Sustav za pomoć u odabiru računalnih komponenata

Dragić, Domagoj

Undergraduate thesis / Završni rad

2021

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:183900

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-19**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

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

Sveučilišni studij

SUSTAV ZA POMOĆ U ODABIRU RAČUNALNIH KOMPONENATA

Završni rad

Domagoj Dragić

Osijek. 2021.

SADRŽAJ

1.UVOD 1
1.1. Zadatak završnog rada 2
2. PREGLED PODRUČJA TEME 3
2.1. Postojeće mobilne i web aplikacije 3
2.1.1. PC Builder 3
2.1.2. Upgrade PC 4
2.1.3. PCPartPicker 5
2.1.4. Jeftinije.hr 6
2.2. Kompatibilnost računalnih komponenata 6
2.2.1. Kućište 7
2.2.2. Matična ploča 7
2.2.3. Procesor 8
2.2.4. RAM memorija 9
2.2.5. Grafička kartica 9
2.2.6. Hladnjak za procesor 9
2.2.7. Napajanje 9
3. SUSTAV ZA POMOĆ U ODABIRU RAČUNALNIH KOMPONENATA 10
3.1. Funkcionalni zahtjevi na aplikaciju 10
3.2. Arhitektura aplikacije 11
3.2.1. Dijagram tijeka aplikacije 11
3.2.2. JSON datoteka i baza podataka 12
3.3. Korištene tehnologije 14
3.4. Implementacija rješenja 16
3.4.1. Baza podataka 16
3.4.2. Modul za prikazivanje 19
3.4.3. Modul za pretraživanje 23
3.4.4. Prikupljanje podataka 24
3.4.5. Popis komponenata 26
4. ISPTIVANJE FUNKCIONALNOSTI 27
4.1. Opis načina korištenja aplikacije 27
4.2. Testiranje rada aplikacije 30
4.2.1. Prvi slučaj 30
4.2.2. Drugi slučaj 30
4.2.3. Interpretacija rezultata testiranja 31

5. ZAKLJUČAK 32
LITERATURA 34
SAŽETAK 35
ABSTRACT 36
ŽIVOTOPIS 37

1.UVOD

U današnje vrijeme teško je zamisliti kućanstvo bez nekakve moderne tehnologije. Razvojem interneta i računalnih tehnologija usporedno se pojavljuje i rastuća potražnja za samim računalima, laptopima, mobitelima i slično. Većina bolje opremljenih informatičkih trgovina u svojoj ponudi ima gotove, unaprijed sastavljene računalne konfiguracije, kao i pojedinačne komponente koje kupac može izabrati. Ovdje se javlja prvi problema koji je usko vezan uz temu ovoga završnoga rada. Naime, pitanje je isplati li se više kupcu uzeti već unaprijed sastavljenu računalnu konfiguraciju ili je ipak bolje kupiti pojedinačne komponente? Odgovor je svakako kupovina pojedinačnih komponenata, a iza toga stoje 2 najbitnija razloga: cijena i fleksibilnost. Usporedbom cijene unaprijed sastavljenog računala i ukupne cijene istih komponenata kupljenih pojedinačno, može se zaključiti kako je uvijek jeftinije samostalno kupiti komponente. Razlog tomu je što sve trgovine uzimaju veliku proviziju kako za sastavljanje samoga računala, tako i za brojne druge stavke koje često nisu potrebne. Što se tiče fleksibilnosti, vrlo je lako uočiti prednost kupovine pojedinačnih komponenata. Unaprijed sastavljene konfiguracije dolaze s fiksnim komponentama i vrlo je vjerojatno da sve te komponente neće zadovoljiti želje kupca. Upravo zbog ovih nabrojanih razloga, sve se više ljudi odlučuje za samostalnu nabavu komponenata za svoja računala. Drugi problem kojim se ovaj rad bavi jest upravo dostupnost i nabava računalnih komponenata na domaćem, Hrvatskom, tržištu.

Cilj ovog završnog rada je dati pregled postojećih komercijalnih rješenja ovih problema, napraviti usporedbu tih rješenja, te na temelju dobivenih informacija napraviti vlastito rješenje u obliku mobilne aplikacije koja će korisniku olakšati izbor računalnih komponenata. Aplikacija korisniku daje pregled komponenata po kategorijama (procesori, grafičke karte, ram pločice,…). Nakon što korisnik odabere željenu kategoriju te izabere specifičnu komponentu dobiva uvid u detaljnije informacije o samoj komponenti, a klikom na link može saznati potpune specifikacije o željenoj komponenti. Također, korisnik klikom na jedan od 2 ponuđena gumba može vidjeti dostupnost i cijenu izabrane komponente na domaćem tržištu. U slučaju da se korisniku svidi izabrana komponenta, može ju spremiti i pogledati kasnije.

U drugom poglavlju bit će prikazana slična web i mobilna rješenja te će biti opisano kako pravilno upariti sve komponente kako bi računalo funkcioniralo. U trećem poglavlju bit će prikazan rad aplikacije kroz dijagram tijeka, funkcionalni zahtjevi traženi od aplikacije te prikazuje konkretnu implementaciju rješenja. Četvrto poglavlje prikazuje praktičan način korištenja aplikacije te testiranje i isplativost same aplikacije.

1.1. Zadatak završnog rada

U završnom radu potrebno je analizirati postojeće aplikacije za odabir računalnih komponenti. Također, potrebno je proučiti i opisati tehnologije korištene u izradi takvog rješenja. Uz to, potrebno je implementirati vlastito rješenje u smislu mobilne android aplikacije te opisati okolinu korištenu za razvoj mobilne aplikacije. U aplikaciji je potrebno obuhvatiti sve komponente potrebne za izgradnju računala. Nadalje, potrebno je korisniku omogućiti uvid u dostupnost i cijenu izabranih komponenata na domaćem tržištu. Na kraju je potrebno ispitati rad aplikacije te analizirati dobivena rješenja.

2. PREGLED PODRUČJA TEME

Ovo poglavlje bazirat će se na već postojećim komercijalnim rješenjima te pravilima za ispravan odabir odgovarajućih računalnih komponenata.

2.1. Postojeće mobilne i web aplikacije

Unutar ovog odlomka bit će prikazane neke od poznatih mobilnih i web aplikacija koje su usko vezane uz odabir računalnih komponenata.

2.1.1. PC Builder

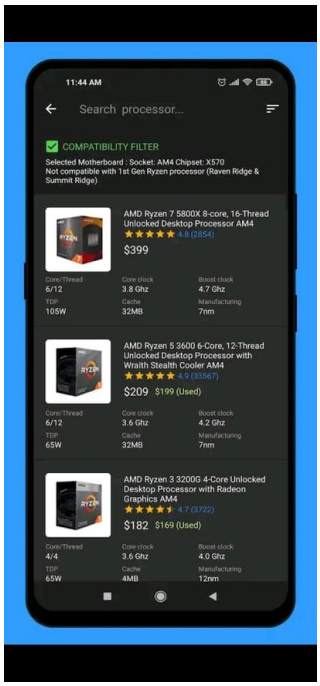
PC Builder – Part picker & Generate Builds! [1] (PC Builder u nastavku), je mobilna aplikacija koja korisniku pomaže u samostalnoj izradi vlastitog računala. Aplikaciju je napravio strani programer koji zbog anonimnosti koristi ime Codewaster, a pruža izvrsno rješenje za izbor komponenata na stranom tržištu što dokazuje više od 100 tisuća preuzimanja ove aplikacije te izvrsna ocjena 4.3/5 na Google Play Storeu.

