Mobilna Android aplikacija za potporu pri izboru usluga zasnovanom na višekriterijskom dodjeljivanju

Aščić, Barbara

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

MOBILNA ANDROID APLIKACIJA ZA POTPORU PRI IZBORU USLUGA ZASNOVANOM NA VIŠEKRITERIJSKOM DODJELJIVANJU

Završni rad

Barbara Aščić

Osijek, 2020.

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

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

Osijek, 14.09.2020.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime studenta:	Barbara Aščić
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4026, 24.09.2019.
OIB studenta:	36882794194
Mentor:	Prof.dr.sc. Goran Martinović
Sumentor:	Dr. sc. Krešimir Romić
Sumentor iz tvrtke:	
Naslov završnog rada:	Mobilna Android aplikacija za potporu pri izboru usluga zasnovanom na višekriterijskom dodjeljivanju
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Predložena ocjena završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	14.09.2020.
Datum potvrde ocjene Odbora:	23.09.2020.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

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

IZJAVA O ORIGINALNOSTI RADA

Osijek, 24.09.2020.

Ime i prezime studenta:	Barbara Aščić
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4026, 24.09.2019.
Turnitin podudaranje [%]:	11

Ovom izjavom izjavljujem da je rad pod nazivom: **Mobilna Android aplikacija za potporu pri izboru usluga zasnovanom na višekriterijskom dodjeljivanju**

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

i sumentora Dr. sc. Krešimir Romić

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

Tijekom cijelog života čovjek sudjeluje u raznim procesima odlučivanja. Odlučivanje predstavlja odabir najpovoljnijeg rješenja između više alternativa. Kako bi se odabralo najpovoljnije rješenje moraju postojati uvjeti koje to rješenje zadovoljava. Rješenje ne mora biti optimalno najbolje ali će u odnosu na druge alternative pružiti najbolji odgovor. Kriterije postavlja donositelj odluke prema svojoj subjektivnoj odluci.

U odlučivanju se može koristiti više metoda za odlučivanje, a one se biraju prema sposobnostima donositelja. Postoje jednokriterijski i višekriterijski modeli odlučivanja. Jednokriterijski modeli sastoje se samo od jedne funkcije, dok se višekriterijski modeli sastoje od dvije ili više funkcija.

Cilj ovog završnog rada je izraditi mobilnu android aplikaciju koja će implementirati model višekriterijskog odlučivanja, te pomoću modela odabrati najbolju tarifnu uslugu koju pruža neki operater.

U drugom poglavlju opisani su načini pružanja usluga, prikazan je rad u softveru za višekriterijsko dodjeljivanje - Pico Blaze Pro. U trećem poglavlju objašnjena je AHP (engl. *Analytic Hierarchy Process*) metoda. U četvrtom poglavlju prikazano je programsko rješenje mobilne aplikacije. Peto poglavlje opisuje korištenje mobilne aplikacije te testiranje i analizu rezultata aplikacije. U šestom poglavlju dan je zaključak s potrebnom literaturom, opisanim tablicama i slikama koje su korištene u izradi ovog rada.

1.1.Zadatak završnog rada

U radu je potrebno objasniti implementiranje softvera za višekriterijsko odlučivanje točnije odabrani model. Napraviti android aplikaciju koja će odlučivati o odabiru najprihvatljivije tarife koja je dana od strane određenog operatera. Odabrano rješenje mora biti unutar kriterija koje je zadao podnositelj kriterija. Mobilnu aplikacija treba testirati na odabranom skupu ulaznih podataka, te izlazna rješenje analizirati.

2. NAČINI IZBORA USLUGA

U ovom poglavlju predstavljeni su načini na koje se može izabrati usluga. Dani su primjeri pojedinih rješenja i njihova usporedba. Objašnjene su prednosti i mane određenih modela.

2.1. Višekriterijsko odlučivanje

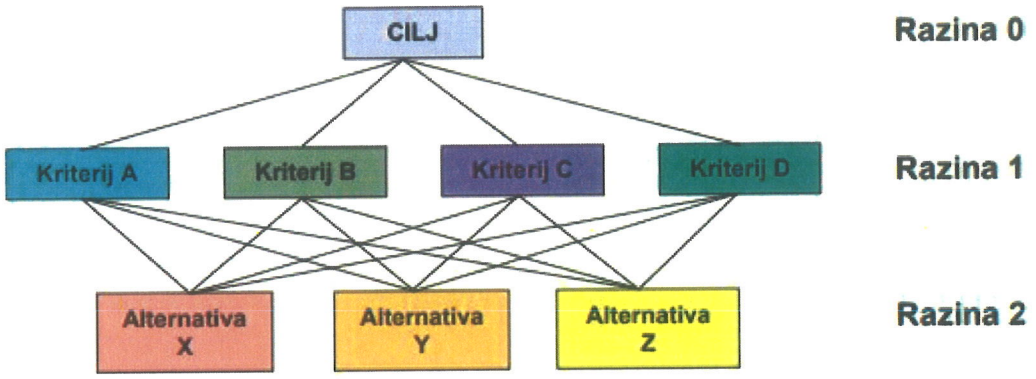
Višekriterijsko odlučivanje vrijedan je alat koji možemo primijeniti u mnogim složenim odlukama. Upotrebljava se u gotovo svim aspektima društvene djelatnosti. Najprimjereniji je za rješavanje problema u izboru između različitih karakteristika. Prema [1], višekriterijsko dodjeljivanje ili MCDM (engl. *Multi-Criteria Descision-making*) ima mogućnost poboljšanja svih područja za odlučivanja u inženjerstvu, od dizajna do proizvodnje, ali posebice u visokim tehnologijama gdje se konkurentni proizvodi uspoređuju.

2.1.1. MCDM metode

Prema [2] problem višekriterijskog odlučivanja sastoji se od 5 komponenti:

- Cilj
- Donositelj odluke ili skupina donositelja odluke s izborom
- Alternative ili opcije
- Kriterij za ocjenjivanje
- Ishodi ili posljedice povezane s kombinacijama alternativa

Nakon ove podjele, ciljeve, kriterije i opcije (alternative) može se prikazati kao skup individualnih komponenti povezanih s istaknutim vezama.



Slika 2.1. Struktura AHP metode

MCDM ne može garantirati izbor najboljeg rješenja za određeni problem, nije objektivna (kriterije postavlja sam podnositelj zahtjeva) te zbog toga nije primjerena za donošenje teških i odgovornih presuda.

2.2.Postojeće metode

Većina DM softvera [3] temelji se na odlučivanju s više kriterija. Višekriterijsko nasljeđivanje uključuje ocjenjivanje i kombiniranje karakteristika alternativna na dva ili više kriterija kako bi se odredili prioriteti ili kako bi se biralo između više ponuđenih alternativa.

Neke od najpoznatijih vrsta MCDM metoda su:

- Analitički hijerarhijski proces (engl. *Analytic Hierarchy Process*)- koja će se koristiti u ovom radu
- Analitički mrežni proces (engl. *Analytic network process*)
- Višekriterijska metoda ELECTRE (engl. *Elimination and Choice Expressing Reality*)
- Višekriterijska metoda AIRM (engl. *Aggregated Indices Randomization Methos*)

Postoje značajne razlike između svake od ovih metoda. Razlikuju se u broju kriterija, omjeru preferencija, vrstama usporedba, itd.

2.2.1. Analitički hijerarhijski proces -AHP

Prema [4] AHP ili proces analitičke hijerarhije je strukturna tehnika organiziranja i analiziranja kompleksnih odluka, utemeljena na matematici i psihologiji. AHP model ima hijerarhijski model u kojem se cilj nalazi na vrhu, kriteriji u sredini te alternative na dnu modela. AHP metoda je jedna od najpoznatijih i najkorištenijih metoda za višekriterijsko odlučivanje. Stoga je ona izabrana za rješenje ovog završnog rada te je detaljnije objašnjena u poglavlju 3.3.

2.2.2. Analitički mrežni proces -ANP

ANP [5] je proširenu oblik AHP-a koji razmatra informacije o svakoj opciji ili alternativi, utjecaj svakog elementa na ostale kriterije i neovisnost između i unutar razina. Za razliku od AHP-a ne predstavlja linearnu hijerarhiju već modelira utjecaje između elemenata mreže. Rezultira stabilnijim rješenjima i preciznijim određivanjem prioriteta, ali zahtjeva više vremena i koncentracije. U Analitičkom mrežnom procesu razlikuju se 3 razine modela: hijerarhija kombinirana s mrežom, mreža elemenata i BOCR (engl. *Benefits*, *Opportunities*, *Costs* i *Risks*) odnosno mreža elemenata s kontrolnim i strateškim kriterijima.

