metrika uspoređuje performanse dva računala nakon što se svakom računalu zada niz linearnih jednadžbi u standardnoj formi i onda ovisno o rezultatu smješta ih negdje na listu *„TOP500“*. S vremenom se pokazalo da metrika LINPACK visokih performansi nije dovoljna sama po sebi jer prilikom ispitivanja uzima u obzir samo performanse procesora.

Na scenu stupa zavod za računarstvo *Virginia Polytehnic Institute and State University* koji je predložio novu metriku pomoću koje je moguće niz mjerenja objediniti u jedan broj – *The Green Index (Zeleni indeks). Zeleni indeks* rezultat je tri različite metrike: HPL, STREAM i IOzone. HPL je metrika za ispitivanje mogućnosti procesora te se prikazuje po broju operacija s pomičnim zarezom po sekundi (*engl. Floating-point operations per second* - FLOPS), STREAM je metrika vezana za memoriju, a IOzone metrika prikazuje ulazno-izlazni podsustav, a obje metrike imaju zajedničku mjernu jedinicu megabajt po sekundi (MB/s).

Temelj metrike *Zeleni indeks* je SPEC mjerenje (*engl. Standard Performance Evaluation Corporation*) koje za svoj izračun koristi referentni model, a referentni modeli su izuzetno korisno mjerilo jer se s napretkom i razvojem tehnologije mijenja referentni model, što znači da su podaci neprestano *„up to date“* [13].

TGI se dobiva nakon 4 koraka. Prvo se računa energetska učinkovitost EE prema izrazu 2-4. Ona je po formuli jednaka omjeru performansi sustava i ukupne potrošene snage sustava, a sam sustav se mjeri više puta kako bi konačni rezultat bio što točniji.

$$EE_i = \frac{performanse\ sustava}{potrošena\ snaga_i} \tag{2-4}$$

U narednom koraku potrebno je dobiti energetsku učinkovitost i to referentnu (REE) koja predstavlja omjer energetske učinkovitosti i rezultata referentnog sustava prikazanog u izrazu 2-5.

$$REE_i = \frac{EE_i}{EE_{Refi}} \tag{2-5}$$

Što se tiče mjernih jedinica za EE i REE, one su bezdimenzijske veličine.

Sljedeći korak je pridruživanje težinskog čimbenika W koji ima ukupni iznos 1, a dobije se zbrojem svih težinskih faktora čije formule možete vidjeti prikazane u nastavku za svaki pojedini faktor.

$$W_{t_i} = \frac{t_i}{\sum_i t_i} \tag{2-6}$$

pri čemu je:

t_i – vrijeme trenutnog mjerenja

$\sum t_i$ – zbroj svih vremena mjerenja

$$W_{p_i} = \frac{p_i}{\sum_i p_i} \quad (2\text{-}7)$$

pri čemu je:

t_i – vrijeme trenutnog mjerenja

$\sum t_i$ – zbroj svih vremena mjerenja

$$W_{e_i} = \frac{e_i}{\sum_i e_i} \quad (2\text{-}8)$$

pri čemu je:

t_i – vrijeme trenutnog mjerenja

$\sum t_i$ – zbroj svih vremena mjerenja

Posljednji korak računanja *Zelenog indeksa* je formula 2-9, koja kaže da je on suma umnožaka svih težinskih faktora te relativnih energetskih učinkovitosti za svako pojedino mjerenje.

$$TGI = \sum_i W_i * REE_i \quad (2\text{-}9)$$

Ovaj rad neće se detaljno baviti težinskim faktorima, nego će se fokusirati na jednostavniju računicu koja se bazira na aritmetičkoj sredini.

3.3.2. Metoda aritmetičke sredine za izračun *Zelenog indeksa*

Kao što je već spomenuto u prethodnom podpoglavlju, ovaj rad će se baviti izračunom Zelenog indeksa korištenjem aritmetičke sredine (TGI_{AS}) gdje svi proračuni imaju identičan težinski faktor (W). U nastavku rada možete pogledati raspis formule. [13]

$$TGI_{AS} = \frac{\sum_{i=0}^{n} REE_i}{n} \quad (2\text{-}10)$$

$$TGI_{AS} = \frac{\sum_{i=0}^{n} \frac{EE_i}{EE_{Ref_i}}}{n} \quad (2\text{-}11)$$

$$TGI_{AS} = \frac{\sum_{i=0}^{n} \frac{performanse\ sustava}{potrošena\ snaga_i}}{n * EE_{Ref_i}} \quad (2\text{-}12)$$

pri čemu je:

i – početni broj mjerenja

n – posljednji (konačan) broj mjerenja

$\sum REE_i$ – suma relativnih energetskih učinkovitosti

EE_i – energetska učinkovitost stvarnog mjernog sustava

EE_{Ref} – energetska učinkovitost referentnog sustava

U praktičnoj primjeni za potrebe rada mobilne aplikacije će se koristiti parametar „n“ s iznosom 1 jer će postojati samo jedno mjerenje. Prema tome, u aplikaciji će se u konačnici koristiti izraz 2-13:

$$TGI = \frac{EE_i}{REE_i} \quad (2\text{-}13)$$

Sada još ostaje tumačenje rezultata koji može biti u jednoj od tri kategorije:

- TGI < 1 – manja učinkovitost u odnosu na referentni sustav
- TGI = 1 - jednaka učinkovitost u odnosu na referentni sustav
- TGI > 1 - veća učinkovitost u odnosu na referentni sustav

Već je ranije u radu spomenuta prednost TGI metrike (u okviru objašnjavanja SPEC mjerenja), a to je činjenica da se metrika temelji na referentnom sustavu koji se mijenja i ažurira kako se tehnologija razvija. Istovremeno je referentni sustav sam po sebi nedostatak zato što se može dogoditi da su podatci za neki referentni sustav javno nedostupni stoga nije moguće odraditi izračune.

3.4. Metrika DWPE

3.4.1. O nastanku metrike

Iako su prethodno navedene metrike doživjele značajan uspjeh (pomoću PUE metrike povećala se energetska učinkovitost podatkovnih središta s prosjeka 2.7 u 2007. godini na 1.65 u 2013. godini), one samostalno nisu davale realnu sliku o energetskoj učinkovitosti u podatkovnim središtima. Čak i obje metrike zajedno nisu davale ispravnu sliku o energetskoj učinkovitosti jer se karakteristike podatkovnih središta razlikuju od jednog do drugog. Prema tome, 2014. godine, grupa autora s bavarske akademije znanosti i umjetnosti je uvela novu kompletnu metriku koja

može izračunati koliko je energetski učinkovit neki veliki računalni sustav pod određenim radnim opterećenjem u određenom podatkovnom središtu.

3.4.2. **Opis metrike DWPE**

Energetska učinkovitost radnog opterećenja podatkovnog središta ili skraćeno DWPE (*engl . Data Center Workload Power Efficiency*) računa se prema [14] prema sljedećoj formuli:

$$DWPE = \frac{WPE}{sPUE} \quad (2\text{-}14)$$

pri čemu je:

WPE – energetska učinkovitost radnog opterećenja

sPUE - učinkovitost potrošnje snage sustava

Prema [14], WPE (*engl. Workload Power Efficiency*) predstavlja energetsku učinkovitost radnog opterećenja. Ova metrika jednaka je omjeru prosječno dostignutih performansi i prosjeka potrošene električne energije velikih računalnih sustava te na ovaj način pokriva potrošnju električne energije u svakom pojedinom podsustavu koji se nalazi u „clusteru“. To prikazuje izraz 2-16:

$$WPE = \frac{prosječno\ postignute\ performanse}{prosjek\ potrošene\ el.energije\ HPC\ sustava} \quad (2\text{-}15)$$

Mjerna jedinica za WPE je FLOPS/W.