Korisnik može izabrati komponente na temelju filtera međusobne kompatibilnosti ili može generirati gotovu konfiguraciju na temelju osobnog budžeta. U slučaju samostalnog odabira komponenata, korisnik izabire komponente grupirane u kategorije (*CPU, GPU, RAM, Motherboard.,…*) te dobiva pregled trenutno dostupnih komponenata na tržištu. Budući da je ovaj projekt dio programa *Amazon Associate*, željene komponente mogu se kupiti putem Amazona preko ponuđenih linkova u aplikaciji.

Značajka automatske izgradnje konfiguracije pokušava pronaći najbolje komponente po performansi za zadani budžet. Također, u obzir se uzimaju ocjene na tržištu kao i prethodna iskustva starih korisnika. Na taj način, osigurava se veća uspješnost pronalaska komponenti kao i što bolje iskustvo za korisnika.

Na slikama 2.1.a), 2.1.b), 2.1.c), prikazan je izgled aplikacije PC Builder.

Slika 2.1.a) Prikaz kategorija komponenata



Slika 2.1.b) Prikaz liste komponenata

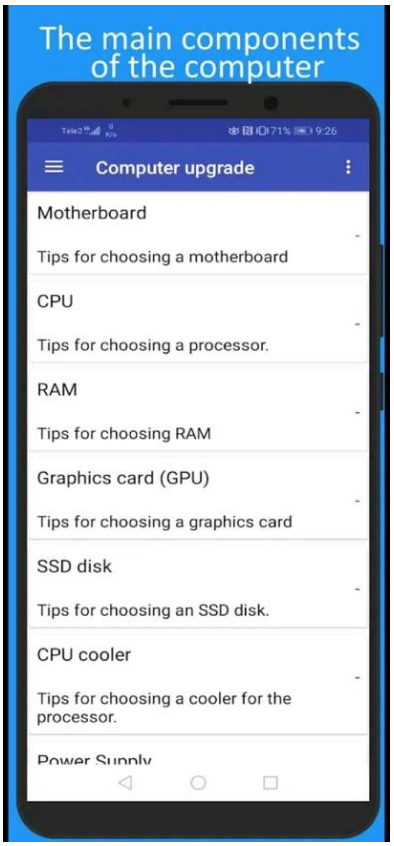
Slika 2.1.c) Prikaz detalja odabrane komponente

2.1.2. Upgrade PC

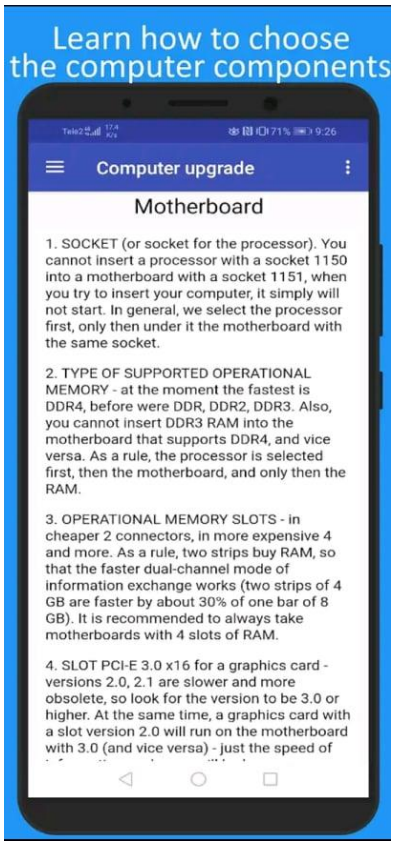
Upgrade PC - build or upgrade your computer [2] (Upgrade PC u nastavku), je mobilna aplikacija koju je proizvela tvrtka FutureDevelop. Ova aplikacija ima veliku raznovrsnost uporabe. Korisniku pruža osnovne informacije o računalnim komponentama, pruža savjete o kompatibilnosti pojedinih komponenata, pruža savjete kako unaprijediti svoj stari sustav, te daje informacije kako sastaviti samo računalo. Aplikaciju je preuzelo više od 50 tisuća ljudi, a ocijenjena je vrlo visoko, 4.4/5.

Aplikacija služi kao izvrstan vodič za one malo manje upućene u svijet moderne računalne tehnologije. Brojnim savjetima o svakoj pojedinačnoj komponenti vodi korisnika kroz svaku specifikaciju komponente i razjašnjava razlike između sličnih komercijalnih ponuda iste komponente. Također, aplikacija ima dio za usporedbu različitih komponenata iste vrste pa tako korisnik dobiva dobar uvid u sve prednosti i mane određene komponente nad drugom.

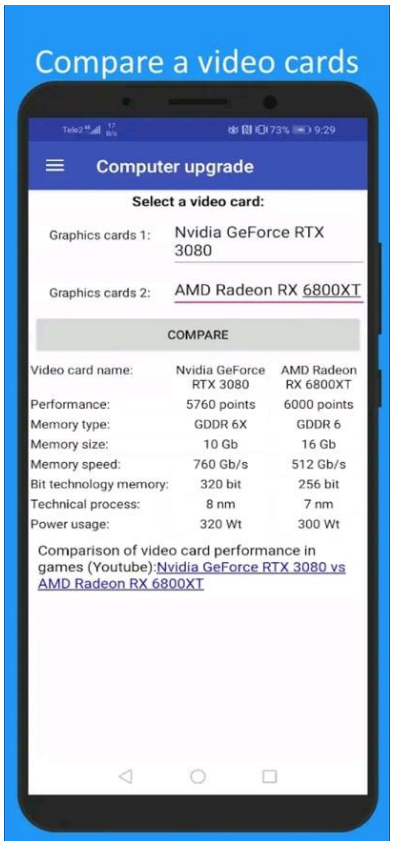
Slike 2.2.a), 2.2.b), 2.2.c) daju pregled aplikacije Upgrade PC.



Slika 2.2.a) Prikaz dijela za unaprjeđenje računala



Slika 2.2.b) Prikaz savjeta za izbor komponente



Slika 2.2.c) Prikaz usporedbe dvaju komponenata

2.1.3. PCPartPicker

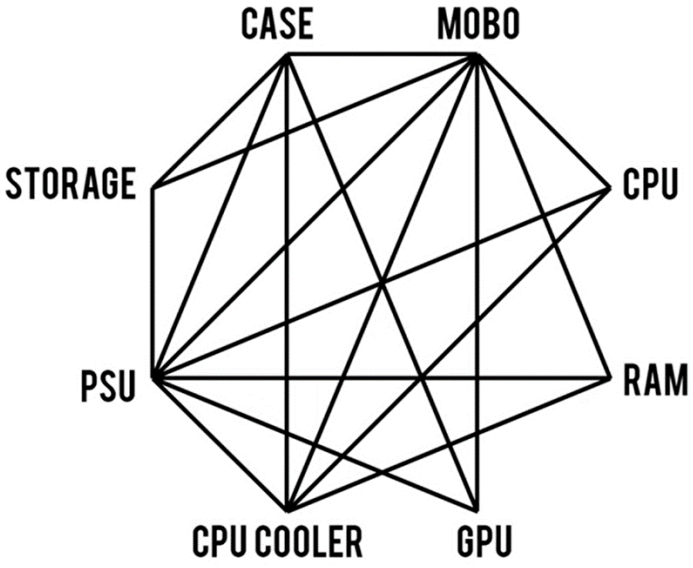
Prema [3], PCPartPicker je web aplikacija koja pruža odabir računalnih komponenata. Jedna je od najpoznatijih stranica koje se bave računalnim komponentama. Imaju vrlo razvijenu zajednicu koja putem foruma komentira te daje svoje prijedloge vezane za izgradnju računala novim članovima. PCPartPicker je originalno napravljen za američko tržište, no koriste ga ljudi diljem svijeta zbog jednostavnosti korištenja same aplikacije. Neke od funkcionalnosti ove web aplikacije su izbor računalnih komponenata, pružanje smjernica o kompatibilnosti izabranih komponenata, pregled cijena i korištenja energije pojedinih komponenata. PCPartPicker povlači podatke iz desetak najpopularnijih trgovina računalnim komponentama te ih filtrira po cijeni i dostupnosti. Svakodnevno se ažuriraju cijene komponenata, a također postoje i grafikoni prijašnjih cijena za svaku komponentu te tako korisniku pružaju potpuni uvid u povijest cijene za pojedinu komponentu. Korisniku se također pruža mogućnost pretplate na određenu komponentu putem e-maila. Kada cijena željene komponente padne ispod zadane cijene, korisnik putem e-maila prima obavijest o tome.