2.2.3. Višekriterijska metoda ELECTRE

Metoda ELECTRE (engl. *Elimination and Choice Expressing Reality*) [6] najprije se koristila za odabir najboljih akcija iz određenog niza radnji, ali ubrzo nakon toga primijenjena je na glavna 3 problema: izbor, poredak i razvrstavanje. Neke od inačica ELECTRE su ELECTRA I, II, II, IV, IS, TRI te se koriste u područjima poslovanja, razvoja, dizajna i malih hidroelektrana. Obično se smatra „pretjeranom metodom" za donošenje odluka. Koristi se za odbacivanje nekih drugih problema koji su neprihvatljivi te za to koristi dva različita skupa parametara: koeficijent važnosti i veto pragova.

2.2.4. Višekriterijska metoda AIRM

Glavna prednost AIRM metode (engl. *Aggregated indices randomization method*) [7] u odnosu na druge MCDM metode je njena sposobnost da se nosi s nekvalitetnim ulaznim varijablama. Za ulazne varijable mogu se postaviti nenumerički, netočni i nepotpuni podaci. AIRM metoda ima različite primjene:

- podrška u menadžerskim odlukama na visokoj razni, koje su okarakterizirane nepreciznim informacijama

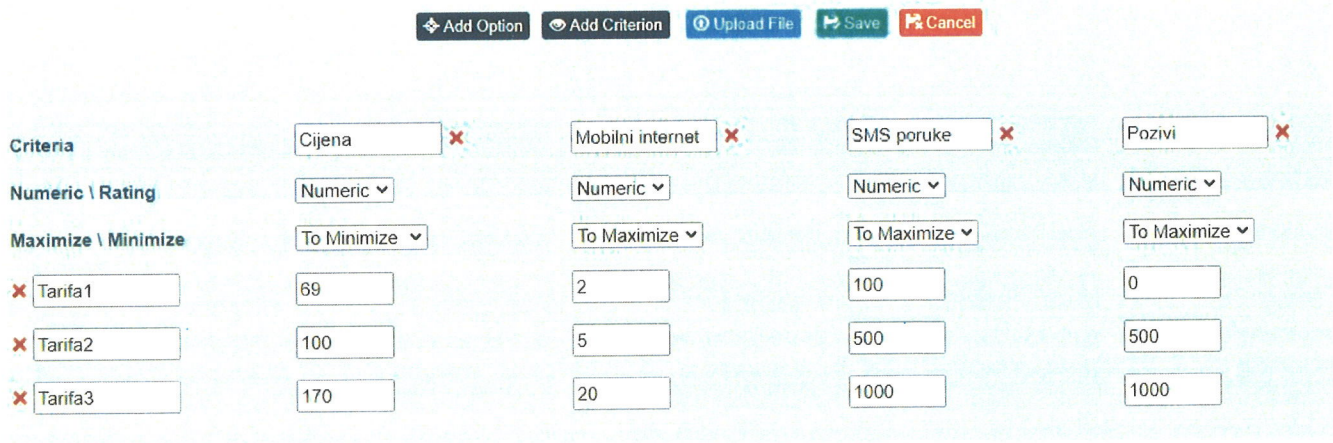
- procjena složenih tehničkih sustava zbog nesigurnih i učinkovitih performansi
- sinteza kolektivnog stručnog mišljenja u nedostatku podataka o kvalifikaciji stručnjaka
- višekriterijska procjena dinamičkih alternativna unutar ekonomskog i financijskog odnosa

2.3. Primjer postojećeg rješenja

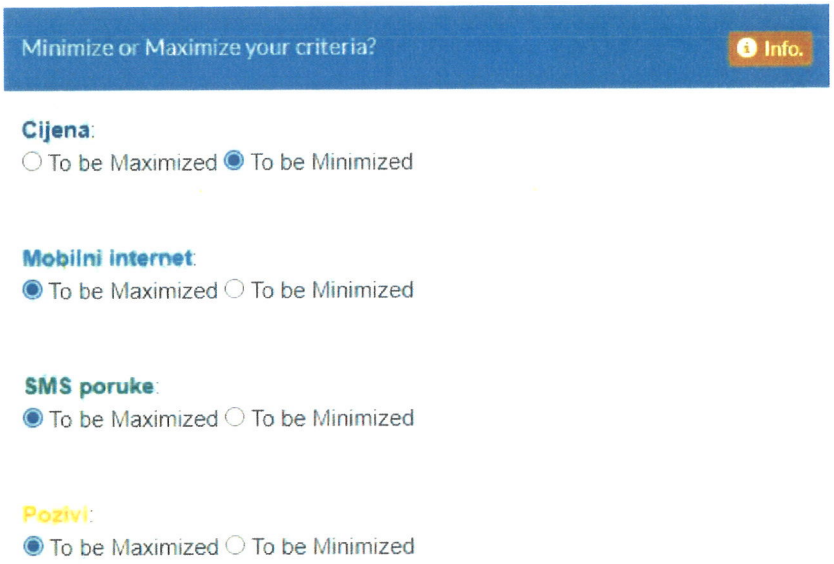
Najčešći softver koji se koristi za rješavanje problema višekriterijskog dodjeljivanja je Smart Picker Pro [8]. Napravljen je 2011. godine s dva cilja: pomoći ljudima u donošenju odluka i kako bi poboljšao proces donošenja odluka. Smart Picker koristi se u različitim aplikacijama gdje smanjuje vrijeme i povećava efektivnost rada dajući važnost svim kriterijima određenog problema i definirajući specifične parametre. Softver je dostupan svima te pruža jednostavne alate za odlučivanje.

2.3.1. Primjer rješenja u Smart Picker Pro softveru

Za primjer problema korišten je odabir najpogodnije tarife za neodređenog operatera. Imena alternativa su: Tarifa1, Tarifa2 i Tarifa3. Kriteriji po kojima se određuje najpogodnija tarifa su: cijena, količina mobilnog interneta izraženog u gigabajtima, količina SMS poruka te broj minuta za pozive. Određuje se treba li kriterij biti umanjen ili uvećan. Cijena je postavljena na najmanju vrijednost, dok su ostali kriteriji postavljeni na najveću vrijednost. Parametri su postavljeni subjektivno.

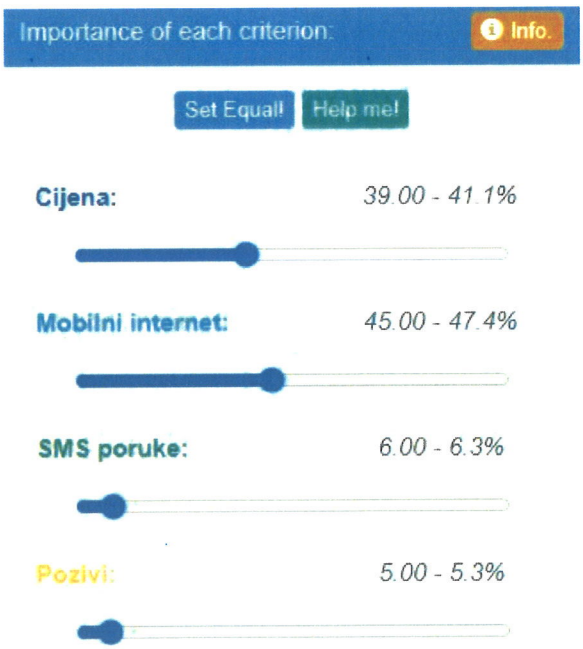


Slika 2.1. Parametri



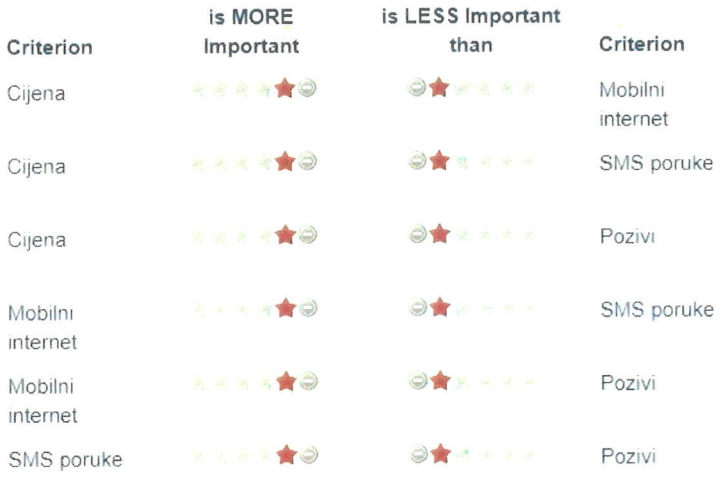
Slika 2.2. Ručno postavljanje parametara

Nadalje, dodane su težine kriterija.