Preostali parametar sPUE (*engl. System Power Usage Effectivness*) prikazuje učinkovitost snage računalnog sustava. Ovaj parametar uzima u obzir koliko je ukupno električne energije potrošio cijeli podatkovni centar, samo što se prilikom izračuna ta potrošnja „razbija“ na tri dijela P_{IT}, P_{PDCL} i $P_{COOLING}$. Računa se prema sljedećem izrazu:

$$PUE = \frac{P_{IT} + P_{PDCL} + P_{COOLING}}{P_{IT}} \quad (2\text{-}16)$$

pri čemu je:

P_{IT} – potrošnja energije u podatkovnom središtu na rad informatičke opreme

P_{PDCL} – potrošnja energije u podatkovnom središtu na pretvaranje i raspodjelu električne energije

$P_{COOLING}$ - potrošnja energije u podatkovnom središtu na hlađenje .

Sve jednadžbe su preuzete iz [14], a za potrebe praktičnog dijela rada koristit će se izraz :

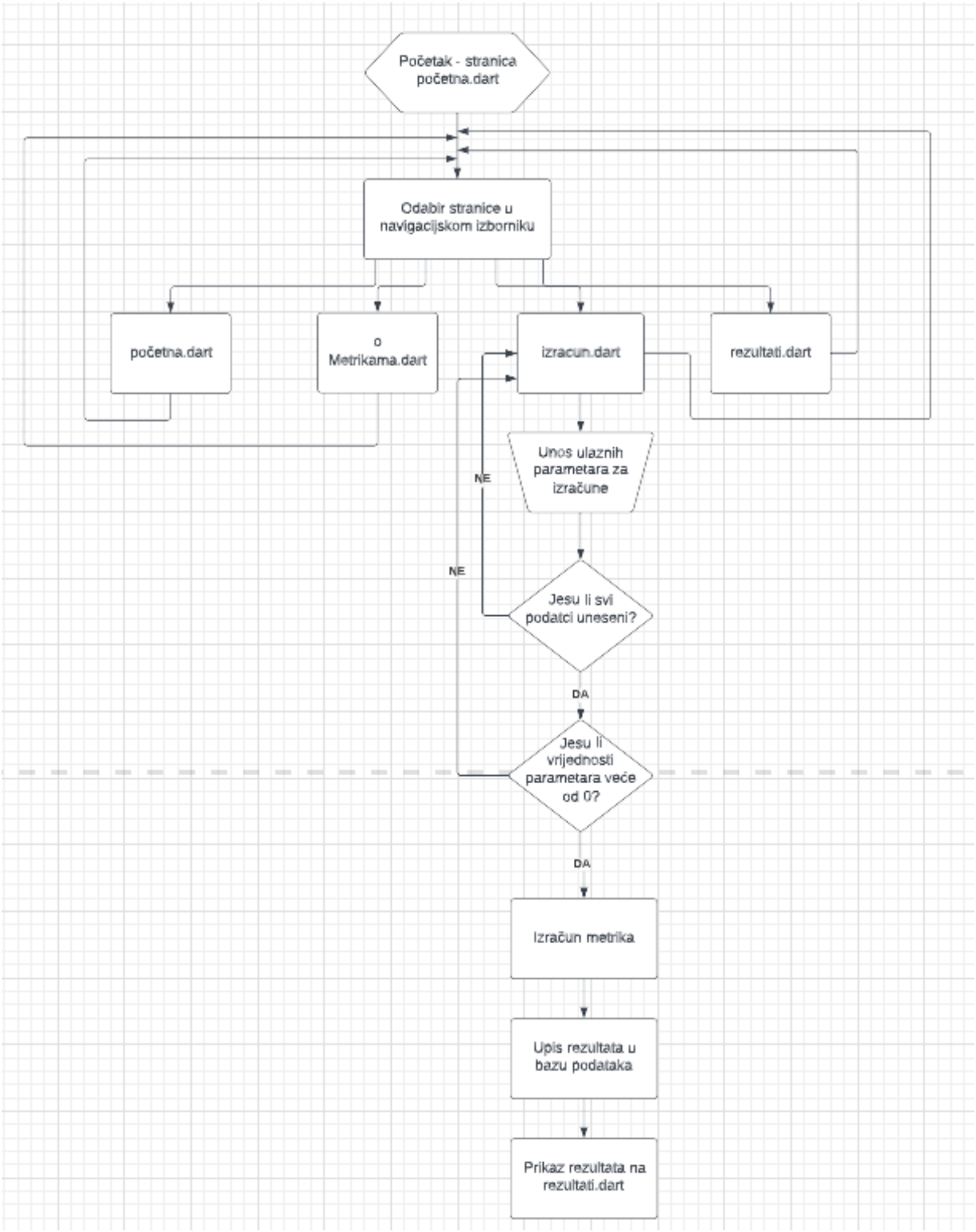
$$DWPE = \frac{\frac{prosječno\ dostignute\ performanse}{prosječno\ potrošena\ el.energija\ HPC\ sustava}}{\frac{P_{IT} + P_{PDLC} + P_{COOLING}}{P_{IT}}} \quad (2\text{-}17)$$

Ova metrika također ima nedostataka, upravo zbog nedostupnosti ulaznih podataka te zbog najvećeg broja ulaznih parametara od svih metrika koje smo opisali, ali je upravo zbog složenosti izračuna rezultat ove metrike možda i najtočniji jer se mnogo faktora obuhvaća u izračunu.

3.5. Funkcionalni zahtjevi mobilne aplikacije

Kada se govori o funkcionalnim zahtjevima mobilne aplikacije, ona zahtjeva da korisnik poznaje parametre koje treba unijeti prije nego se oni unesu. Kada se unesu svi potrebni parametri za izračun, onda se pritisne gumb na kraju stranice koji vrši izračun metrika i sprema rezultate izračuna u bazu podataka.

Kao što je prikazano na slici 2.5 pristupom aplikaciji korisniku će se prikazat početni zaslon aplikacije, a nakon čega će se u navigacijskom izborniku moći odabrati bilo koji zaslon aplikacije, no isto tako će se na prvom zaslonu nalaziti dodatni gumb za odlazak na drugi zaslon, na drugom gumb za treći, itd. Sljedeća funkcionalnost bit će ostvarena na zaslonu za izračun gdje će se korisniku omogućiti unos parametara u poljima za unos podataka, a u isto vrijeme ga sprječavajući da unese lošu vrijednost tako što će se napraviti dvije provjere. Prvom provjerom bit će onemogućeno ostaviti prazno polje za unos parametara, a drugom će provjerom korisnik biti spriječen u izračunu metrika te odlasku na naredni zaslon ukoliko unese vrijednost nula ili manje od nula. Ostvarena je komunikacija s bazom podataka u koju se spremaju rezultati izračuna za svaku pojedinu metriku dok su isti ti rezultati prikazani na zaslonu *Rezultati*.



Sl. 3.1. *Dijagram toka mobilne Android aplikacije za analizu energetske učinkovitosti velikih računalni sustava*

4. PROGRAMSKO RJEŠENJE APLIKACIJE ZA ANALIZU ENERGETSKE UČINKOVITOSTI

Zasebno objasnivši svaku metriku koje se koriste prilikom računanja energetske učinkovitosti velikih računalnih sustava, red je objasniti programsko rješenje mobilne Android aplikacije.

4.1. Korišteni alati i tehnologije

Pri izradi mobilne Android aplikacije korišteni su sljedeći programski alati i/ili programski jezici:

- Dart
- Visual Studio Code
- Flutter
- Firebase

Detaljne opise svih navedenih programskih alata i/ili jezika možete pronaći u narednim podpoglavljima.

4.1.1. Programski jezik Dart

Dart je „open-source“ objektno-orjentirani jezik sa sintaksom koja podsjeća na C. Razvio ga je Google 2011. godine, a svrha je bila napraviti „frontend“ korisnička sučelja za web i mobilne aplikacije. Razvija se još uvijek, a inspiraciju vuče iz ostalih programskih jezika poput C#, Jave, itd. Iako je objektno-orjentiran jezik koji podržava neke temeljne koncepte takvih programskih jezika, on ipak ne podržava polja, nego ima u sebi integrirane „kolekcije“ koje se koriste umjesto polja [15].