2.1.4. Jeftinije.hr

Jeftinije.hr [4], je web aplikacija koja uspoređuje cijene proizvoda brojnih domaćih web-trgovina. Trenutno se na stranici prikazuju proizvodi iz 349 hrvatskih web-trgovina. Stranica svojim korisnicima nudi izbor među raznim kategorijama proizvoda, pa tako i računalnim komponentama. Nakon što se odabere željeni proizvod, ova aplikacija prolazi kroz sve svoj partnerske web-trgovine te korisniku prikazuje cijenu tog proizvoda. Na taj način, vrlo je lako pronaći trenutnu najjeftiniju ponudu na tržištu te tako kupac može uštediti mnogo novca. Upravo zbog ove funkcionalnosti i činjenice da stranica dohvaća cijene računalnih komponenata, ona će biti korištena kao dio rješenja aplikacije ovog završnog rada.

2.2. Kompatibilnost računalnih komponenata

Kako bi samostalno sastavljanje računala bilo uspješno, potrebno je imati neka predznanja o kompatibilnosti samih komponenata. Iako mnoge današnje web stranice i aplikacije nude provjere kompatibilnosti komponenata, vrlo je vjerojatno kako neće sve komponente biti kupljene na istom mjestu. Isto tako, poznavanjem ovih pravila olakšan je pronalazak komponenata te se tako štedi vrijeme. Upravo zbog gore navedenih razloga, prema [5,6] bit će definirana skupina pravila kompatibilnosti za osnovne računalne komponente. Kako bi objašnjene bilo što jednostavnije i razumljivije, komponente će biti postavljene u osmerokut (Slika 2.3.) i povezane linijama. Ako postoji linija između dvije komponente, tada se one moraju slagati u određenoj specifikaciji kako bi bile međusobno kompatibilne.



Slika 2.3. Osmerokut kompatibilnosti komponenata, izvor: [6]

2.2.1. Kućište

Iako osmerokut komponenata izgleda poprilično složeno, ako se razdvoji na svaku komponentu zasebno te se redom gledaju njezine relacije tada se osmerokut značajno pojednostavljuje. Prva relacija koja će biti analizirana je kućište-matična ploča. Matične ploče dolaze u raznim veličinama (*mini-ITX, micro-ATX, ATX, extended-ATX*[1]). Isto tako i kućišta dolaze u raznim veličinama te je bitno dobro proučiti koje matične ploče određeno kućište podržava prije kupovine. Također, kućište treba imati odgovarajuće kablove za spajanje prednjeg panela kućišta na matičnu ploču. Ova informacija se također može provjeriti u specifikacijama samog kućišta.

Iduća relacija je kućište-grafička kartica. Jedina stvar na koju ovdje treba pripaziti jest dužina grafičke kartice. Bitno je provjeriti koja je maksimalna dužina grafičke kartice koju kućište može podržati.

Slično je i sa sljedećom relacijom, kućište-hladnjak procesora. U ovom slučaju bitno je provjeriti koja je visina hladnjaka za procesor te podržava li kućište takve hladnjake.

Kod relacije kućište-napajanje situacija je nešto složenija. Napajanja, kao i matične ploče, dolaze u više od jedne veličine. To znači da je bitno provjeriti podržava li kućište određeni format željenog napajanja. Nadalje, bitno je znati sadrži li kućište nekakve dodatne kontrolore za ventilatore ili led svijetla koja se spajaju na napajanje. Ovakve stvari su najčešće povezane SATA ili mulex kablovima pa je bitno da napajanje ima dovoljno odgovarajućih utora kako bi se sve moglo spojiti.

Posljednja relacija vezana za kućište je kućište-uređaji za pohranu. Budući da se većina novih tvrdih i *solid state* diskova proizvode u dvije veličine: 6.35 cm i 8.89 cm (2.5″ i 3.5″) potrebno je provjeriti ima li kućište dovoljan broj mjesta za montiranje određenih diskova za pohranu.

2.2.2. Matična ploča

Iduća komponenta na redu je matična ploča. Budući da je ovo komponenta na koju se spajaju sve komponente računala, za očekivati je da ima relacija sa svim ostalim komponentama.

Relacija matična ploča-kućište je već objašnjena u prošlom poglavlju.

Relacija matična ploča-procesor, ove dvije komponente moraju se slagati u odgovarajućem utoru za procesor (eng. *socket type*). Postoje 2 proizvođača procesora: Intel i AMD. Noviji Intelovi

[1] ATX- Advanced Technology eXtended, je prihvaćeni format matičnih ploča te se danas koristi kao standard u izradi matičnih ploča

procesori imaju utore : LGA 1150, 1151, 1200, dok su kod AMD-ovih procesora to: FM2+ i AM3+. Ovi tipovi se moraju slagati s matičnom pločom kako bi mogli funkcionirati.

U slučaju relacije matična ploča-RAM memorija bitno je da matična ploča podržava odgovarajući standard, kapacitet i brzinu RAM pločice. Današnje matične ploče podržavaju DDR-3 i DDR-4 memoriju. Također je bitno znati koliki je maksimalni kapacitet RAM memorije koji matična ploča podržava, kao i brzine u megahertzima kako bi sve ispravno radilo.

Matična ploča-grafička kartica. U današnje vrijeme, većina matičnih ploča ima PCIe utore za grafičku karticu koji je standard u današnjoj proizvodnji. Ipak, dobro je provjeriti slažu li se generacije PCIe-a među komponentama.

Relacija matična ploča-hladnjak za procesor je vrlo slična relaciji s procesorom. I u ovom slučaju jedina bitna stvar jer poklapanje u tipu utora. Moderni hladnjaci većinom podržavaju veći broj tipova utora, tako da ovo ne bi trebao biti problem.

Kao i sve ostale komponente u računalu, matičnu ploču također pokreće napajanje. Ova relacija, matična ploča-napajanje, vezana je uz konektore na kablovima od napajanja. Prvo treba pogledati utor za napajanje procesora na matičnoj ploči. Ovakav utor zahtjeva 4 ili 8-iglični konektor. Zatim je potrebno pogledati ATX utor. Za spajanje na ovaj utor potreban je 20 ili 21-iglični konektor. Za oba slučaja bitno je ustanoviti ima li željeno napajanje odgovarajuće konektore na kablovima.

Posljednja relacija ove komponente je matična ploča-uređaj za pohranu. Ako se odluči staviti više uređaja za pohranu u računalo, potrebno je provjeriti podržava li matična ploča dovoljan broj odgovarajućih utora. Tvrdi diskovi spajaju se SATA-dana kablovima dok se *solid state* diskovi mogu montirati na M.2 utor.

2.2.3. Procesor

Relacija procesor-matična ploča je već objašnjena u prethodnom poglavlju.

Relacija procesor-hladnjak također je već spomenuta. Ako je pronađen odgovarajući procesor za matičnu ploču i hladnjak za procesor se slaže sa tom istom matičnom pločom tada se i procesor i hladnjak za procesor moraju slagati po tipu utora.

Relacija procesor-napajanje je gotovo zanemariva budući da je grafička kartica glavni potrošač. No ipak, u specifikacijama procesora može se pronaći podatak koji govori koliko snage je potrebno za pokretanje procesora.