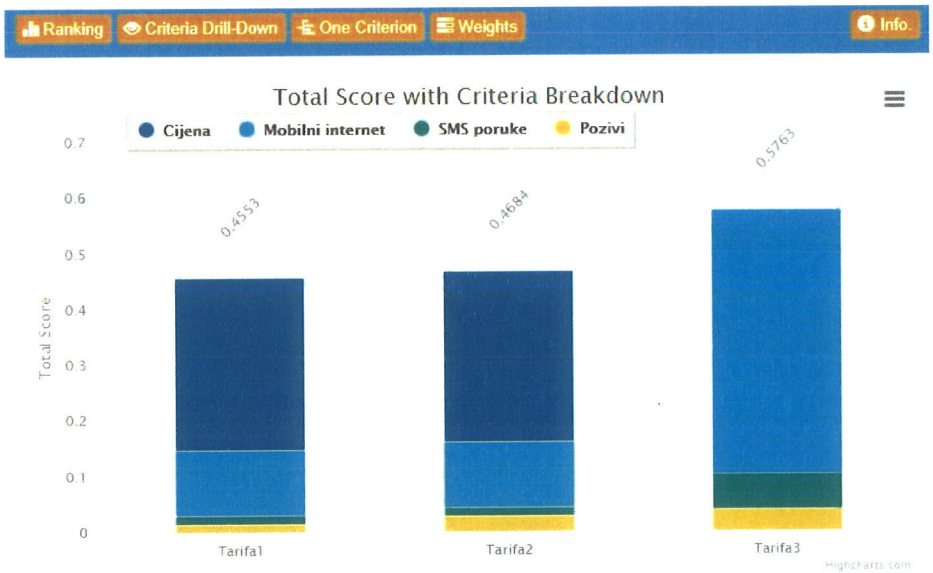
Slika 2.3. Težine kriterija

Umjesto davanja izravnih vrijednosti za sve težine kriterija, može se usporediti svaki par kriterija i odrediti koji je manje ili više važan u usporedbi s drugim parom. Na temelju danih usporedba sustav će sam izračunati vrijednosti za težine kriterija. Na slici 3.4. prikazano je 6 uspoređenih parova.



Slika 2.4. usporedba parova

Na grafu 3.1. prikazano je rješenje problema odabira najpogodnije tarife. Na grafu je također vidljivo koji kriterij je bio od najveće važnosti a koji od najmanje. Tarifa3 je izabrana kao optimalno rješenje u usporedbi s ostale dvije tarife.



Slika 2.5. rješenje odluke

2.3.2. AHP aplikacija

Aplikaciju AHP moguće je preuzeti na *Google* trgovini. Aplikacija prije korištenja zahtjeva prijavu putem g-maila. Kada korisnik uđe u aplikaciju postavlja broj alternativa i broj kriterija. Nakon toga od korisnika se traži unos imena za svaku alternativu, te tako i za svaki kriterij. Zatim korisnik uspoređuje svaki kriterij između svake od alternative i daju mu numeričku vrijednost. Aplikacija traži od korisnika 12 usporedbi i zatim izračunava najpogodniju tarifu. Rezultat je prikazan u postocima. Aplikacija koja je rađena u ovom završnom radu ima nekoliko prednosti u odnosu na AHP aplikaciju. Prva prednost je što ne zahtjeva od korisnika unos alternativa i kriterija jer ima već ugrađene vrijednosti. Cijena, količina internetskog prometa, SMS-a i poziva uvijek je fiksna za pojedinu tarifu, te su zbog toga i odnosi između kriterija uvijek jednaki. Druga prednost je grafički prikaz rješenja. Treća prednost je što korisnik ima uvid u informacije o kojima odlučuje. To je objašnjeno u daljnjim poglavljima.

Slika 2.6. AHP aplikacija

Slika 2.7. AHP aplikacija

3. PRIJEDLOG MODELA ZA IZBOR USLUGA

Kako bi se izabrao najbolji model višekriterijskog odlučivanja, prvo se trebaju analizirati zahtjevi korisnika. U poglavlju 3.1 i 3.2. definirane su usluge koje će aplikacija pružati te je obrazloženo zašto se koristi AHP metoda odlučivanja. Za najpreciznije odluke između dva para koristi se Saaty-jeva skala. U poglavlju 3.3.2 objašnjeno je korištenje Saaty-jeve skale, njegova pretvorba u matrični prikaz, izračun svojstvenog vektora

3.1.Zahtjev korisnika

Korisnik aplikacije želi prema svojim zahtjevima izabrati najbolju tarifu. Postoje 4 tarife koje korisnik može izabrati. Svaka tarifa razlikuje se u cijeni, količini SMS poruka, količini poziva i količini Internetskog prometa izraženog u gigabajtima (GB). Korisnik treba odrediti važnost pojedinog kriterija te će na temelju toga dobiti najpogodniju tarifu. Tarife koje korisnik može izabrati su prikazane u tablici 3.1. Tarife su preuzete od operatera A1 kako bi rezultat višekriterijskog odlučivanja bio što realniji.

Tablica 3.1. Tarife

naziv Tarife	cijena (kn)	Internet (GB)	SMS	pozivi
Strimalica	139	10	1500	1500
Surferica	99	7	1000	1000
Sheralica	79	4	500	500
Spikalica	49	1	300	300

Ukoliko korisnik želi tarifu sa što manjom cijenom, pretpostavka je da će aplikacija kao rezultat vratiti tarifu Spikalicu. Isto tako ako korisnik želi tarifu sa što većim brojem internetskog prometa, pretpostavka je da će aplikacija kao rezultat vratiti tarifu Strimalicu.

Odabir najbolje tarife je složen proces zbog utjecaja više kriterija na konačno rješenje, utjecaja velikog broja sudionika na odluku te upitne pouzdanosti u donositeljevu odluku. Zbog ovih karakteristika AHP metoda odlučivanja je izabrana za korištenje kao programsko rješenje.

3.2.Usluge

Aplikacija pruža korisniku unos važnosti pojedinog kriterija putem klizača (eng. *slider*). Ako su mu svi kriteriji jednako važni korisnik će koristiti funkciju set_equal koja postavlja sve kriterije na istu vrijednost. Na temelju tih parametara računa se optimalna tarifa za korisnika. Prikaz

rješenja je grafički, te su prikazane sve tarife sa svojim postocima. Ukoliko mu ta tarifa ne odgovara korisnik može izabrati i neku drugu tarifu. Ako se korisnikovi zahtjevi promjene, budući da se koristi AHP metoda, na jednostavna način se mogu napraviti izmjene.

3.3. AHP- metoda odlučivanja

Kao što je već spomenuto u poglavlju 2.2.1 AHP je utemeljena na matematici i psihologiji te je jedna od vrsta tehnika koje se bave organiziranjem i analiziranjem kompleksnih odluka.

Situacije odlučivanja u kojima AHP može biti korištena uključuju:

- izbor
- rangiranje
- određivanje prioriteta
- alokacija resursa

AHP je osobito pogodan za kompleksne odluke koje uključuju uspoređivanje elementa odluke koje je teško kvantificirati [9]. AHP i njegove novije verzije ANP-a (engl. *Analytic Network Process*) koje je razvio Thomas Saaty koriste se za širok spektar višekriterijskog odlučivanja. Najveća vrijednost AHP-a proizlazi iz njegove mogućnosti da kombinira, sintetizira, kvantitativno i kvalitativno razmatra u cjelokupnom ocjenjivanju alternativa. AHP ima raznovrsnu primjenu u raznim područjima kao što su ekonomija, poduzetništvo, agrokultura, elektrotehnika, politika, itd. U razvoju softvera može se koristiti za višekriterijsko odlučivanje u različitim fazama, od razvoja zahtjeva do dizajna za pregled, testiranja, ocjenjivanja, održavanja, razgradnje i uključivanja situacija s više objekata.

3.3.1. Faze modeliranja

Postoje 4 faze modeliranja AHP metodom [10]. Rangiraju se od nulte do treće faze. Nulta faza predstavlja strukturiranje problema. U prvoj fazi određuju se najznačajniji kriteriji, u drugoj fazi određuju se najznačajnije alternative. Konačno rješenje, odnosno ispunjenje cilja određuje se u trećoj fazi. Ljudskoj prirodi najlakši i najjednostavniji oblik je uspoređivanja samo dvije opcije. Zbog toga AHP metoda koristi uspoređivanje dva para. Prioritete između dvije alternative izražavamo opisnim vrijednostima kao jako, slabo, umjereno, itd.

3.3.2. Saaty-jeva skala

Kako bi se napravila najpreciznija odluka između dva para čovjek koristi Saaty-jevu skalu koja je podijeljena na 9 odjeljaka.