Dart je jezik optimiziran za klijenta. Dart također čini temelj Fluttera. Dart pruža jezik i vrijeme izvođenja koji pokreću Flutter aplikacije, ali Dart također podržava mnoge osnovne zadatke razvojnog programera kao što su formatiranje, analiza i testiranje koda.

4.1.2. Razvojno okruženje Visual Studio Code

Visual Studio Code je „lakša“ inačica razvojnog okruženja Visual Studio. Edit-build-debug ciklus je mnogo brži što znači da se za manje vremena može odraditi „debagiranje“ aplikacije. Visual Studio Code podržava Intellisense, a između ostalog i brojne programske jezike poput programskog jezika C, C++, Dart, Java, Phyton, PHP, i tako dalje [16].

4.1.3. Razvojni okvir Flutter

Flutter je razvojni program otvorenog koda koga je kreirao Google. Koristi se kako bi se razvijale aplikacije namijenjene za razne platforme, Android, iOS, Linux, MacOS, Windows, Google Fuchsia, i web aplikacije sa jednom bazom koda [17].

Prvi put opisan 2015. godine, Flutter je objavljen za javnost u svibnju 2017. godine. Flutterov kod, pisan primarno u C++, nudi jednostavno prikazivanje podrške koristeći Google Skia grafičku knjižnicu. Pomoću Flutter framework-a programski jezik Dart dobiva posve novu razinu jer se „dodavanjem“ Flutter-a otvara posve nova dimenzija razvoja jedne aplikacije (sam izgled aplikacije, widget-i, itd.). Postoje 3 vrste widget-a (widget bi se na hrvatskom jeziku mogao prevesti kao komponenta sučelja): Stateless, Stateful i widgeti koji su naslijeđeni. Zajednička osobina svih widget-a je da su nepromijenjivi, ali, na primjer, Stateful widget-i omogućuju interakciju između korisnika i aplikacije pomoću metode „setState“. Flutter dodatno ima skup library-ja (u Flutter-u se oni zovu „package-s“) koje može koristiti kako bi se olakšao razvoj aplikacije, a koje to sve „pakete“ korisnik može koristiti mogu se naći na službenoj stranici Fluttera (https://pub.dev). Flutter ima podršku za sljedeća razvojna okruženja: IntelliJ, Android Studio, Visual Studio Code, Emacs.

4.1.4. Firebase

Firebase je platforma za razvoj aplikacija Backend-as-a-Service (BaaS) kojom se pružaju hostirane pozadinske usluge kao što su baza podataka u stvarnom vremenu, pohrana u „oblaku“, prijavljivanje „crash-ova“, strojno učenje, udaljena konfiguracija [18].

Dostupno s klijentskih uređaja - bazi podataka u stvarnom vremenu Firebase može se pristupiti izravno s mobilnog uređaja ili web preglednika; nema potrebe za aplikacijskim poslužiteljem. Sigurnost i provjera valjanosti podataka dostupni su putem *Firebase Realtime Database Security Rules*, pravila temeljenih na izrazima koja se izvršavaju kada se podaci čitaju ili pišu.

„Baza podataka u stvarnom vremenu“ je je NoSQL baza podataka i kao takva ima različite optimizacije i funkcionalnost u usporedbi s relacijskom bazom podataka.

API baze podataka u stvarnom vremenu dizajniran je tako da dopušta samo operacije koje se mogu brzo izvršiti a što omogućuje da se izgradi sjajno iskustvo u stvarnom vremenu koje može služiti milijunima korisnika bez kompromisa u pogledu odziva.

4.2. Prikaz glavnih dijelova programskog koda

4.2.1. Bočna navigacijska traka

Iako se ulaskom u aplikaciju korisniku otvori „*Početni zaslon*“, i dalje se može odabrati na koji bi se zaslon prebacilo, a to bi bili rezultati prethodnih izračuna ili pak neke informacije o metrikama. Kako bi se to moglo postići napravljena je bočna navigacijska traka kojom se korisniku aplikacije omogućuje takav odabir. U nastavku na slikama od 4.1. do 4.3. se nalaze dijelovi programskog koda zaslužni za implementaciju navedene trake. Slika 4.1. prikazuje klasu *CustomDrawer* koja „produžuje“ značajku sa stanjem te se sastoji od četiri znakovna niza u listi koja će se prikazivati svaki put kad se navigacijski izbornik otvori. Počevši od linije koda 22 prikazuje se slaganje

```
7   class CustomDrawer extends StatefulWidget {
8     const CustomDrawer({Key? key}) : super(key: key);
9
10    @override
11    State<CustomDrawer> createState() => _CustomDrawerState();
12  }
13
14  class _CustomDrawerState extends State<CustomDrawer> {
15    List<String> titleDrawer = [
16
17      Type: List<String>
18      "Izracun",
19      "Rezultat",
20    ];
21
22    @override
23    Widget build(BuildContext context) {
24      return Drawer(
25        backgroundColor: const Color(0xff01080F),
26        child: SafeArea(
27          child: Padding(
28            padding: const EdgeInsets.symmetric(vertical: 20.0),
29            child: Column(
30              children: [
31                SizedBox(
32                  height: 200,
33                  child: Image.asset(
34                    "assets/drawer.png",
35                    fit: BoxFit.contain,
36                  ), // Image.asset
37                ), // SizedBox
38                ListView.builder(
39                  shrinkWrap: true,
40                  itemBuilder: (context, index) => InkWell(
```

Sl. 4.1. *Programski kod za navigacijsku traku 1*

značajki (engl. *widget*). U Flutter okviru smatra se temeljnom filozofijom izgradnja zaslona aplikacije od značajki pa se tako unutar temeljne značajke ista nadopunjuje pomoću parametra *child* ili *children* kao što se može vidjeti na slici 4.1 u linijama koda od 23 do 30. Unutar tog parametra značajke nalazi se poziv na drugu značajku *SafeArea* koji izbjegava sučelja korisničkim sustava, a njegov parametar *child* poziva *Padding zbog* kojeg će se njegovom „djetetu“ biti omogućeno namještanje prikladne pozicije na ekranu. U konačnici imamo „dijete“ *Column* u kojem se nalazi ostatak koda, a koji kao parametar ima *children* što znači da se može naslijeđivati više značajki odjednom koje će pritom biti poredane u stupcu.

```
4  abstract class Routes {
5    Routes._();
6    static const IZRACUNPARAM = _Paths.IZRACUNPARAM;
7    static const REZULTATI = _Paths.REZULTATI;
8    static const OMETRIKAMA = _Paths.OMETRIKAMA;
9    static const POCETNA = _Paths.POCETNA;
10 }
11
12 abstract class _Paths {
13   _Paths._();
14   static const IZRACUNPARAM = '/izracun';
15   static const REZULTATI = '/rezultat';
16   static const OMETRIKAMA = '/oMetrikama';
17   static const POCETNA = '/pocetna';
18 }
```

Sl. 4.2. *Programski kod za navigacijsku traku 2*

Na slikama 4.2.i 4.3 se prikazuje apstraktna klasa *Routes* koja služi za određivanje statičkih konstanti koje označavaju putanje kojima se korisnika odvodi na sljedeći ekran. Apstraktna klasa *_Paths* sastoji se od upotpunjene putanje.

```
12  class AppPages {
13    AppPages._();
14
15    static const INITIAL = Routes.POCETNA;
16
17    static final routes = [
18      GetPage(
19        name: _Paths.IZRACUNPARAM,
20        page: () => const Izracun(),
21        binding: HomeBinding(),
22      ), // GetPage
23      GetPage(
24        name: _Paths.REZULTATI,
25        page: () => const Rezultati(),
26        binding: ResultBinding(),
27      ), // GetPage
28      GetPage(
29        name: _Paths.OMETRIKAMA,
30        page: () => const oMetrikama(),
31        binding: HomeBinding(),
32      ), // GetPage
33      GetPage(
34        name: _Paths.POCETNA,
35        page: () => const Pocetna(),
36        binding: HomeBinding(),
37      ), // GetPage
38    ];
39  }
```