2.2.4. RAM memorija

Budući da je relacija RAM memorija-matična ploča već objašnjena, jedina relacija koju je još potrebno objasniti jest RAM memorija-hladnjak procesora. Ova relacija je na neki način specifična budući da se radi o fizičkoj kompatibilnosti. U ovom slučaju potrebno je provjeriti visinu RAM pločica kao i širinu samog hladnjaka procesora kako ne bi došlo do međusobnih smetnji između ove dvije komponente. Jedan od načina rješavanja ovoga problema je kupovina RAM pločica s nižim profilom kako bi bilo dovoljno prostora da si komponente međusobno ne smetaju.

2.2.5. Grafička kartica

Dvije od tri relacije grafičke kartice su već objašnjene tako da je potrebno analizirati još samo relaciju grafička kartica-napajanje. U specifikacijama boljih grafičkih kartica navedeno je minimalno napajanje potrebno za pokretanje sustava. Dobra praksa je uzeti barem 100W jače napajanje od preporučenog. Mnoge stranice nude provjeru potrošnje samih komponenata pa je to još jedan od načina na koji se može izabrati odgovarajuće napajanje. Također je bitno provjeriti ima li napajanje odgovarajuće PCIe kablove za grafičku karticu. Većina današnjih kartica zahtjeva 6 ili 8-iglične konektore.

2.2.6. Hladnjak za procesor

Do sada su analizirane skoro sve relacije vezane uz hladnjak za procesor, no postoje rijetke situacije kada može doći do smetnje na relaciji hladnjak za procesor-napajanje. Ovo je slučaj samo u konfiguracijama s vrlo malim kućištem kada se napajanje montira direktno iznad hladnjaka za procesor. U tom slučaju potrebno je kupiti niskoprofilni hladnjak.

2.2.7. Napajanje

Jedina relacija koju je još potrebno objasniti je napajanje-uređaji za pohranu. Potrebno je utvrditi ima li napajanje dovoljan broj SATA kablova za sve potrebne diskove za pohranu.

3. SUSTAV ZA POMOĆ U ODABIRU RAČUNALNIH KOMPONENATA

Ovo poglavlje daje pregled svih funkcionalnih zahtjeva koje aplikacija treba zadovoljiti te kroz dijagram tijeka prikazuje rad aplikacije i daje uvid u najbitnije dijelove koda.

3.1. Funkcionalni zahtjevi na aplikaciju

Cilj aplikacije je dati korisniku na uvid ponudu komponenata na domaćem tržištu te mu tako olakšati izbor pri kupovini komponenata. Kako bi korisnik mogao što lakše rukovati aplikacijom postavljeni su osnovni funkcionalni zahtjevi koje sama aplikacija mora zadovoljiti, a ti zahtjevi su sljedeći:

- Na početnom zaslonu korisniku se na izbor nude svi tipovi računalnih komponenata (procesori, grafičke kartice, ram memorija,…)
- Nakon odabira kategorije korisniku je potrebno prikazati sve komponente iz odabrane kategorije na idućem zaslonu
- Kako bi korisnik dobio potpuni uvid u komponente koje se nude na tržištu potrebno je popis komponenata preuzeti s nekog vanjskog provjerenog izvora
- Zbog velikog broja komponenata potrebno je koristiti bazu podataka te u nju spremiti sve komponente
- Također, zbog velikog broja komponenata potrebno je implementirati sustav za pretragu kako bi korisnik brže i lakše došao do željene komponente
- Korisniku je potrebno omogućiti detaljan pregled izabrane komponente kao i prikaz cijene i ponude na domaćem tržištu za tu komponentu
- Potrebno je napraviti zasebnu listu omiljenih komponenata na koju će korisnik moći spremiti komponente koje mu se svide ili ih želi pogledati kasnije

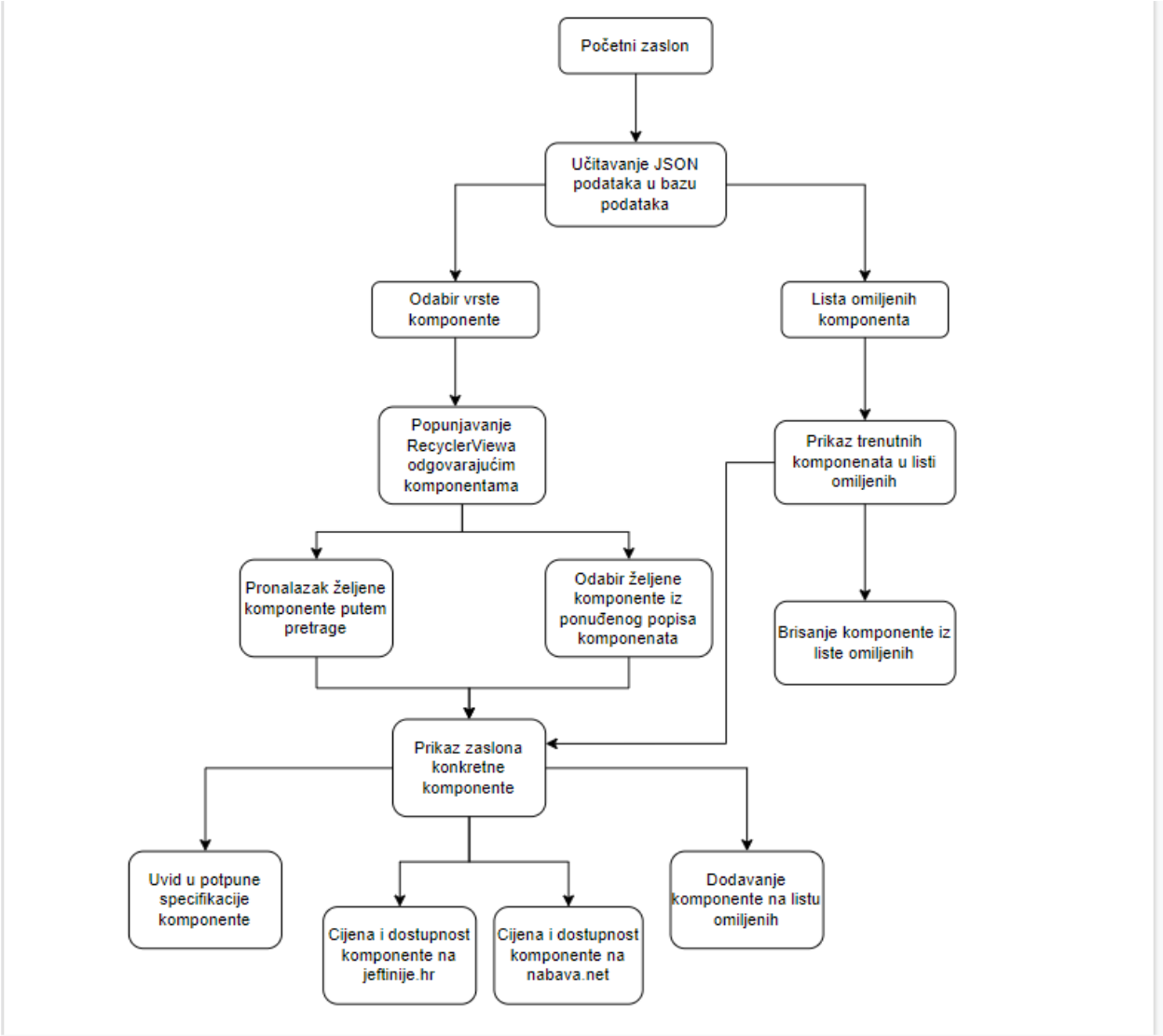
Svi ovi zahtjevi postavljeni su u svrhu dobre konstrukcije aplikacije te što boljeg korisničkog iskustva. Konkretna implementacija navedenih funkcionalnih zahtjeva bit će prikazana i opisana u sljedećem poglavlju kao najvažniji dijelovi koda.