Skala	Opis inteziteta važnosti
1	jednako važno
3	umjereno važno
5	važnije
7	puno važnije
9	ekstremno važnije
2, 4, 6, 8	međuvrijednosti

Slika 3.1. Saaty-jeva skala

Spikalica	9	Strimalica	1
Spikalica	5	Suferica	1
Spikalica	3	Sheralica	1
Sheralica	6	Strimalica	1
Sheralica	2	Suferica	1
Surferica	4	Strimalica	1

Slika 3.2. Usporedbe alternativa Saaty-jevom skalom u odnosu na cijenu

U ovom primjeru postoje 4 alternative (4 tarife) koje se uspoređuju jedna s drugom s obzirom na kriterij cijena. Što je cijena manja to je intenzitet važnosti veći. Cijena tarife Spikalice ekstremno je važnija od cijene Strimalice, važnija je od cijene Surferice i umjereno važnija od cijene Sheralice. Na isti način su uspoređene ostale tarife. Slijedeći korak je prijenos težina u matricu.

Tablica 3.2. Težina alternativa za kriterij cijena

CIJENA	STRIMALICA	SURFERICA	SHERALICA	SPIKALICA
STRIMALICA	1	1/4	1/6	1/9
SURFERICA	4	1	1/2	1/5
SHERALICA	6	2	1	1/3
SPIKALICA	9	5	3	1

Na isti način radi se usporedba parova s obzirom na količinu Internetskog prometa, SMS-ova i poziva. Kada se dobije matrični prikaz za sve kriterije, koji se još naziva usporedna matrica izračunava se svojstveni vektor. Svojstveni vektor je normalizacija Eigenove vektor matrice. Prvi korak izračuna svojstvenog vektora je zbrajanje svakog stupa matrice kao što je prikazano u Tablici 3.3.

Tablica 3.3. Zbroj stupaca

CIJENA	STRIMALICA	SURFERICA	SHERALICA	SPIKALICA
STRIMALICA	1	1/4	1/6	1/9
SURFERICA	4	1	1/2	1/5
SHERALICA	6	2	1	1/3
SPIKALICA	9	5	3	1
ZBROJ	20	33/4	14/3	74/45

Zatim se svaki element matrice dijeli sa zbrojem stupca u kojem se nalazi, ovo se naziva normalizacija relativne težine koja je prikazana u tablici 3.4. Zbroj svakog stupca mora bit 1.

Tablica 3.4. Dijeljenje elemenata sa zbrojem stupca

CIJENA	STRIMALICA	SURFERICA	SHERALICA	SPIKALICA
STRIMALICA	1/20	1/33	1/28	5/74
SURFERICA	1/5	4/33	3/28	9/74
SHERALICA	3/10	8/33	3/14	15/74
SPIKALICA	9/20	20/33	9/14	45/74
ZBROJ:	1	1	1	1

Svaki redak matrice se zbroji te zatim pomnoži s ¼. Ovo se naziva normalizacija glavnog Eigenovog vektora koji se još označava kao svojstveni vektor. Postupak normalizacije vektora prikazan je u tablici 3.5.

Tablica 3.5. Normalizacija vektora

CIJENA		STRIMALICA	SURFERICA	SHERALICA	SPIKALICA	PRIORITET
STRIMALICA	1/4*	1/20 +	1/33 +	1/28 +	5/74	0,0459
SURFERICA		1/5 +	4/33 +	3/28 +	9/74	0,1375
SHERALICA		3/10 +	8/33 +	3/14 +	15/74	0,2398
SPIKALICA		9/20 +	20/33 +	9/14 +	45/74	0,5768
ZBROJ:		1	1	1	1	

Zbroj svih elemenata u prioritetu je 0.0459+0.1357+0.2399+0.5768=1. Svojstveni vektor prikazuje relativnu težinu s obzirom na kriterij za koji se uspoređuje. U ovom primjeru tarifa Strimalica ima težinu od 4.59%, tarifa Surferica 13.75%, tarifa Spikalica 57.68%. Dobiveno rješenje odgovara pretpostavki jer tarifa Spikalica ima najmanju cijenu. Istim postupkom dobije se svojstveni vektor za ostale kriterije kao što je prikazano u tablicama 3.6., 3.7. i 3.8.

Tablica 3.6. Težina alternativa za kriterij količina Internetskog prometa sa prioritetima

Internet	STRIMALICA	SURFERICA	SHERALICA	SPIKALICA	Prioritet
STRIMALICA	1	3	6	9	0,5854
SURFERICA	1/3	1	3	6	0,2588
SHERALICA	1/6	1/3	1	3	0,1080
SPIKALICA	1/9	1/6	1/3	1	0,0478

Tablica 3.7. Težina alternativa za kriterij količina SMS-ova sa prioritetima

SMS	STRIMALICA	SURFERICA	SHERALICA	SPIKALICA	Prioritet
STRIMALICA	1	3	6	8	0,5872
SURFERICA	1/3	1	3	5	0,2560
SHERALICA	1/6	1/3	1	2	0,0990
SPIKALICA	1/8	1/5	1/2	1	0,0578

Tablica 3.8. Težina alternativa za kriterij količina poziva sa prioritetima

POZIVI	STRIMALICA	SURFERICA	SHERALICA	SPIKALICA	Prioritet
STRIMALICA	1	3	6	8	0,5872
SURFERICA	1/3	1	3	5	0,2560
SHERALICA	1/6	1/3	1	2	0,0990
SPIKALICA	1/8	1/5	1/2	1	0,0578

Kada su prioriteti u odnosu na kriterij izračunati, izračunava se prioritet u odnosu na cilj. U ovom slučaju korisnik sam određuje u aplikaciji koliko mu je važan pojedini kriterij. Prioritet u odnosu na cilj izračunavaju se prema formuli prikazanoj na slici 3.3.

STRIMALICA= 0.0459*Cijena+0.5854*Internet+0.5872*SMS+0.5872*Pozivi
SURFERICA= 0.1375*Cijena+0.2577*Internet+0.2560*SMS+0.2560*Pozivi
SHERALICA= 0.2398*Cijena+0.1080*Internet+0.0990*SMS+0.0990*Pozivi
SPIKALICA= 0.5768*Cijena+0.0478*Internet+0.0578*SMS+0.0578*Pozivi

Slika 3.3. Formula za izračun prioriteta za sve tarife

Vrijednosti varijabla cijena, Internet, sms i pozivi definira korisnik u aplikaciji.

4. PROGRAMSKO RJEŠENJE

Za izradu mobilne aplikacije neophodno je znanje osnova višekriterijskog odlučivanja i implementacija AHP metode. U poglavlju 4.1. upoznaje se s programskim jezikom java koji je korišten u pisanju mobilne aplikacije. Korištenje integriranog razvojnog okruženja Android Studia te njegova primjena objašnjena je u poglavlju 4.2. U daljnjim poglavljima objašnjena je izrada aktivnosti za početni zaslon, za unos kriterija, za prikaz rezultata te prikaz tarifa.

4.1.Java

Java [12] je programski jezik opće namjene, temeljen na klasama i objektno orijentiranom načelima. Sun Microsystems 1995. godine izdaje prvu verziju programskog jezika Jave pod vodstvom Jamesa Goslinga. Java jezik morao je zadovoljiti 5 osnovnih ciljeva:

1. Biti jednostavan, objektno-orijentiran i poznat
2. Biti čvrst i siguran jezik
3. Biti lako prenosiv i neutralan o arhitekturi računala
4. Izvršavati se s visokim performansama
5. Biti dinamičan i lako čitljiv

Izvodi se na svim platformama koje podržavaju Javu bez potrebe za ponovnim prepisivanje. Java aplikacije pokreću se na Java virtualnom stroju (JVM) bez obzira na arhitekturu računala. Jezici koji su slični Javi su C i C++ zbog svoje sintakse. Za razliku od C++ koji kombinira sintaksu za strukturno, generičko i objektno orijentirano programiranje, kod Jave se sav programski kod upisuje u klase. 2019. godine Java je bila jedan od najpopularnijih jezika koji se koristio u razvoju mobilnih aplikacija te je zbog toga izabrana za korištenje u ovom radu.