Sl. 4.3. *Programski kod za navigacijsku traku 3*

4.2.2. **Inicijalizacija Firebase baze podataka**

Kako bi se uspostavila komunikacija s Firebase-om potrebno je inicijaliziranje instance baze podataka unutar *main* funkcije kao što je to prikazano u liniji koda broj 10 na slici 4.4. Funkcija *main* sastoji se od linija koda za deklariranje i instanciranje baze podataka ispod kojih se nalazi funkcija *runApp* kojom se omogućujje prikaz značajki na zaslonu.

```
8   void main() async {
9     WidgetsFlutterBinding.ensureInitialized();
10    await Firebase.initializeApp();
11    runApp(
12      GetMaterialApp(
13        theme: ThemeData(scaffoldBackgroundColor: const Color(0xff04121F)),
14        debugShowCheckedModeBanner: false,
15        title: "Analiza energ. učinkovitosti HPC sustava",
16        initialRoute: AppPages.INITIAL,
17        getPages: AppPages.routes,
18      ), // GetMaterialApp
19    );
20  }
21
```

Sl. 4.4. *Inicijalizacija baze podataka*

No, kako bi se uopće mogla obaviti inicijalizacija baze, prvo se u istoj .dart datoteci mora na vrhu dodati paket koji omogućuje inicijalizaciju kao što je to prikazano na sljedećoj slici.

```dart
import 'package:firebase_core/firebase_core.dart';
```

Sl. 4.5. *Dodavanje paketa s bazom podataka Firebase*

4.2.3. Spremanje podataka u bazu

Spremanje podataka u Firebase zapisano je i omogućeno kodom koji se nalazi na slici 4.6.

```dart
addData()async {
isLoading.value=true;
    var bodyParam = {
      "DCIE": DCiE?.toStringAsFixed(2),
      "DWPE": DWPE?.toStringAsFixed(2),
      "PUE": PUE?.toStringAsFixed(2),
      "TGI":TGI?.toStringAsFixed(2),
      "created":DateTime.now()
    };
    firestore
        ?.collection("Result").
    add(bodyParam)
        .then((value) => print(""))
        .catchError((error) => print("Failed to add user: $error"));
    isLoading.value=false;
  }
```

Sl. 4.6. *Dio koda za spremanje podataka u bazu*

Funkcijom *addData* skladišti se programski kod pomoću kojeg se spremaju izračunati podaci za svaku metriku zasebno u bazu podataka.

4.2.4. Čitanje podataka iz baze

Suprotno spremanju rezultata analize energetske učinkovitosti je čitanje podataka iz baze koje se obavlja pomoću funkcije *getData,* a na narednoj slici može se vidjeti i prikladni programski kod funkcije za obavljanje dohvaćanja podataka iz baze. Napravi se instanca baze podataka pomoću koje se dohvaćaju podatci spremljeni u bazu podataka.

```dart
  getData() async {
    querySnapshot = await FirebaseFirestore.instance
        .collection("Result")
        .orderBy("created", descending: false)
        .get();
    isLoading.value = false;
  }
}
```

Sl. 4.7. *Programski kod za dohvaćanje iz baze podataka*

4.2.5. Opis programskog koda korišten za izračun

Kao što je prikazano na sljedećoj stranici rada na slici 4.8. gdje se mogu vidjeti prikazane sve izračune korištene u aplikaciji. Linije koda od 5 do 15 prikazuju deklaracije varijabli u obliku upravljača u kojemu je sadržano svojstvo „slušanja“ teksta kojeg tek treba ukucati u dijelu aplikacije predviđenom za ukucavanje parametara koje treba izračunati. Ovo je moguće u sklopu Flutter-ovog okvira. Kako ovdje nije deklarirana varijabla tipa *double*, nego upravljač (engl. *controller*) koji čeka da se ukuca broj tipa *double*, onda se ne mora dodijeliti početna vrijednost nula. Linija koda 28 prikazuje izračun za PUE, a linija koda 32 za DCE metriku koje su objašnjene u poglavlju 2.6. Izračun za *Zeleni indeks,* objašnjen u poglavlju 2.7, nalazi se u linijama koda 36 i 37 dok se za DWPE metriku, čije objašnjenje možete pronaći u poglavlju 2.8, izračun može naći od linije koda broj 41 pa sve do linije koda 45.

```
4  class HomeController extends GetxController {
5    TextEditingController pueTEC1 = TextEditingController();
6    TextEditingController pueTEC2 = TextEditingController();
7    TextEditingController tgiTEC1 = TextEditingController();
8    TextEditingController tgiTEC2 = TextEditingController();
9    TextEditingController tgiTEC3 = TextEditingController();
10   TextEditingController tgiTEC4 = TextEditingController();
11   TextEditingController dwpeTEC1 = TextEditingController();
12   TextEditingController dwpeTEC2 = TextEditingController();
13   TextEditingController dwpeTEC3 = TextEditingController();
14   TextEditingController dwpeTEC4 = TextEditingController();
15   TextEditingController dwpeTEC5 = TextEditingController();
16   final homeFormKey = GlobalKey<FormState>();
17   double? PUE;
18   double? DCiE;
19   double? TGI;
20   double? DWPE;
21
22   @override
23   void onInit() {
24     super.onInit();
25   }
26
27   void calculatePUE() {
28     PUE = double.parse(pueTEC1.text) / double.parse(pueTEC2.text);
29   }
30
31   void calculateDCiE() {
32     DCiE = (double.parse(pueTEC2.text) / double.parse(pueTEC1.text)) * 100;
33   }
34
35   void calculateTGI() {
36     TGI = (double.parse(tgiTEC1.text) / double.parse(tgiTEC2.text)) /
37         (double.parse(tgiTEC3.text) / double.parse(tgiTEC4.text));
38   }
39
40   void calculateDWPE() {
41     DWPE = (double.parse(dwpeTEC1.text) / double.parse(dwpeTEC2.text)) /
42         ((double.parse(dwpeTEC3.text) +
43                 double.parse(dwpeTEC4.text) +
44                 double.parse(dwpeTEC5.text)) /
45             double.parse(dwpeTEC3.text));
46   }
47 }
48
```

Sl. 4.8. *Programski dio koda za deklaraciju varijabli te za izračune metrika*

Prije nego li se obavi sav izračun u programskom dijelu koda koji je prikazan na slici 4.9, ostvaren je dio koda kojim se korisniku onemogućuje izračun metrika te odlazak na naredni ekran ukoliko nisu ispunjeni određeni uvjeti.

```
3  class TextFieldCustom {
4    static TextFormField textField(
5        {required TextEditingController textEditingController,
6        required String hntText,
7        TextAlign textAlign = TextAlign.center,
8        Color fillColour = Colors.white,
9        TextStyle hintTextStyle = const TextStyle(color: Color(0xff686c70)),
10       TextInputType keyBoardType = TextInputType.number,
11       TextStyle textStyle =
12           const TextStyle(fontSize: 22.0, color: Color(0xff01080F))}) {
13     return TextFormField(
14       textAlign: textAlign,
15       validator: (value) {
16         if (value == null || value.isEmpty) {
17           return 'Molimo unesite vrijednost!';
18         }
19         if (double.parse(value) < 1) {
20           return 'Molimo unesite vrijednost veću od 0!';
21         }
```

Sl. 4.9. *Programski dio koda za provjeru unesenih vrijednosti*

Navedeno je ostvareno u klasi *TextFieldCustom* u linijima koda od 16 do 20 gdje se nalazi kod s dva *if* grananja. Ukoliko prvi ili drugi uvjet unosa vrijednosti nije ispunjen, ispiše se poruka kojom se korisnika obaviještava kako se od njega zahtijeva unos vrijednosti, odnosno kako se od njega zahtijeva unos vrijednosti veće od nule.