3.2. Arhitektura aplikacije

Aplikacija korisniku pruža uvid u računalne komponente, te na temelju njegovog odabira, prikazuje najpovoljnije cijene komponente na domaćem tržištu.

3.2.1. Dijagram tijeka aplikacije

Zbog olakšanja kreiranja aplikacije kao i lakšeg i razumljivijeg objašnjenja aplikacije, napravljen je dijagram tijeka rada aplikacije koji je prikazan na slici 3.1. Prilikom prvog pokretanja aplikacije baza podataka se puni JSON podacima koji sadrže informacije o komponentama te se prikazuje početni zaslon. Na početnom zaslonu, korisnik može izabrati tip komponente (procesor, grafička kartica, ram memorija,…) koja ga zanima ili kliknuti na gumb favoriti. Ako korisnik odabere neki tip računalne komponente, aplikacija ga odvodi na idući zaslon koji se popunjava odgovarajućim komponentama iz ranije spomenute baze podataka. Korisnik sada vidi listu konkretnih komponenata koju može pretražiti putem jednostavne pretrage ili kliknuti na željenu komponentu. Nakon odabira konkretne komponente, aplikacija vodi korisnika na idući zaslon gdje mu se nudi više mogućnosti. Klikom na prikazani link korisnika se odvodi na jednu od provjerenih stranica koje sadrže detaljne informacije o svim specifikacijama odabrane komponente. Ako korisnik klikne na jedan od dva gumba: jeftinije.hr ili nabava.net, odvodi ga se na odgovarajuću stranicu. Na tim stranicama korisnik dobiva uvid o dostupnosti i cijeni izabrane komponente. Korisnik također može kliknuti na gumb dodaj u favorite kojim odabranu komponentu dodaje na popis omiljenih komponenata. Popis omiljenih komponenata moguće je pogledati klikom na gumb favoriti na početnom zaslonu, a on se također popunjava komponentama iz iste baze podataka kao i ranije. Također, ako se korisnik predomisli, u svakom trenutku može ukloniti određenu komponentu s liste omiljenih.



Slika 3.1. Dijagram tijeka rada aplikacije

3.2.2. JSON datoteka i baza podataka

Kao što je već ranije spomenuto, za ostvarivanje ove aplikacije potrebno je koristiti bazu podataka. Budući da su informacije o komponentama prikazane JSON datotekom spremit će ih se u bazu podataka. Puno je ispravnije jednom učitati sve komponente u bazu podataka pa onda, kada su one potrebne, jednostavno pročitati iz baze podataka nego prilikom svakog pokretanja aplikacije ponovno čitati JSON datoteku. Svaki atribut računalne komponente JSON datoteke predstavljen je identičnim atributom u bazi podataka. Tako je vrlo jednostavno prebaciti podatke iz JSON datoteke u željenu bazu podataka. Drugi razlog korištenja baze podataka u ovoj aplikaciji je zahtjev da korisnik mora moći samostalno dodavati i uklanjati pojedine komponente s liste omiljenih komponenata. Budući da je potrebno pamtiti koje komponente se nalaze na toj listi, idealno

rješenje je baza podataka. U bazu popunjenu informacijama o komponentama dodan je još jedan atribut *„favorite“* koji kod svih komponenata ima početnu vrijednost 0. U ovoj aplikaciji korištena je Room baza podataka. Na slici 3.2. bit će prikazane informacije o komponentama u JSON datoteci.

components.json

```
{
 "components":[
 {
   "Type": "CPU",
   "Part Number": "BX8070811900K",
   "Brand": "Intel",
   "Model": "Core i9-11900K",
   "Rank": 1,
   "Benchmark": 109,
   "Samples": 2268,
   "URL": "https://cpu.userbenchmark.com/Intel-Core-i9-11900K/Rating/4110"
 },
 {
   "Type": "CPU",
   "Part Number": "BX8070811700K",
   "Brand": "Intel",
   "Model": "Core i7-11700K",
   "Rank": 2,
   "Benchmark": 107,
   "Samples": 4308,
   "URL": "https://cpu.userbenchmark.com/Intel-Core-i7-11700K/Rating/4107"
 },
```

Slika 3.2. Prikaz JSON datoteke računalnih komponenata

Na slici 3.3. je prikazan izgled baze podataka popunjene istom tom JSON datotekom.