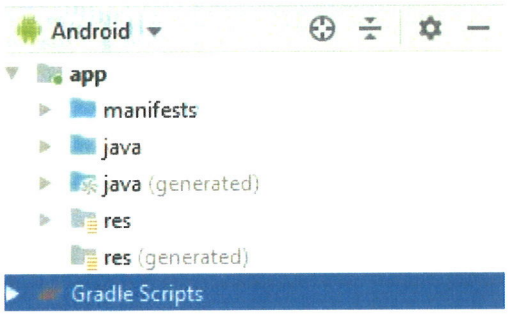
4.2.Android Studio

Android Studio je službeno integrirano razvojno okruženje za Google-ov operativni sustav Android. Android Studio u sebi sadrži razvojni paket ali se može nadopuniti dodatnim paketima. Linux, Windows i Mac OS X su operativni sustavi koji ga podržavaju. Preuzimanje Android Studia je besplatno i dostupno je svima. Android Studio nudi više značajki za poboljšanje produktivnosti i izgleda Android aplikacija, prema [11] to su:

- Podržavanje C++ i NDK
- Android virtualni uređaj ili emulator za pokretanje i uklanjanje pogrešaka u aplikaciji
- Različiti alati i okviri namijenjeni za testiranje

- Jedinstveno programsko okruženje za razvijanje svih Android uređaja
- Brzi emulator
- Integracija s GitHub-om koji pomaže u izgradnji često korištenih funkcija i uvoza jednostavnog koda

Android Studio prikazuje projektne datoteke a to su *manifest* mapa, *java* mapa, *res* mapa i *Gradle* skripta.



Slika 4.1. Projektne datoteke u Android prikazu

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'androidx.appcompat:appcompat:1.1.0'
    implementation
'androidx.constraintlayout:constraintlayout:1.1.3'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.1'
    androidTestImplementation 'androidx.test.espresso:espresso-
core:3.2.0'
    implementation 'androidx.recyclerview:recyclerview:1.2.0-
alpha01'
    implementation 'androidx.cardview:cardview:1.0.0'
   implementation'com.github.PhilJay:MPAndroidChart:v3.0.2'

}
```

Slika 4.2. *Gradle* skripta-*build.gradle*

Zadatak *Gradle* skripte je implementiranje, testiranje i pretvaranje koda u *dex.* datoteku, odnosno pokretanje aplikacije na uređaju.

4.2.1. Manifest

U *manifest* mapi nalazi se datoteka *AndroidManifest.xml* koja se u svakom projektu naziva isto a pisana je XML opisnim jezikom. Sastoji se od naziva paketa aplikacije, aktivnosti, usluga, prijemnika, poslužitelja usluga, dozvola i hardverskih i softverskih obilježja. Dozvole služe za pristup zaštićenim dijelovima sustava ili nekim drugim aplikacijama. U datoteci *manifesta* definiraju se komponente koje mogu biti stvorene u aplikaciji:

- *Activity*
- *Service*
- *Broadcast Receiver*
- *Content Provider*

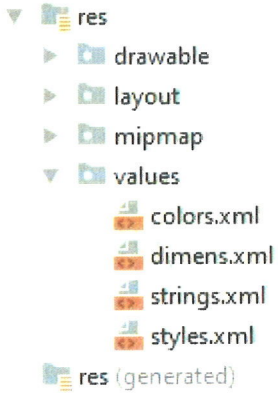
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.choosingrighttarifa">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".TariffActivity"></activity>
        <activity android:name=".CalculatingActivity"></activity>
        <activity android:name=".ResultActivity"></activity>
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Slika 4.3. Manifest datoteka

Iz Slike 4.3. vidljivo je da aplikacija sadrži 4 aktivnosti (engl. *activity)* : MainActivity, ResultActivity, CalculatingActivity i TariffActivity. Kada se pokrene aplikacija, prvi aktivnost koja se prikazuje je Main jer je ona navedena u *intent-filteru.*

4.2.2. Resursi

Resursi su dodatne datoteke koje aplikacijski kod koristi, odnosno sve slike, tekstovi, audio zapisi ikone i drugi podaci. Resursi se definiraju u mapi *res* unutar projekta. Kao što postoje različiti tipovi resursa, tako postoje i različite mape u koje se oni spremaju ovisno o vrsti. To su slike (eng. *drawable)*, izgled (engl. *layout*), male ikonice (eng. *mipmap)* i mapa *values* u kojoj su spremljene boje, dimenzije, tekstovi i stilovi. Slika 4.4. prikazuje mapu resursa.



Slika 4.4. Resursi

Layout mapa napisana je opisnim jezikom XML. XML (eng. *EXtensible Markup Language*) je široko rasprostranjen jezik te služi za razdvajanje podataka od njihove prezentacije. Sličan je HTML opisnom jeziku. Unutar *layout* datoteke stvaraju se elementi kao što su gumb, klizač, polje za unos, slike itd. Svaki *layout* mora imati jedan korijenski element u kojeg se ugnježđuju drugi elementi.

4.2.3. RecyclerView

Prije korištenja *RecyclerView-a* u *Gradle* skriptu se mora dodati komponenta. Klasa koja se koristi uz *RecyclerView* je *ViewHolder* koja predstavlja jedan element. Napraviti će se onoliko klasa *ViewHoldera* koliko je potrebno da se popuni jedan zaslon. Podaci koji se ne nalaze trenutno na ekranu recikliraju već postojeći *ViewHolder*. Zbog načina na koji radi *RecyclerView* je dobio ime. Za ostvarivanje *RecyclerView* potrebna je još jedna klasa Adapter. Ona brine o podacima koji se prikazuju u *RecyclerView.* U aktivnosti se metodom *setAdapter* povezuju adapter i *RecyclerView*. Nakon toga Adapter je spreman za popunjavanje podacima i aplikacija se može pokrenuti.

4.2.4. Aktivnosti

Klase aktivnosti nalaze se u mapi *java* zajedno s drugim klasama koje se koriste u aplikaciji.

Slika 4.5. Klase aktivnosti

Aktivnost (engl. *Activity*) je komponenta aplikacije koja predstavlja jedan zaslon, odnosno prozor aplikacije u kojem korisnik može raditi neke izmjene. Aplikacija mora imati barem jednu aktivnost, prvi zaslon koji se otvara je najčešće *MainActivity*. Prema [13] aktivnosti su jedne od osnovnih gradivnih komponenti aplikacija na Android platformi. Android aplikacije nemaju *main* metodu koja služi kao ulazna točka programa, te zbog toga aktivnost predstavlja prvu interakciju korisnika s aplikacijom i kao takva služi za daljnje kretanje u aplikaciji. Životni ciklus aktivnosti sastoji se od nekoliko metoda: *onCreate, onStart, onResume, onPause, onStop, onRestart* i *onDestroy*. Ove metode služe za upravljanje ponašanja aktivnosti, a metodom *setContentView* postavlja se sučelje koje definira izgled aktivnosti kao što je vidljivo na Slici 4.6.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
}
```

Slika 4.6. metoda *setContentView*

Aktivnosti mogu imati 4 stanja: aktivno, pauzirano, zaustavljeno i ugašeno. Aktivno stanje je kada je aktivnost vidljiva korisniku, pauzirano stanje je kada je druga aktivnosti pokrenuta.

4.3.Izrada aktivnosti za početni zaslon

U *MainActivity* klasi nalazi se početni zaslon s dva gumba. Prvi gumb „Tariffs" omogućuje korisniku da prikaže aktivnost koja sadrži sve tarife, kako bi se prije odluke o najboljoj tarifi informirao o tarifama koje može dobiti i koje su njihove vrijednosti. Ukoliko korisnik ne želi prethodno vidjeti sve tarife pritisnut će gumb „Start" koji ga vodi na zaslon za izračun najbolje tarife.

```
tariffButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        openTariffActivity();
    }
});

public void openTariffActivity(){
    Intent intent=new Intent(this,TariffActivity.class);
    startActivity(intent);
}
```

Slika 4.7. Metoda za otvaranje novog zaslona

Slika 4.7. prikazuje način na koji se otvara novi zaslon. Kada je pritisnut gumb „tariffButton" poziva se metoda *openTariffActivity* u kojoj se pokreće nova aktivnost *TarrifActivity.*

4.4.Izrada aktivnosti za unos kriterija

Kao što je prethodno spomenuto, nakon što korisnik stisne gumb „*Start*" otvara se *CalculatingActivity.* Ova klasa sadrži četiri *SeekBar*-a čije su vrijednosti zapisane u četiri *TextView*-a. Vrijednost na *SeekBaru* postavlja korisnik s obzirom na važnost pojedinog kriterija. *SeekBar* je dodatak na *ProgressBar*, razlikuje se u tome što on ima dodatak za pomicanje lijevo ili desno u slučaju napretka. Aktivnost sadrži dva gumba. Gumbom „*Calculate*" otvara se novi zaslon koji prikazuje rezultat višekriterijskog odlučivanja, gumbom „*Set equal*" vrijednosti na *seekbaru* postavljaju se na jednaku. Vrijednosti o kojima korisnik odlučuje su: cijena, količina internetskog prometa, količina SMS poruka i količina poziva. Korisnik ukoliko ne želi da su svi kriteriji jednako važni, po svojoj želji postavlja vrijednosti. Nakon što gestom dodira napravi promjene na *seekbaru* korisniku se prikazuje promjena na zaslonu. Za to je zaslužan povratni poziv *SeebBar.OnSeekBarChangeListener* koji obavještava klijenta o promjeni razine napretka. Tri su funkcije koje za to služe: *onProgressChanged, onStartTrackingTouch* i *onStopTrackingTouch.*