5. NAČIN KORIŠTENJA I ISPITIVANJE RADA APLIKACIJE

Peto poglavlje se bavi prikazivanjem načina rada mobilne aplikacije te njenim funkcionalnostima. Još dodatno se prikazuju ispitni slučajevi koji prikazuju način na koji radi aplikacija.

5.1. Način rada mobilne aplikacije

Nakon što se instalira aplikacija na mobilnom telefonu, korisniku se prikazuje zaslon s osnovnim informacijama o zadatku završnog rada te kratki zaključak rada. Na kraju klizajućeg dijela zaslona nalazi se gumb *Opis metrika* kojim se korisnika prebacuje na sljedeći zaslon *O metrikama*. U gornjem lijevom kutu svakog zaslona se još dodatno nalazi gumb čijim se pritiskom otvara „ladica“, to jest, navigacijski izbornik koji sadrži četiri opcije za četiri zaslona koje aplikacija ima: *Početna*, *O metrikama*, *Izračun* te *Rezultat*. Iz navigacijskog izbornika se izlazi tako da se „ladica“ „klizne nazad“ ili se pritisne na bilo koji dio zaslona na kojem se ne nalazi navigacijski izbornik.

Nakon što se pritisne gumb *Opis metrika* ili opcija za odlazak na zaslon *O metrikama* korisnika se upućuje na naredni zaslon na kojem se nalazi kratko objašnjenje za svaku pojedinu metriku, kao i izrazi koji se koriste prilikom izračuna metrike te naziv parametara koji se koriste u tim istim izrazima. Iako su pojedine formule šire od širine zaslona, vodoravnim kliznikom se korisniku omogućio pogled na cijeli izraz. Okomiti kliznik je zaslužan za prikaz ostatka metrika te njihovih izraza, na čijem se kraju nalazi gumb *Idi na izračun* čijim pritiskom se odlazi na stranicu za unos parametara potrebnih za izračun metrika. Pritiskom na gumb *Idi na izračun* ili pritiskom na dio navigacijske trake s nazivom *Izračun* odlazi se na sljedeći zaslon na kojem su prikazana polja za unos parametara svake pojedine metrike. Okomitim kliznikom bit će prikazan i ostatak polja za unos parametara kao i gumb *Izračunaj i sačuvaj*. Ukoliko se pritisne na bilo koje polje otvorit će se brojčana tipkovnica pomoću koje će se obaviti unos podataka. Brojčana tipkovnica se još sastoji od tipke za brisanje brojeva, tipke za zatvaranje brojčane tipkovnice te tipke za unos decimalne točke ili znaka minus. Od korisnika se zahtijeva unos vrijednosti u svako polje za unos podataka. U slučaju da se dogodi da korisnik ne unese vrijednost u neko polje ispod tog polja pojavit će se odgovarajuća poruka kojom se obavještava korisnika kako je potrebno unijeti podatak te da polje ne smije biti prazno. Ukoliko se dogodi unos vrijednosti nula ili brojeva manjih od nule ispod jednog polja ili više njih, ovisno o tome u koliko polja nije unesena vrijednost veća od nule, ispod svakog se prikaže odgovarajuća poruka kojom se korisnika obavijesti kako je potreban unos broja većeg od nula. Pritiskom na gumb *Izračunaj i sačuvaj* ili odabirom dijela navigacijskog izbornika Rezultat se odlazi na posljednji zaslon aplikacije na kojem su prikazani rezultati.

Na posljednjem zaslonu nalaze se rezultati prethodnih mjerenja poredani jedni iza drugoga u nizu tablice okomito prema dolje, a na kraju tablice nalazi se rezultat posljednjeg izračuna metrika. Posljednji skup rezultata u nizu je naglašen posebnom bojom kako bi se istaklo da je to zadnji izračunan skup podataka. Podatci u tablici sastoje se od pet stupaca, prvim se označava broj mjerenja, drugim PUE metrika, trećim DCE, četvrtim TGI metrika, a petim stupcem se označava DWPE metrika. Kao i na svakom prethodnom zaslonu u gornjem lijevom kutu nalazi se gumb za navigacijski izbornik.

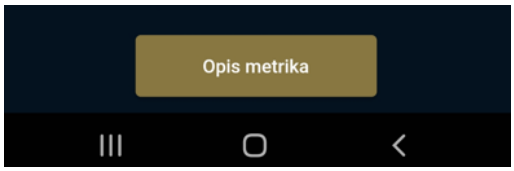
5.2. Ispitivanje rada aplikacije

5.2.1. Pristupanje i početni zaslon aplikacije

Pristupanje mobilnoj aplikaciji započinje ulaskom u aplikaciju te se otvara početni zaslon na kojem ukratko pišu neki detalji vezani za završni rad. Generalno, početni zaslon sastoji se iz dva dijela, prvi dio sadrži jednu fotografiju, a zatim i naredni dio kojim se ukratko prikazuju o zadatku rada, kao i zaključak o radu.

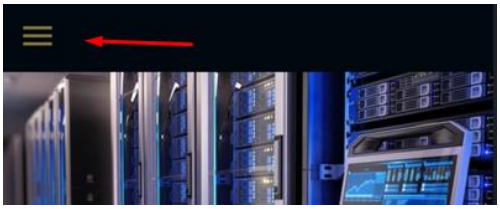
Sl. 5.1. *Početni zaslon*

Na kraju klizajućeg dijela zaslona nalazi se gumb *Opis metrika* kojim se korisnika prebacuje na sljedeći zaslon *O metrikama*.



Sl. 5.2. *Gumb „Opis metrika“*

U gornjem lijevom kutu svakog zaslona se još dodatno nalazi gumb čijim se pritiskom otvara „ladica“, to jest, navigacijski izbornik u kome su sadržane opcije za odlazak na ostale ekrane.



Sl. 5.3. *Gumb za navigacijski izbornik*

5.2.2. Navigacijski izbornik

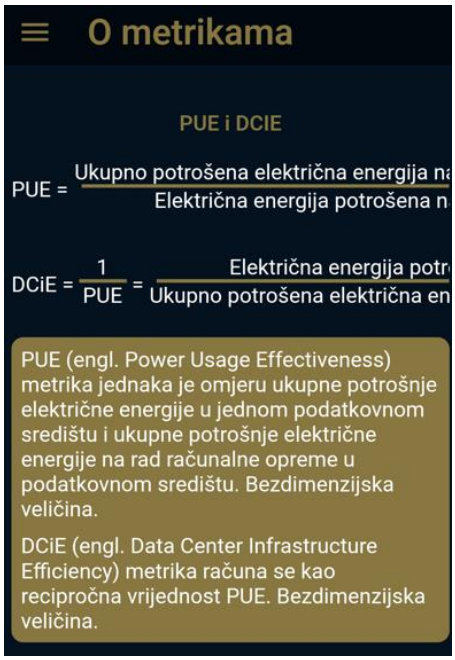
Kao što je spomenuto u prethodnom poglavlju, korisniku se klikom na lijevi gornji kut zaslona otvara navigacijski izbornik za biranje zaslona aplikacije. Bira se između četiri kategorije, ali može se ostati i na istom zaslonu. Navigacijski izbornik sastoji se od četiri opcije za četiri zaslona koje aplikacija ima: *Početna, O metrikama, Izračun* te *Rezultat*. Iz navigacijskog izbornika se izlazi tako da se „ladica“ „klizne nazad“ ili se pritisne na bilo koji dio zaslona na kojem se ne nalazi navigacijski izbornik. U nastavku slijedi snimka zaslona koja prikazuje rješenje navigacijskog izbornika.