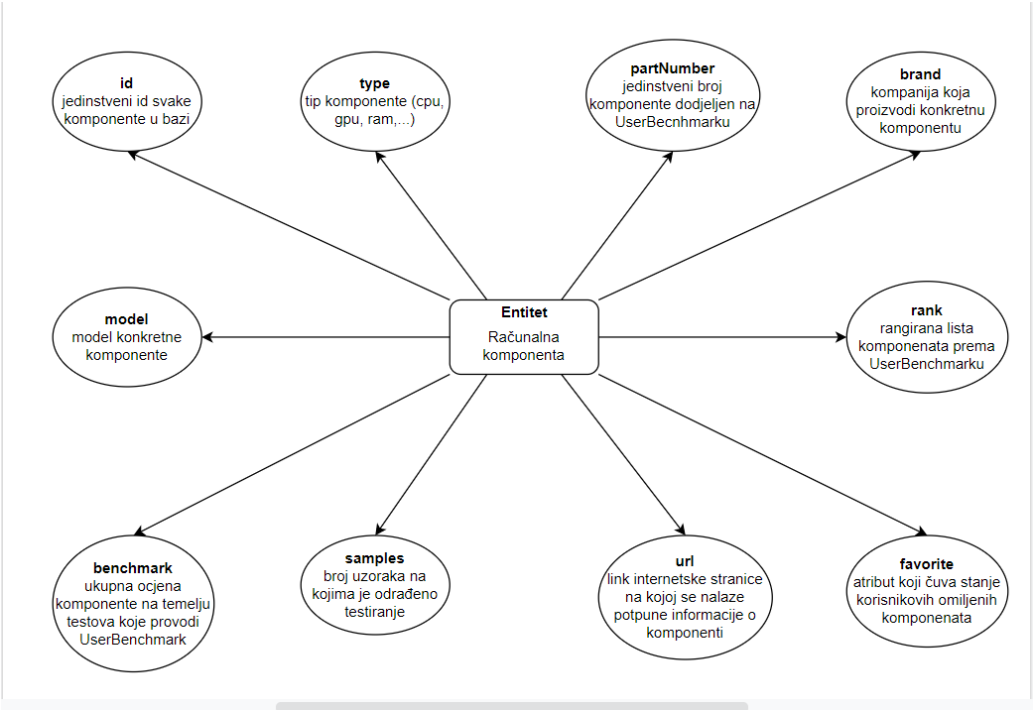
component_table

Refresh table Live updates

	id	type	partNumber	brand	model	rank	benchmark	samples	url	favorite
1	1	CPU	BX8070811900K	Intel	Core i9-11900K	1	109	2268	https://cpu.userbench	0
2	2	CPU	BX8070811700K	Intel	Core i7-11700K	2	107	4308	https://cpu.userbench	0
3	3	CPU	BX8070811900	Intel	Core i9-11900	3	106	227	https://cpu.userbench	0
4	4	CPU	BX8070811700KF	Intel	Core i7-11700KF	4	105	702	https://cpu.userbench	0
5	5	CPU	BX8070811900F	Intel	Core i9-11900F	5	104	217	https://cpu.userbench	0
6	6	CPU	BX8070811600K	Intel	Core i5-11600K	6	104	2145	https://cpu.userbench	0
7	7	CPU	BX8070811600KF	Intel	Core i5-11600KF	7	104	433	https://cpu.userbench	0
8	8	CPU	BX8070811900KF	Intel	Core i9-11900KF	8	103	423	https://cpu.userbench	0
9	9	CPU	BX8070811600	Intel	Core i5-11600	9	103	67	https://cpu.userbench	0
10	10	CPU	BX8070110900KF	Intel	Core i9-10900KF	10	102	10432	https://cpu.userbench	0
11	11	CPU	BX8070811700	Intel	Core i7-11700	11	102	953	https://cpu.userbench	0
12	12	CPU	BX8070110900K	Intel	Core i9-10900K	12	102	47216	https://cpu.userbench	0
13	13	CPU	BX8070811500	Intel	Core i5-11500	13	101	453	https://cpu.userbench	0
14	14	CPU	BX80684I99900KS	Intel	Core i9-9900KS	14	101	9406	https://cpu.userbench	0
15	15	CPU		Intel	Core i9-11980HK	15	101	4	https://cpu.userbench	0
16	16	CPU	BX8070110850K	Intel	Core i9-10850K	16	100	37915	https://cpu.userbench	0
17	17	CPU	100-100000059WOF	AMD	Ryzen 9 5950X	17	100	20291	https://cpu.userbench	0
18	18	CPU	100-100000061WOF	AMD	Ryzen 9 5900X	18	99.9	42008	https://cpu.userbench	0
19	19	CPU		Intel	Core i9-11900H	19	99.6	1	https://cpu.userbench	0

Slika 3.3. Popunjena Room baza podataka

Na slici 3.4. prikazana su sva polja korištene baze podataka. Budući da su u bazu spremane informacije o računalnim komponentama, baza je kreirana tako da postoji jedan entitet tj. Računalna komponenta sa svojim atributima koji ju opisuju kao što se može vidjeti na slici.



Slika 3.4. Prikaz svih polja baze podataka

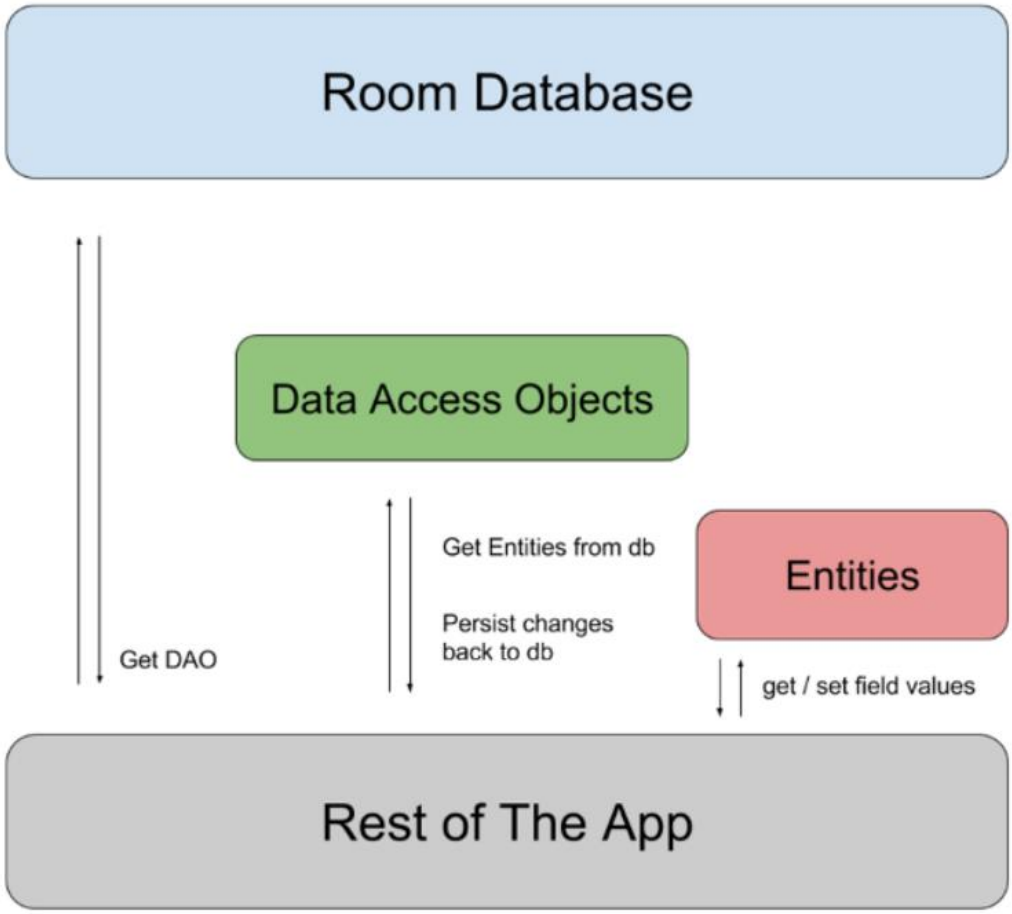
3.3. Korištene tehnologije

U tablici 3.1. moguće je vidjeti tehnologije koje su korištene pri razvijanju aplikacije.

Tablica 3.1. Prikaz korištenih tehnologija

TEHNOLOGIJA	**KRATAK OPIS**
Android studio	Razvojno okruženje korišteno u izradi aplikacije
Java	Programski jezik kojim je pisan programski kod aplikacije
JSON	Tekstualno bazirani format za razmjenu podataka
Room	Biblioteka koja omogućuje pristup bazi podataka
UserBenchmark Resources for Developers	Pristup podacima vezanim za računalne komponente

Aplikacija je razvijena u *Android studiu* koji je službeno integrirano razvojno okruženje koje bazu svoga rada temeljni na *InteliJ* razvojnom okruženju [7]. Programski kod aplikacije napisan je u Javi, jednom od najpopularnijih programskih jezika napravljenih 1995. godine [8]. Prikupljeni podaci o komponentama prikazani su u JSON [9] formatu koji služi za prikazivanje jednostavnih struktura podataka ili nizova te olakšava spremanje istih podataka u bazu. Room [10] biblioteka omogućuje vrlo lagan pristup bazi podataka preko *SQLite* apstrakcije čime zadržava punu snagu te baze podataka. Slika 3.5. prikazuje međusobni odnos 3 glavne komponente *Rooma*. *UserBenchmark Resources for Developers* [11], pruža izvrsne podatke o računalnim komponentama s više od 4000 komponenata u svojoj bazi podataka. Upravo iz tog razloga u ovom završnom radu koristit će se njihova baza podataka za procesore, grafičke kartice, ram memoriju, *solid state* diskove i tvrde diskove. Ostale komponente (napajanja, kućišta i matične ploče) bit će ručno nadopunjene podacima iz pouzdanih izvora budući da ih u bazi *UserBenchmarka* nema.



Slika 3.5. Prikaz komponenata *Rooma*, izvor: [10]

3.4. Implementacija rješenja

Odlomak u kojemu su predstavljene najbitnije funkcionalnosti aplikacije kroz programski kod i snimke zaslona aplikacije.