Kako bi se razlikovala promjena koju je napravio korisnik od promjene koje su se dogodile programski, funkcija koristi parametar *fromUser* kako je prikazano na slici 4.8. Druge dvije funkcije provjeravaju kada korisnik pokreće gestu dodira, odnosno kada ju je završio. Sve tri funkcije kao parametar primaju *SeekBar* u kojem su se promjene dogodile.

```
private void setUpListeners(){
    for(SeekBar sb : seekBars){
        sb.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int
progress, boolean fromUser) {
                updateTextViews();
            }
            tv1.setText(Double.toString(Math.round(sb1.getProgress()*100/total*1
            000)/1000));
            tv2.setText(Double.toString(Math.round(sb2.getProgress()*100/total*1
            000)/1000));
            tv3.setText(Double.toString(Math.round(sb3.getProgress()*100/total*1
            000)/1000));
            tv4.setText(Double.toString(Math.round(sb4.getProgress()*100/total*1
            000)/1000));
            }
        });
    }
}
```

Slika 4.8. Funkcionalnost ProgressBar-a

Funkcija *updateTextViews* bilježi promjene sa *SeekBar*-a te ih prepisuje na odgovarajući *TextView*. Ukoliko su svi klizači na 0, aplikacija bi dijelila s nulom i dobila kao rezultat NaN. Zbog toga ukoliko je ukupna suma svih klizača jednaka nuli, definirano je da se sve vrijednosti postave na jednaku. Podaci se dohvaćaju funkcijom *getProgress*, kako bi rezultat koji se prikazuje bio cijeli broj koristi se funkcija *Math.round* kao što je prikazano na slici 4.9.

```
tv1.setText(Double.toString(Math.round(sb1.getProgress()*100/total*1
000)/1000));
tv2.setText(Double.toString(Math.round(sb2.getProgress()*100/total*1
000)/1000));
tv3.setText(Double.toString(Math.round(sb3.getProgress()*100/total*1
000)/1000));
tv4.setText(Double.toString(Math.round(sb4.getProgress()*100/total*1
000)/1000));
```

Slika 4.9. Korištenje funkcije *getProgress* i *Math.round*

Kada se klinke gumb „*Calculate*“ poziva se funkcija *openResultActivity* koja u sebi poziva funkciju *calculate*. Funkcija *calculate* sadrži polje s 4 elementa povratnog tipa *double*. Svaki element predstavlja prioritet jedne tarife koji se računa prema formuli prikazanoj na slici 3.3. Programsko rješenje višekriterijskog odlučivanja prikazano je na slici 4.10.

```
private double[] calculate(){
    int total=calculateTotalProgress();

    double[] Tariff=new double[4];


Tariff[0]=(0.0459*sb1.getProgress()/total)+(0.5854*sb2.getProgress()/total)+
(0.5872*sb3.getProgress()/total)+(0.5872*sb4.getProgress()/total);
//Strimalica

Tariff[1]=(0.1375*sb1.getProgress()/total)+(0.2577*sb2.getProgress()/total)+
(0.2560*sb3.getProgress()/total)+(0.2560*sb4.getProgress()/total);
//Surferica

Tariff[2]=(0.2398*sb1.getProgress()/total)+(0.1080*sb2.getProgress()/total)+
(0.0990*sb3.getProgress()/total)+(0.0990*sb4.getProgress()/total);
//Sheralica

Tariff[3]=(0.5768*sb1.getProgress()/total)+(0.0478*sb2.getProgress()/total)+
(0.0578*sb3.getProgress()/total)+(0.0578*sb4.getProgress()/total);
//Spikalica

    return (Tariff);

}
```

Slika 4.10. Funkcija *calculate*

Varijabla *total* dobiva se pozivanjem funkcije *calculateTotalProgress* koja prolazi kroz listu *Seekbars* i zbraja vrijednosti sa klizača. Polje Tariff je polje prioriteta tarifa gdje Tariff[0] predstavlja prioritet tarife Strimalice, Tariff[1] prioritet tarife Surferice, Tariff[2] prioritet tarife Sheralice i Tariff[3] prioritet tarife Spikalice. Ovisno o vrijednosti koja se dohvaća s klizača metodom *getProgress*, mijenjat će se prioriteti za svaku tarifu. Prioritet za jednu tarifu računa se tako što se prioritet s obzirom na cijenu pomnoži s vrijednošću koju je korisnik postavio na klizaču i cijeli taj iznos se podjeli s ukupnim zbrojem vrijednosti sa svih klizača, nadalje taj se iznos zbraja sa umnoškom prioriteta s obzirom na količinu internetskog prometa i vrijednosti koju je korisnik podstavio na klizaču podijeljenu s ukupnim zbrojem vrijednosti sa svih klizača. Na isti način se zbrajaju se s prioritetom s obzirom na količinu SMS poruka i količinu poziva. Funkcija vraća polje prioriteta.

4.5.Izrada aktivnosti za prikaz rezultata

U ovoj aktivnosti prikazano je rješenje višekriterijskog odlučivanja. Nakon što su prioriteti izračunati metodom *calculate,* poslani su u ovu aktivnost prosljeđivanjem ID aktivnosti u kojoj su se nalazili. ID polja je „Tarifa" te se on dohvaća metodom *extras.getDoubleArray.*

```
Bundle extras=this.getIntent().getExtras();
double[] array=extras.getDoubleArray("Tarifa");
```

Slika 4.11. Dohvaćanje podataka

Prioriteti su prikazani strukturnim krugom (engl. *Pie chart*) kako bi korisnik imao grafički prikaz svake tarife. Strukturni krug podijeljen je na 4 dijela i svaki dio predstavlja vrijednost jednog prioriteta, dok cijeli krug predstavlja njihov zbroj.

```
ArrayList<PieEntry> yvalues=new ArrayList<>();

yvalues.add(new PieEntry((float)array[0],"Strimalica"));
yvalues.add(new PieEntry((float)array[1],"Surferica"));
yvalues.add(new PieEntry((float)array[2],"Sheralica"));
yvalues.add(new PieEntry((float) array[3],"Spikalica"));
```

Slika 4.12. PieChart

Na slici 4.12. vidljivo je da je stvorena lista ulaza za strukturni krug. Ulazi se dodaju metodom *add* koja kao argument prima vrijednost tipa *float* i naziv tipa *string.* Dodane su vrijednosti prioriteta za svaku tarifu i naziv te tarife. Graf je animiran metodom *animateY,* boja grafa postavljena je metodom *setColors* u koju se predaje već ugrađeni skup boja *ColorTemplate.colorfulcolors.*

4.6.Izrada aktivnosti za prikaz tarifa

Korisniku je omogućen prikaz tarifa i njenih vrijednosti u *TarrifActivity*-u. Ova aktivnost izrađena je komponentom *RecyclerView* kako bi se podaci mogli reciklirati. Kao što je objašnjeno u poglavlju 4.2.3. za funkcioniranje *RecycleView* potrebna je klasa *ViewHolder* čije se kreiranje događa u klasi Adapter. *ViewHolder* sadrži 5 atributa koji predstavljaju naziv tarife, cijenu, količinu internetskog prometa, količinu SMS-ova i količinu poziva. Konstruktor *ViewHolder*

zahtjeva slanje *View* koji predstavlja XML. XML datoteka koja opisuje izgled komponenta ima *id recyclerViewID*. Adapter klasa uz *ViewHolder* implementira još tri funkcije: *onBindViewHolder* koja ga povezuje s podacima iz adaptera, *getItemCount* koja broji koliko je elemenata u *RecyclerView* i *onCreateViewHolder* koja ga kreira. U *TariffActivity* se metodom *setAdapter* povezuje adapter s *RecyclerView* i stvaranjem nove liste se popunjavaju elementi kao što je prikazano na slici 4.13.