Sl. 5.4. *Navigacijski izbornik*

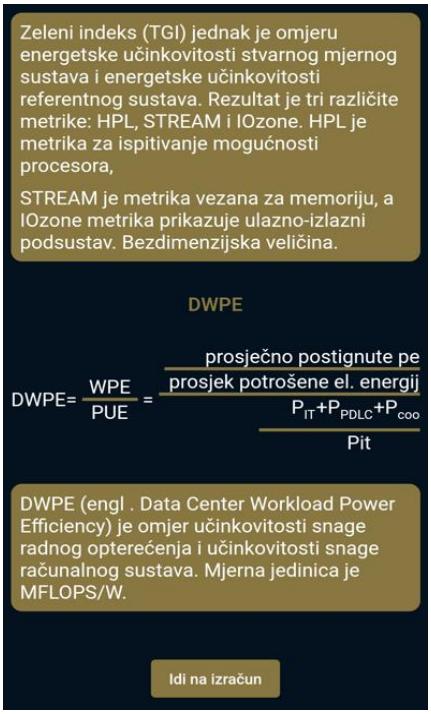
5.2.3. Drugi zaslon „*O metrikama*“

Na slikama 5.5. i 5.6. možete vidjeti sadržaje zaslona „*O metrikama*“. Sastoji se iz dva dijela. U svakom „kontejneru“ (jedna od vrsta widget-a koji se koriste u Flutter-u) nalaze se naziv metrike ili formule za svaku pojedinu metriku ili kratki opisi metrika kao i informacija o tome koju mjernu jedinicu koriste ako je imaju. Dakle, na ekranu se nalaze četiri metrike i opisi metrika.

Prvo se prikazuje formula pa se opisuju PUE i DCiE, a nakon njih slijede i ostale dvije metrike Zeleni indeks i DWPE.

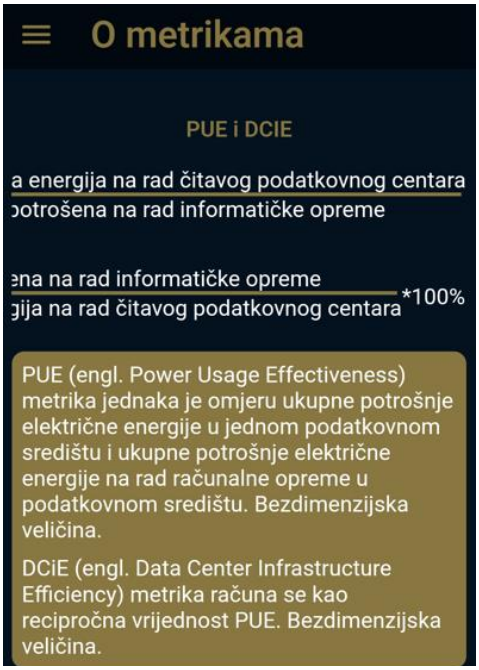


Sl. 5.5. *Zaslon „O metrikama“*



Sl. 5.6. *Zaslon „O metrikama“*

Kod izraza koji imaju veću dužinu nego što je to bilo moguće smjestiti na zaslon, nalaze se vodoravni kliznici pomoću kojih možemo vidjeti čitav izraz. Na kraju zaslona nalazi se gumb koji korisnika šalje na narednu stranicu.



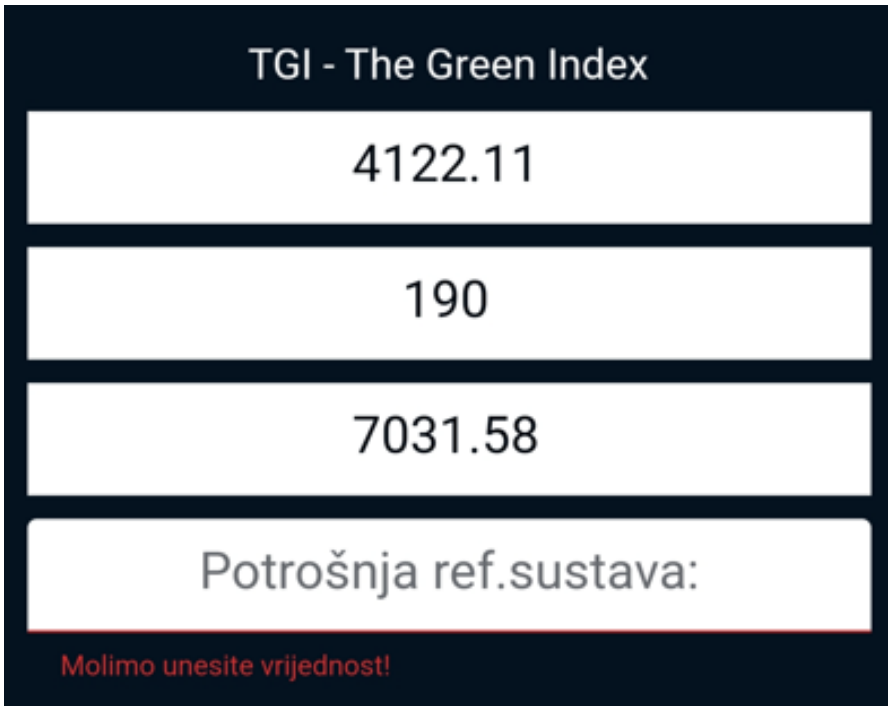
Sl. 5.6. *Okomiti kliznici*

5.2.4. Treći zaslon „*Izračun*“

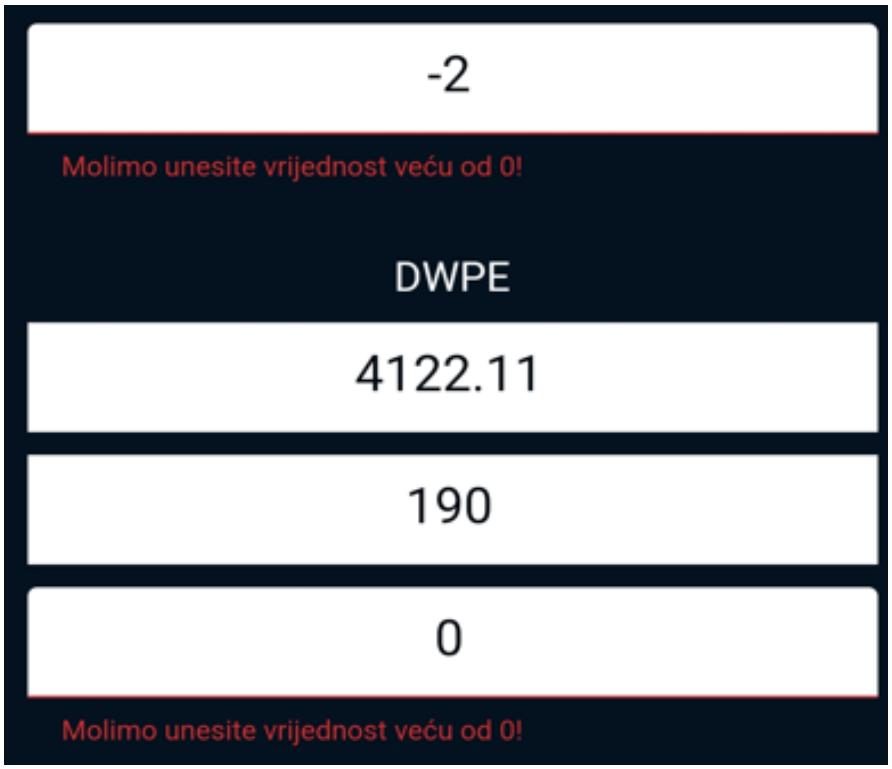
Kao što je već spomenuto u poglavlju 5.1, pritiskom na gumb *Idi na izračun* ili pritiskom na dio navigacijske trake s nazivom *Izračun* odlazi se na sljedeći zaslon na kojem su prikazana polja za unos parametara svake pojedine metrike. Na trećem zaslonu unose se parametri za svaku metriku zasebno. Parametri su vidno odvojeni, a ako kojim slučajem makar ijedan parametar nije unesen, onda korisniku nije dozvoljeno da nastavi dalje na zaslon *Rezultat*. Isto tako ako je slučajno unesen bilo koji parametar u vrijednosti 0 ili manje, onda je korisniku isto tako zabranjen odlazak na zaslon „Rezultati“ što se može vidjeti na slici 5.7. Nakon što su uneseni svi parametri i nakon što su vrijednosti parametara veće od nula, oni se zapisuju u bazu podataka. Na kraju zaslona ponovno se nalazi gumb koji korisnika šalje na stranicu rezultati.