3.4.1. Baza podataka

Kao što je već ranije spomenuto, kako se ne bi svaki put iznova učitavala JSON datoteka s podacima o komponentama te kako bi se mogao spremiti status omiljenih komponenata, u aplikaciji će biti korištena Room baza podataka. U ranijem odlomku opisani su dijelovi Room baze podataka, a na slici 3.6. može se vidjeti konkretna implementacija baze u apstraktnoj klasi *ComponentDatabase*. Kako bi aplikacija znala da se radi o bazi podataka prvo jer potrebno iznad definicije same klase označiti anotacijom *@Database* te kao parametar predati klasu koja predstavlja entitet te baze podataka, u ovom slučaju *Component.class* jer ona sadrži sve atribute komponenata. Još se dodaje i verzija baze podataka koja je 1. Bitno je primijetiti kako apstraktna klasa *ComponentDatabase* proširuje biblioteku *RoomDatabase* te na taj način ona zapravo poprima funkcionalnosti baze podataka. Također je bitno uočiti da se unutar same baze podataka inicijalizira sučelje *ComponentDao* jer ono služi kao pristupna točka operacijama nad bazom te se u tom sučelju propisuju upiti koji će se koristiti za operacije nad bazom. Metoda *getInstance()* vraća objekt baze podataka tako što prvo provjeri postoji li već neka baza podataka, a zatim kroz konstruktor propisan Room bibliotekom pravi novu bazu podataka pod nazivom *component_database*.

```java
@Database(entities = {Component.class}, version = 1)
public abstract class ComponentDatabase extends RoomDatabase {

    private static ComponentDatabase instance;
    private static Context activity;

    public abstract ComponentDao componentDao();

    public static synchronized ComponentDatabase getInstance(Context context){

        activity=context.getApplicationContext();

        if (instance==null){
            instance= Room.databaseBuilder(context.getApplicationContext(),
                    ComponentDatabase.class, name: "component_databse")
                    .fallbackToDestructiveMigration()
                    .allowMainThreadQueries()
                    .addCallback(roomCallback)
                    .build();
        }
        return instance;
    }
```

Slika 3.6. Apstraktna klasa *ComponentDatabase*

Na slici 3.7. prikazan je izgled sučelja *ComponentDao* kao i sve metode koje se koriste za manipulaciju podacima u bazi podataka.

```java
@Dao
public interface ComponentDao {

    @Query("SELECT*FROM component_table ORDER BY brand ASC" )
    LiveData<List<Component>> getAllComponent();

    @Transaction
    @Query("UPDATE component_table SET favorite='1' WHERE brand=:brand AND model=:model")
    void makeFavorite(String brand, String model);

    @Transaction
    @Query("UPDATE component_table SET favorite='0' WHERE brand=:brand AND model=:model")
    void removeFavorite(String brand, String model);

    @Query("SELECT*FROM component_table WHERE favorite='1'")
    List<Component> favoriteComponents();

    @Insert
    void insert(Component component);
    @Delete
    void delete(Component component);
}
```

Slika 3.7. Sučelje *ComponentDao*

Ispred definicije samog sučelja postavlja se anotacija *@Dao* kako bi aplikacija znala da se radi o *Data Access Objectu*. Ovdje se propisuju metode, tj. SQL upiti za operacije nad bazom podataka. Upit *getAllComponent()* jednostavno vraća sve objekte tipa *Component* i sortira ih prema brendu proizvođača. Ovaj upit koristi se kod popunjavanja baze podataka svima informacijama o komponentama. Upiti *makeFavorite()* i *removeFavorite()* su dva suprotna upita. Dok *makeFavorite()* određenoj komponenti atribut *favorite* postavlja na vrijednost 1, *removeFavorite()* taj isti atribut postavlja na vrijednost 0. Upravo na ovaj način se komponente označuju kao omiljene komponente kada ih korisnik poželi dodati na listu favorita. Upit *favoriteComponents()* se koristi za popunjavanje liste omiljenih komponenata zato što ovaj upit vraća sve komponente koje kao atribut *favorite* imaju postavljene u vrijednost 1. Na kraju su definirana dva standardna upita za dodavanje i brisanje komponenata *insert()* i *delete()*.

```
private  static JSONArray loadJSONArray(Context context){
    StringBuilder builder= new StringBuilder();
    InputStream in = context.getResources().openRawResource(R.raw.components);
    BufferedReader reader= new BufferedReader(new InputStreamReader(in));

    String line;

    try {
        while ((line=reader.readLine()) !=null){
            builder.append(line);
        }

        JSONObject json= new JSONObject(builder.toString());
        return json.getJSONArray( name: "components");
    }catch (IOException | JSONException exception){
        exception.printStackTrace();
    }
    return null;
}
```

Slika 3.8. Učitavanje JSON datoteke

Na slici 3.8. prikazana je metoda *loadJSONArray()* kojom se učitava JSON datoteka koja je prethodno ubačena u *raw* resurse Android studia. *InputStream* napuni se resursom pod imenom *components* koja predstavlja naziv JSON datoteke, a zatim se napravi novi objekt tipa *BufferedReader* te se njemu predaje varijabla *in*. Dok god *reader* ne dođe do kraja datoteke on čita podatke iz predanog resursa. Zatim se stvara novi *JSONObject* kojemu se predaje gotovi *string* dobiven iz *readera*. Sve ove operacije bile su potrebne kako bi od tekstualnog formata JSON

datoteke bilo stvoreno polje JSON objekata razumljivih aplikaciji te se sada ti objekti mogu dalje koristiti bez poteškoća.

```
private static void fillWithStartingData(Context context){
    ComponentDao dao= getInstance(context).componentDao();

    JSONArray components = loadJSONArray(context);

    try {
        for(int i=0;i<components.length();i++){
            JSONObject component= components.getJSONObject(i);

            String componentType=component.getString( name: "Type");
            String componentPartNumber=component.getString( name: "Part Number");
            String componentBrand=component.getString( name: "Brand");
            String componentModel=component.getString( name: "Model");
            String componentRank=component.getString( name: "Rank");
            String componentBenchmark=component.getString( name: "Benchmark");
            String componentSamples=component.getString( name: "Samples");
            String componentURL=component.getString( name: "URL");

            dao.insert(new Component(componentType, componentPartNumber, componentBrand, componentModel, componentRank, componentBenchmark, componentSamples, componentURL));
        }

    } catch (JSONException e){

    }
}
```

Slika 3.9. Popunjavanje baze podataka

Slika 3.9. prikazuje metodu *fillWithStartingData()* kojom se konačno popunjava baza podataka ranije učitanim JSON objektima. Prvo se dohvaća instanca baze podataka preko sučelja *ComponentDao* te se također popunjava polje JSON objekata *components* putem ranije opisane metode *LoadJSONArray()*. Sada se za svaki JSON objekt u polju čitaju njegovi atributi i postavlja ih se kao vrijednosti atributa iz tablice u bazi podataka. Na kraju se kreira nova komponenta *Component* putem parametarskog konstruktora kojemu se predaju ranije učitani atributi te se onda ta komponenta ubacuje u bazu podataka putem *insert()* metode. Na ovaj način prošlo se cijelo polje JSON objekata i od njih su kreirane zasebne komponente koje se sada nalaze u bazi podataka sa svim potrebnim podacima.

3.4.2. Modul za prikazivanje

Budući da je sada baza podataka popunjena svim bitnim informacijama o komponentama, vrijeme je da ih se prikaže u aplikaciji. Nakon što korisnik odabere željenu vrstu komponente iz početne aktivnosti aplikacije prebacuje ga se na aktivnost sa prikazom svih komponenata izabrane vrste. Sve komponente prikazane su u dinamičkoj listi *RecyclerView*. Glavan prednost ovog prikaza jest mogućnost recikliranja svojih elementa pa se na taj način znatno povećava brzina prikaza komponenata. Potrebne su dvije klase za ostvarenje RecyclerViewa i jedna klasa u kojoj se dohvaća sami *RecyclerView*, a u ovom slučaju to su: *ComponentsActivity*, *ComponentViewHolder*

i *ComponentAdapter*. U klasi *ComponentsActivity* dohvaća se ranije kreirani *RecyclerView* u XML-u preko id-a i kasnije se postavljaju podaci u njega kao što je prikazano na slici 3.10.

```
public class ComponentsActivity extends AppCompatActivity implements JumpActivity {

    private ComponentViewModel componentViewModel;
    private String type;
    private ComponentAdapter adapter;
    private List<Component> lista;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_components);



        RecyclerView recyclerView=findViewById(R.id.recyclerComponents);
        recyclerView.setLayoutManager(new LinearLayoutManager( context: this));
        adapter= new ComponentAdapter( clickActivity: this);
        recyclerView.setAdapter(adapter);
```