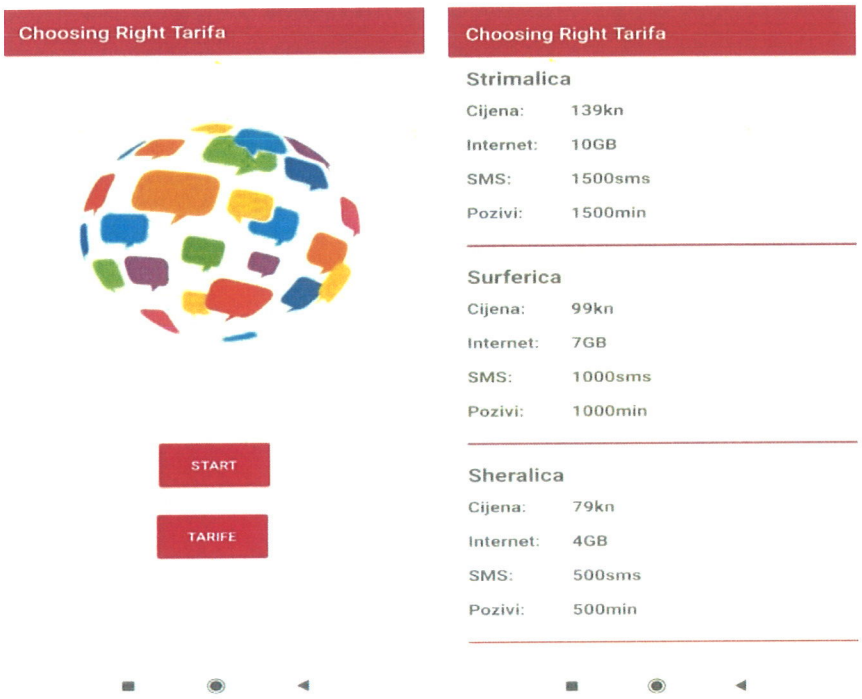
```
ListItem Tarifa1=new ListItem("Strimalica","139kn","1500sms","10GB","1500min");
ListItem Tarifa2=new ListItem("Surferica","99kn","1000sms","7GB","1000min");
ListItem Tarifa3=new ListItem("Sheralica","79kn","500sms","4GB","500min");
ListItem Tarifa4=new ListItem("Spikalica","49kn","300sms","1GB","300min");

listItems.add(Tarifa1);
listItems.add(Tarifa2);
listItems.add(Tarifa3);
listItems.add(Tarifa4);


adapter=new RecyclerAdapter(this,listItems);
recyclerView.setAdapter(adapter);
```

Slika 4.13. Popunjavanje *RecyclerView* i metoda *setAdapter*

5. ISPITIVANJE RADA I ANALIZA REZULTATA APLIKACIJE

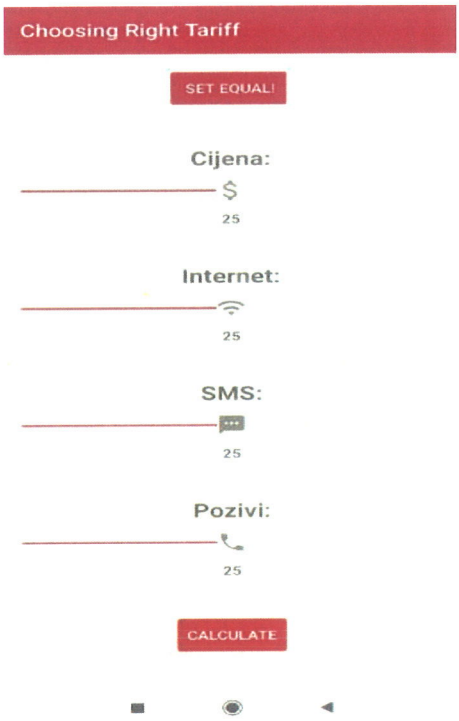
5.1.Način korištenja aplikacije

Aplikacija za izbor najbolje tarife nakon pokretanja otvara zaslon koji je prikazan na slici 5.1.



Slika 5.1. Početni zaslon

Korisnik može pritisnuti gumb „Start“ ukoliko želi početi sa izborom najbolje tarife. Ako želi najprije vidjeti sve tarife klikom na gumb „*Tariff*“ otvoriti će zaslon sa svim tarifama. Zaslon s tarifama prikazuje detalje o svakoj tarifu, kao što su cijena, količina internetskog prometa, količina SMS poruka i količina poziva. Nakon što je proučio sve tarife korisnik se vraća na početni zaslon. Klikom na gumb „*Start*“ otvara se zaslon koji je prikazan na slici 5.2.



Slika 5.2. Zaslon za izračun najpogodnije tarife

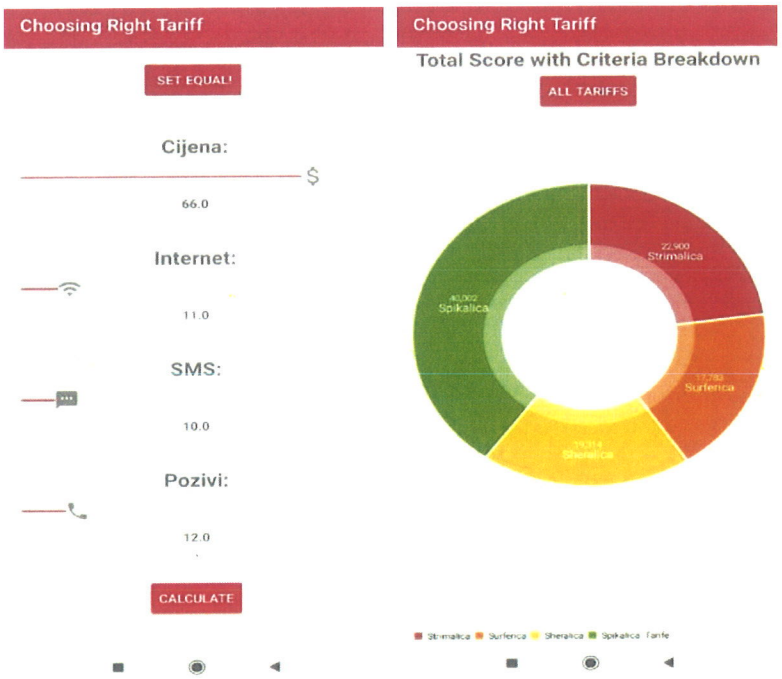
Aplikacija je napravljena vrlo jednostavno kako bi korisnik intuitivno znao što se od njega traži. Zbog toga se ikonica svakog klizača razlikuje ovisno o kojem se kriteriju radi. Na ovom zaslonu korisnik odlučuje koliko su mu važni kriteriji povlačenjem sva četiri klizača. Zbroj na svim klizačima je 100. Ukoliko želi da su svi kriteriji jednako važni korisnik ima mogućnost pritisnuti gumb „*Set equal*" koji će sve klizače postaviti na jednaku vrijednost. Nakon što je postavio svoj kriterije korisnik će odabrati gumb „*Calculate*" koji će otvoriti novi zaslon koji je prikazan na slici 5.3.

Slika 5.3. *Zaslon koji prikazuje rezultat višekriterijskog odlučivanja*

Graf prikazuje rezultat višekriterijskog odlučivanja, te sadrži legendu tarifa. Ukoliko korisnim želi provjeriti informacije o dobivenoj tarifu, pritisnut će gumb „*All tariffs*" koji će otvoriti zaslon s tarifama.

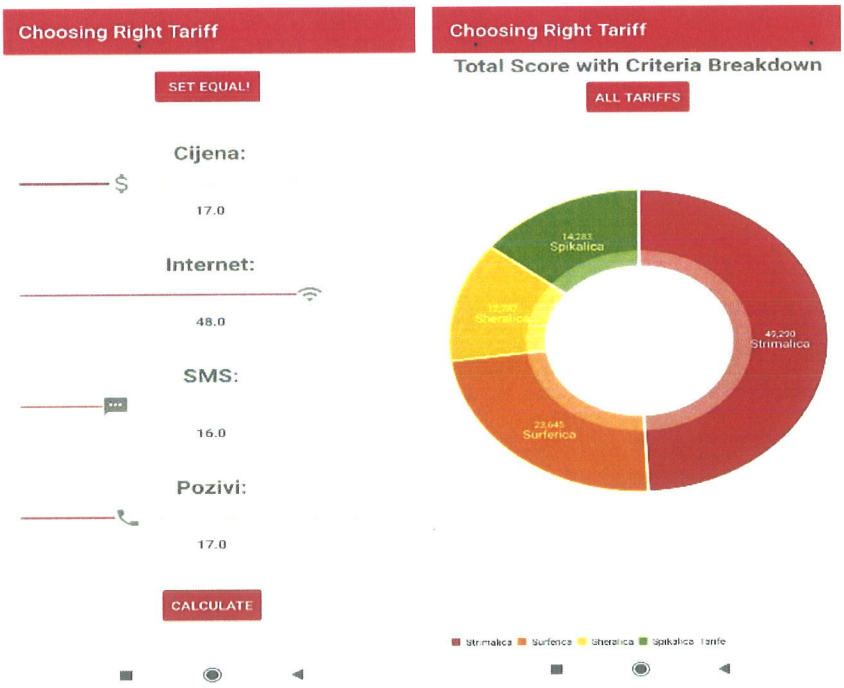
5.2. Testiranje mobilne aplikacije

Testiranje mobilne aplikacije utvrdit će ponaša li se aplikacija očekivano i zadovoljava li kriterije kojih se mora držati. To znači ukoliko se na klizaču koji prikazuje kriterij za cijenu postavi vrijednost na najveću moguću, očekivano je da će kao rezultat aplikacija izračunati najveći prioritet za tarifu Spikalicu koja ima najmanju cijenu. Isto tako ukoliko je korisniku najvažniji kriterij Internet, očekivano je da će kao rezultat tarifa Strimalica imati najveći prioritet. Korisnik prema svojim željama na klizaču postavlja prioriteta za kriterij cijena, Internet, SMS i pozivi.