Sl. 5.7. *Prikaz dijela zaslona za unos parametara*

Sl. 5.8. *Prikaz ostatka zaslona za unos parametara s gumbom za računanje*



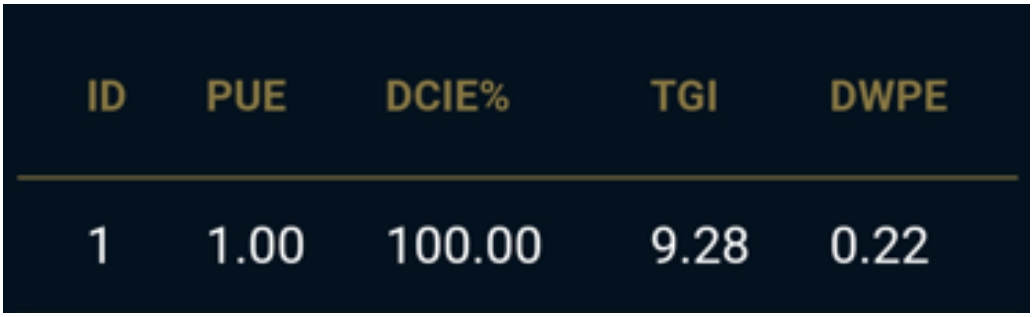
Sl. 5.9. *Prikaz poruke upozorenja za neunošenje vrijednosti*



Sl. 5.10. *Prikaz naredne poruke ukoliko je unesena vrijednost manja od nule*

5.2.5. Zaslon *„Rezultati“*

Posljednji zaslon aplikacije prikazuje rezultate izračuna za svaku pojedinu metriku. Možete naći metrike poredane onim redoslijedom kojim su opisane u drugom poglavlju ovog rada. U ovom zaslonu podaci se čitaju iz baze podataka. Posljednji podatak u nizu koji se čita iz baze podataka je upravo onaj koji je zadnji zapisan te se taj podatak može prepoznati po tome što je označen posebnom bojom. Prema slici 5.11. podatci u tablici sastoje se od pet stupaca, prvim se označava broj mjerenja, drugim PUE metrika, trećim DCE, četvrtim TGI metrika, a petim stupcem se označava DWPE metrika. Kao i na svakom prethodnom zaslonu u gornjem lijevom kutu nalazi se gumb za navigacijski izbornik. Posljednja tri rezultata su testirana na već prije poznatim parametrima i pomoću njih je ustanovljen ispravan rad aplikacije.

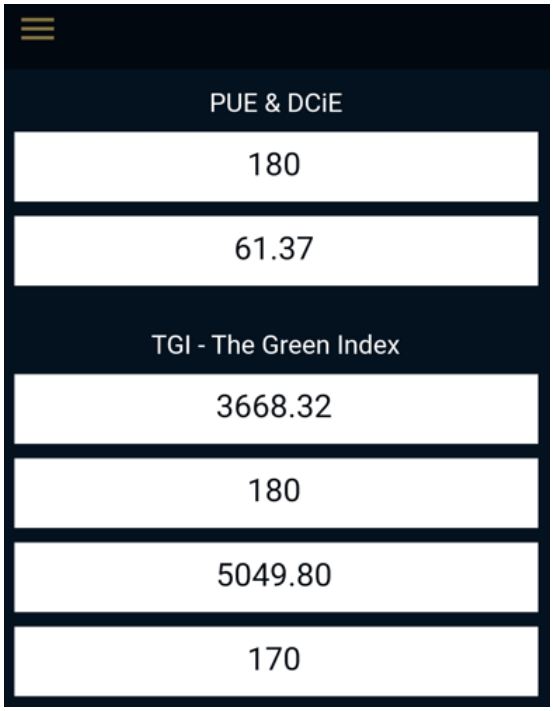


Sl. 5.11. *Prikaz dijela zaslona s nazivima stupaca*

Sl. 5.12. *Prikaz rezultata prethodnih izračuna.*

5.3. Rezultati ispitivanja aplikacije

U ovom poglavlju ukratko će se opisati ispitivanje rada mobilne Android aplikacije. Prvo će se prikazati unos svih parametara kao na slikama 5.13. i 5.14.

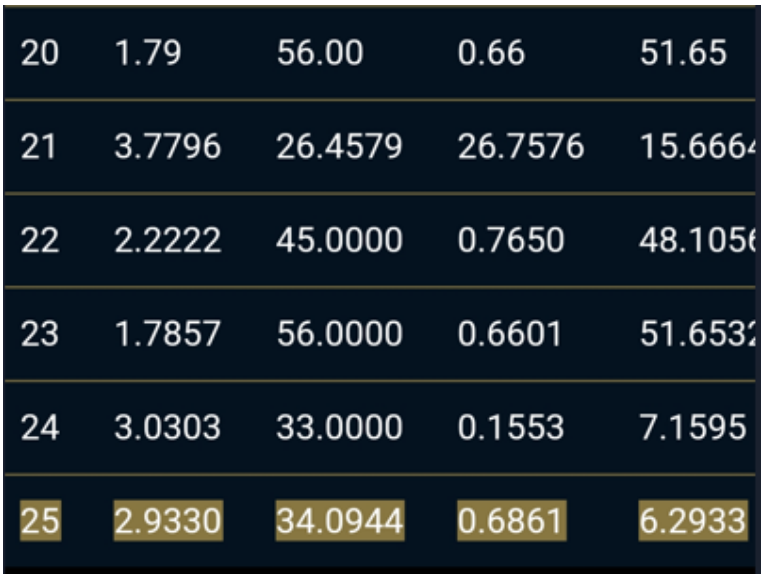


Sl. 5.13. *Prikaz unosa parametara za metrike PUE, DCE i TGI.*

Sl. 5.14. *Prikaz unosa parametara DWPE metrike*

Ukoliko se dogodi da nije unesen makar jedan parametar, onda se korisniku prikaže odgovarajuća poruka kojom mu se daje do znanja kako je potrebno ispraviti grešku nastalu prilikom unosa parametara. Slična poruka će se prikazati ispod polja u kojemu je unesena vrijednost manja od nule kako bi se korisnika uputilo na ispravljanje pogrešne vrijednosti. Primjer navedenog može se naći na slici 5.15., dok se rezultati ovog izračuna mogu naći na slici 5.16.

Sl. 5.14. *Prikaz ispisivanja odgovarajuće poruke nakon krivog unosa parametara*



Sl. 5.14. *Prikaz rezultata izračuna korištenog za testni primjer*

6. ZAKLJUČAK

U ovom završnom radu razvijena je mobilna Android aplikacija za analizu energetske učinkovitosti velikih računalnih sustava pomoću metrika PUE, DCE, TGI i DWPE te izračuna metrika koje opisuju koliko je neki sustav učinkovit. Cilj aplikacije je korisniku omogućiti unos parametara svake pojedine metrike kako bi se one mogle izračunati, a njima se iskazuju pokazatelji učinkovitosti svakog pojedinog sustava. Ukoliko bi se pokazalo da je rezultat izračuna metrike loš što bi značilo da se u podatkovnom središtu mogu napraviti neke od promjena spomenutih u poglavlju dva ovog rada. Nakon odrađenih izmjena, rezultati metrika mogu se popraviti, odnosno računalni sustav visokih performansi može biti energetski učinkovitiji.

Mobilna Android aplikacija napravljena je u razvojnom okruženju Visual Studio Code. Ostvarena je u programskom jeziku Dart koristeći se okvirom Flutter. Baza podataka ostvarena je na platformi Firebase. Ispitni slučajevi pokazali su da aplikacija radi kako je predviđeno, a pokazano je i ostvarivanje većine funkcionalnosti aplikacije s tim da nisu sve funkcionalnosti ostvarene, ali bi se mogle ostvariti u budućnosti nadogradnjom aplikacije. U ostvarene funkcionalnosti spada mogućnost pregleda metrika PUE, DCE, TGI te DWPE, unos parametara potrebnih za izračune ranije navedenih metrika, pohrana rezultata te pregled svih prethodnih rezultata koji su u prošlosti obavljeni iz baze podataka. Postoji mogućnost da se u budućnosti doda zaslon na kojem će korisnik moći vidjeti prijedloge za smanjenje potrošnje energije i povećanja energetske učinkovitosti u podatkovnom središtu. Također, može se implementirati prikaz rang ljestvice koja se sastoji od 20 najbolje rangiranih velikih računalnih sustava čiji su izračuni obavljeni u aplikaciji.

LITERATURA

[1] USGS, https://www.usgs.gov/core-science-systems/sas/arc/about/what-high-performance-computing, datum zadnjeg pristupa 13.07.2021.

[2] Techopedia, https://www.techopedia.com/definition/4595/high-performance-computing-hpc, datum zadnjeg pristupa 13.07.2021.

[3] NetApp, https://www.netapp.com/data-storage/high-performance-computing/what-is-hpc/, datum zadnjeg pristupa 13.07.2021.

[4] Network Intervju, https://networkinterview.com/what-is-green-computing/, datum zadnjeg pristupa 13.07.2021.

[5] IEA, https://www.iea.org/reports/key-world-energy-statistics-2020/final-consumption, datum zadnjeg pristupa 13.07.2021.

[6] Investopedia, https://www.investopedia.com/terms/m/mooreslaw.asp, datum zadnjeg pristupa 13.07.2021.

[7] ResearchGate https://www.researchgate.net/publication/275653947_On_Global_Electricity_Usage_of_Communication_Technology_Trends_to_2030/figures?lo=1, datum zadnjeg pristupa 13.07.2021.

[8] Energy Innovation, https://energyinnovation.org/2020/03/17/how-much-energy-do-data-centers-really-use/, datum zadnjeg pristupa 14.07.2021.

[9] Energy Innovation, https://energyinnovation.org/2020/03/17/how-much-energy-do-data-centers-really-use/, datum zadnjeg pristupa 14.07.2021.

[10] Google, https://www.google.com/about/datacenters/innovations/, datum zadnjeg pristupa 14.07.2021.

[11] Mission Critical Magazine, https://www.missioncriticalmagazine.com/ext/resources/MC/Home/Files/PDFs/TGG_Data_Center_Power_Efficiency_Metrics_PUE_and_DCiE.pdf, datum zadnjeg pristupa 14.07.2021.

[12] Energy Star, https://www.energystar.gov/products/16_more_ways_cut_energy_waste_data_center, datum zadnjeg pristupa 14.07.2021.

[13] W. Feng, B. Subramaniam, The Green Index(TGI): A Metric for Evaluating Energy Efficiency in HPC Systems, 2012 IEEE 26th International Parallel and Distributed ProcessingSymposium Workshops & PhD Forum (IPDPSW), , str. 1007-1013,Virginia, SAD.

[14] T. Wilde, A. Auweter, M.K. Patterson, H. Shoukourian, H. Huber, A.Bode, D. Labrenz, C. Cavazzoni, DWPE, A New Data Center Energy-Efficiency Metric Bridging theGap Between Infrastructure an Workload, 2014 International Conference on High Performance Computing & Simulation (HPCS), , str. 893-901, Bologna, Italija.

[15] Visual Studio Code, https://code.visualstudio.com, datum zadnjeg pristupa 12.09.2022.

[16] Dart, https://dart.dev, datum zadnjeg pristupa 12.09.2022.

[17] Flutter, https://flutter.dev, datum zadnjeg pristupa 12.09.2022.

[18] Firebase, https://firebase.google.com, datum zadnjeg pristupa 12.09.2022

[19] S. Tiwari, S. Shah, V. Kulkarni, P. Hegde Patil, A Review on Green Computing Implementation Using Efficient Techniques, 2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N), str. ?-?, Great Noida, Indija.

[20] M. Asanović, N. Gospić, Implementation of the Green IT recommendations in municipalities, 2016 24th Telecommunications Forum (TELFOR), str. Xx-xx, Beograd, Srbija.

[21] Energy Innovation https://energyinnovation.org/2020/03/17/how-much-energy-do-data-centers-really-use/, datum zadnjeg pristupa 12.09.2022.

[22] IEA, https://www.iea.org/data-and-statistics, datum zadnjeg pristupa, 12.09.2022.

POPIS SLIKA

SAŽETAK

Prilikom izrade ovog završnog rada programski je ostvarena mobilna Android aplikacija za računanje energetske učinkovitosti računalnih sustava visokih performansi U teorijskom dijelu rada opisana su načela zelenog računarstva te je objašnjena uloga velikih računalnih sustava u podatkovnim središtima. Također, pojašnjene su metrike PUE, DCE, TGI te DWPE koje se koriste kako bi se odredilo koliko je učinkovit neki računalni sustav. U aplikaciji je omogućena navigacija između različitih zaslona aplikacije, sažeti opis metrika, kao i njihovi izrazi, unos parametara za svaku pojedinu metriku, odnosno pregled rezultata spremljenih u bazu podataka uključujući i rezultate prethodnih izračuna. Programsko rješenje ostvareno je u razvojnoj okolini Visual Studio Code, korištenjem Dart programskog jezika i baze podataka Firebase. Rezultati ispitivanja pokazuju da….

Ključne riječi: energetska učinkovitost, metrike, mobilna Android aplikacija, podatkovna središta, računalni sustavi velikih performansi.

ABSTRACT

Title: The mobile Android application for analyzing energy efficiency of high performance computers

During the creation of this final paper, a mobile Android application was programmed to calculate the energy efficiency of high-performance computing systems. The principles of green computing are described in the theoretical part of the paper as well as the role of large computer systems in data centers. What is also described are the PUE, DCE, TGI and DWPE metrics which are used to determine how efficient a computer system. The application enables navigation between different application screens, a concise description of the metrics, as well as their expressions, input of parameters for each individual metric and. an overview of the results saved in the database, including the results of previous calculations. The software solution was created in the Visual Studio Code development environment, using the Dart programming language and the Firebase database. The test results show that….

Keywords: data centers, energy efficiency, high performance computing, metrics, mobile Android application

ŽIVOTOPIS

Josip Oršolić rođen je 30. svibnja 1999. godine u Slavonskom Brodu. Pohađao je osnovnu školu „Stjepana Radića“ u mjestu Domaljevac, Bosna i Hercegovina. Upisuje franjevačku klasičnu gimnaziju u Visokom 2014. godine, a istu završava 2018. godine kad upisuje i preddiplomski studij Računarstvo na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku gdje i danas studira.

PRILOZI

Prilog 1. Završni rad „Mobilna Android aplikacija za analizu energetske učinkovitosti velikih računalnih sustava“ u *.docx* formatu

Prilog 2. Završni rad „Mobilna Android aplikacija za analizu energetske učinkovitosti velikih računalnih sustava“ u *.pdf* formatu

Prilog 3. Programski kod mobilne aplikacije