Slika 5.4. Test 1

U prvom testiranju aplikacije koje je prikazano na slici 5.4. najveći prioritet imala je cijena, dok su Internet, SMS i pozivi imali manji prioritet. Tarifa koju je korisnik dobio je Spikalica, što je i pretpostavljeno jer ona ima najmanju cijenu u odnosu na druge tarife. Prikazan je i rezultat ostalih tarifa kako bi korisnik imao mogućnost izabrati drugu najbližu tarifu.



Slika 5.5. Test 2

U slučaju koji je prikazan na Slici 5.5. korisnik je postavio da mu je najveći prioritet količina internetskog prometa, te mu je kao rezultat najpogodnije tarife vraćena tarifa Strimalica. Može se pretpostaviti da s obzirom da tarifa Strimalica pruža primatelju usluge najviše internetskog prometa, SMS-ova i poziva najčešće će biti izabrana kao najpogodnija tarifa. Ostale tarife služe korisniku kako bi imao druge mogućnosti izbora.

6. ZAKLJUČAK

Cilj ovog završnog rada bila je izrada Android mobilne aplikacija za potporu pri izboru najpogodnije tarife zasnovane na višekriterijskom dodjeljivanju. Koristio se programski jezik java u Android Studio okruženju. Cilj je bio predložiti model višekriterijskog odlučivanja, te ga implementirati u aplikaciji. Izabran je AHP model. Aplikacija od korisnika traži unos vlastitih prioriteta putem klizača, te na temelju toga izračunava prioriteta svih tarifa. Korisniku je pružen grafički prikaz rješenja, te uvid u sve tarife. Analiza aplikacije obavljena je za dva slučaja, te je dokazano da aplikacija radi ispravno. Na temelju analize može se zaključiti da će korisnik najčešće kao rezultat optimalne tarife dobiti tarifu Spikalicu koja ima najmanju cijenu ali isto tako najmanju količinu internetskog prometa, sms-ova i poziva ili tarifu Surfericu koja ima najveću cijenu, količinu internetskog prometa, sms-ova i poziva u odnosu na ostale tarife. Aplikacija je brza i efikasna te ne zahtjeva veliku količinu vremena za izvođenje. Moguća su dodatna proširenja u aplikaciji kao što su novi kriteriji: količina internacionalnih poziva, dužina trajanja tarife, itd. Moguće je dodati nove tarife sukladno novitetima na tržištu. Također logika AHP modela omogućuje izradu aplikacija za rješavanje drugih višekriterijskih problema kao što su: izbor najboljeg mobitela, automobila, studija, itd.

LITERATURA

[1] Multi-criteria Decision Analysis for Supporting the Selection of Engineering Materials in Product Design (Second Edition), 2016 ., https://www.sciencedirect.com/topics/engineering/multi-criteria-decision-making [srpanj 2020.].

[2] Multi -Criteria Decision Analysis., https://projects.ncsu.edu/nrli/decision-making/MCDA.php [Srpanj, 2020.].

[3] Methods and features., https://en.wikipedia.org/wiki/Decision-making_software [srpanj, 2020.

[4] Primjena metoda za višekriterijsko odlučivanje u određivanju prioriteta u visokom obrazovanju http://higherdecision.foi.hr/sites/default/files/Primjena%20metoda%20za%20vi%C5%A1ekriterijsko%20odlu%C4%8Divanje_Begicevic.pdf [srpanj, 2020.]

[5] Analytic hierarchy process AHP https://en.wikipedia.org/wiki/Analytic_hierarchy_process [srpanj, 2020.]

[6] ELECTRE https://en.wikipedia.org/wiki/ELECTRE_ [srpanj, 2020.]

[7] Višekriterijska metoda AIRM https://en.wikipedia.org/wiki/Aggregated_indices_randomization_method [srpanj, 2020.]

[8] Smart-Picker http://www.smart-picker.com/ [srpanj, 2020.]

[9] Bijay K. Jayaswal, Peter C, Patton, Ernest H. Forman, The Analytic Hierarchy Process in Software Development, Pearson Education, 2007.

[10] Primjena AHP-metode kao alata za optimalni izbor opreme https://www.fsb.unizg.hr/atlantis/upload/newsboard/21_10_2011__15692_Odrzavanje_AHP_Metoda_Izbor_Opreme.pdf [srpanj, 2020.]

[11] Izračun svojstvenih vektora https://people.revoledu.com/kardi/tutorial/AIIP/Priority%20Vector.htm?fbclid=IwAR0shyvroXtTYOB7GbJZQfjTsE9X690zSLuLjUhHGOcW5GNu5aXK1eMSASI [kolovoz, 2020.]

[12] Java programming language https://e+n.wikipedia.org/wiki/Java_(programming_language) [kolovoz, 2020.]

[13] Meet Andorid Studio https://developer.android.com/studio/intro/?gclid=Cj0KCQjwp4j6BRCRARIsAGq4yME9gPduq7U9HllKpc9gdve79AQlgR7WIJlGZXyrjrP58WQ4moq_1RUaAh_1EALw_wcB&gclsrc=aw.ds [kolovoz, 2020.]

[14] Activity https://developer.android.com/guide/components/activities [kolovoz, 2020.]

SAŽETAK

Cilj ovog završnog rada je izrada Android mobilne aplikacija za potporu pri izboru najpogodnije tarife zasnovane na višekriterijskom dodjeljivanju. Mobilna aplikacija omogućuje korisniku odabir najbolje tarife prema željenim prioritetima. Kriteriji o kojima ovisi izbor najpogodnije tarife su: cijena, količina Internetskog prometa, količina SMS poruka i količina poziva. Za izračun prioriteta koristi se Saaty skala koja je ključan dio AHP modela. AHP ili proces analitičke hijerarhije je strukturna tehnika organiziranja i analiziranja kompleksnih odluka, utemeljena na matematici i psihologiji. Za izradu mobilne aplikacije korišteno je okruženje Android Studio te programski jezik Java. Detaljno je objašnjena izrada svih aktivnosti koje predstavljaju jedan zaslon aplikacije i sve komponente koje se unutar njih koriste. Mobilna aplikacija testirana je na odabranom skupu ulaznih podataka čiji se rezultat prikazuje na kružnom grafu.

Ključne riječi: Android, Java, višekriterijsko dodjeljivanje, AHP, mobilna aplikacija

ABSTRACT

Mobile android application for support in selecting of services based on multicriterium allocation

The main goal of this paper is to develop an Android mobile application to support the selection of the most optimal tariff based on multi-criteria decision making (MCDM). The mobile application allows the user to select the best tariff based on the desired priorities. Criteria on which the choice of the most favorable tariff depends are price, quantity of Internet traffic, quantity of SMS and quantity of calls. The Saaty scale, which is a key part of the AHP model, is used to calculate priorities. AHP is a structural technique of organizing and analyzing complex decisions, based on mathematics and psychology. The mobile application was created by using the Android Studio environment and the programming language Java. The creation of all activities that represent one screen of applications and all components used within them is explained in detail. Mobile application tested on a selected set of input data whose result is displayed on a group graph.

Keywords: AHP, Android, Java, mobile application, multi-criteria decision making

ŽIVOTOPIS

Barbara Aščić rođena je 19.02.1999. godine u Đakovu. Pohađala je Osnovnu školu Tenja u Tenji. Nakon toga upisuje I. Gimnaziju Osijek u Osijeku. Nakon završetka srednjoškolskog obrazovanja s odličnim uspjehom, upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija u Osijeku na sveučilištu Josipa Jurja Strossmayera, smjer preddiplomski studij računarstva. Na drugoj godini studija dobiva STEM stipendiju. Od programskih jezika učila je C, C++, C#, Javu i HTML.

Barbara Aščić

PRILOZI

Programsko rješenje aplikacije (nalazi se na priloženom CD-u)