Slika 3.10. Postavljanje *RecyclerViewa*

Klasa *ComponentViewHolder* predstavlja element *RecyclerViewa*. U ovom dijelu bitno je poslati *View* koji predstavlja ćeliju *RecyclerViewa*. Također, na *TextView* postavlja se metoda *setOnClickListener()* kako bi se prilikom klika na ćeliju mogla izvršiti određena operacija. U ovom slučaju, ta operacija je prebacivanje na novu aktivnost uz slanje određenih podataka o komponenti. Sve ovo opisano može se vidjeti na slici 3.11. koja prikazuje kod *ComponentViewHolder* klase.

```
public class ComponentViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener{
    private TextView tvName;
    JumpActivity activity;
    public ComponentViewHolder(@NonNull View itemView, JumpActivity activity) {
        super(itemView);
        this.activity=activity;
        tvName=itemView.findViewById(R.id.tvComponent);
        tvName.setOnClickListener(this::onClick);
    }

    public void setName(String name) { tvName.setText(name); }

    @Override
    public void onClick(View v) { activity.jumpOnTextClick(getAdapterPosition()); }
}
```

Slika 3.11. ComponentViewHolder klasa

Klasa *ComponentAdapter* isto tako drži podatke koji se prikazuju u RecyclerViewu. Kako bi adapter funkcionirao, potrebno je implementirati tri metode kao što je prikazano na slici 3.12.

```
@NonNull
@Override
public ComponentViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    View itemView= LayoutInflater.from(parent.getContext()).inflate(R.layout.list_item,parent, attachToRoot: false);
    return new ComponentViewHolder(itemView, activity);
}

@Override
public void onBindViewHolder(@NonNull ComponentViewHolder holder, int position) {
    Component currentComponent=components.get(position);
    holder.setName(currentComponent.getBrand()+" "+currentComponent.getModel());
}

@Override
public int getItemCount() { return components.size(); }
```

Slika 3.12. ComponentAdapter

Metoda *onCreateViewHolder()* služi za kreiranje *ComponentViewHoldera*. Metoda *onBindViewHolder()* povezuje upravo kreirani *ComponentViewHolder* i podatke koji su ubačeni u *RecyclerView*, a metoda *getItemCount()* jednostavno vraća broj elemenata koji se nalaze u listi.

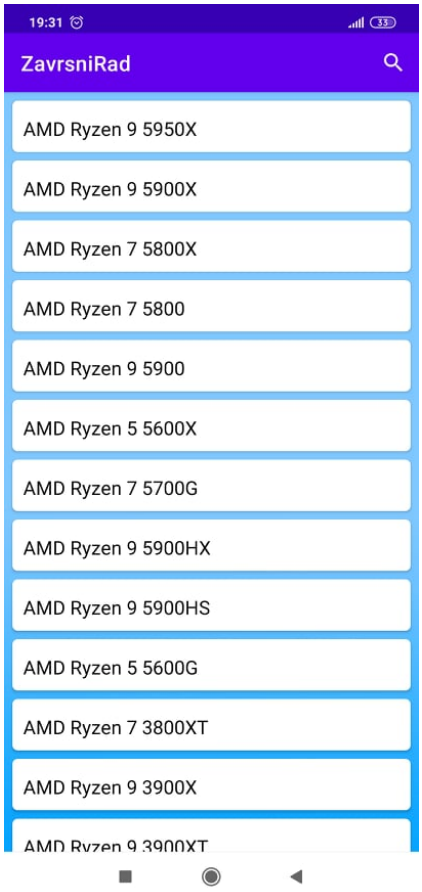
Nakon što su ove 3 klase uspješno usklađene, mogu se ubaciti komponente u *RecyclerView*. To se događa u klasi *ComponentsActivity* kao što je prikazano na slici 3.13.

```
componentViewModel= ViewModelProviders.of( activity: this).get(ComponentViewModel.class);
componentViewModel.getAllComponents().observe( owner: this, new Observer<List<Component>>() {
    @Override
    public void onChanged(List<Component> components) {
        List<Component> componentsByType= new ArrayList<>();
        for(int i=0;i<components.size();i++){
            if (components.get(i).getType().equals(type)){
                Component component=new Component(components.get(i).getType(),components.get(i).getPartNumber(),components.get(i).getBrand(),
                                                  components.get(i).getModel(),components.get(i).getRank(),components.get(i).getBenchmark(),
                                                  components.get(i).getSamples(),components.get(i).getUrl());
                componentsByType.add(component);
            }
        }
        lista=new ArrayList<>(componentsByType);
        adapter.setComponents(componentsByType);
        adapter= new ComponentAdapter(lista,ComponentsActivity.this::jumpOnTextClick);
        recyclerView.setAdapter(adapter);
    }
});
```

Slika 3.13. Ubacivanje komponenata u *RecyclerView*

Prvi korak je postavljanje promatrača na listu komponenata. Prilikom promjene u listi okida se metoda *onChanged()* kojoj se predaje trenutna lista komponenata popunjena svim komponentama iz baze podataka putem metode *getAllComponents()* koja je ranije objašnjena. Zatim se kreira nova pomoćna lista *componentsByType* u koju se spremaju odgovarajuće komponente ovisno o

korisnikovom odabiru. Bitno je napomenuti kako prije ove aktivnosti korisnik odabire vrstu komponente klikom na nju te ga aplikacije prebacuje na ovu aktivnost. Prilikom prelaska između aktivnosti tip komponente se pamti. For petljom prolazi se kroz cijelu listu komponenata i uspoređuje se tip komponente s onim kojega je korisnik odabrao. Ako dođe do podudaranja u tipu komponente tada se kreira nova komponenta s atributima na tom mjestu gdje je došlo do podudaranja te se ta nova komponenta dodaje na pomoćnu listu. Nakon što se prođe kroz cijelu petlju i pronađu se sve odgovarajuće komponente tada tu listu predajemo adapteru koji nadalje prikazuje predane podatke u *RecyclerViewu*.



Slika 3.14. Prikaz zaslona aplikacije

Na slici 3.13. vidi se konkretan rezultat primjene ranije opisanih dijelova programskoga koda. Baza podataka popunjena je podacima o komponentama. Ti podaci koriste se za popunjavanje *RecyclerViewa* i tako korisnik dobiva prikaz sa slike 3.14.

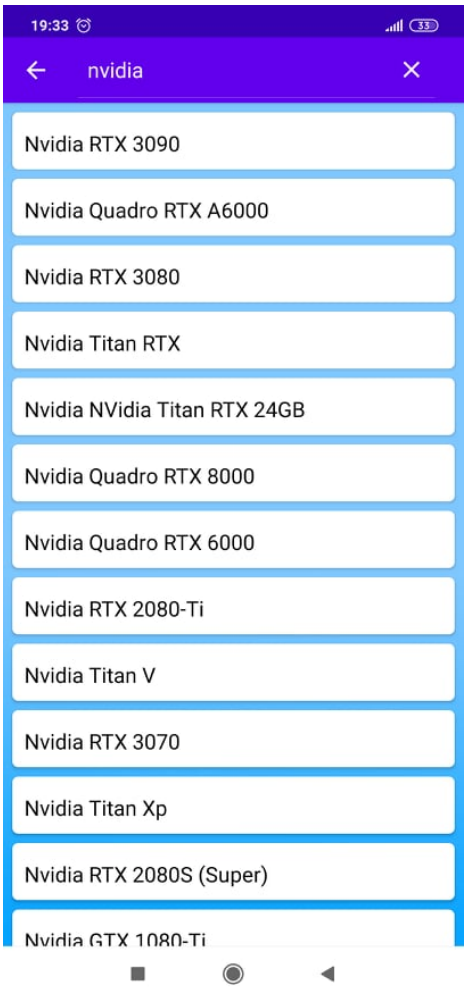
3.4.3. Modul za pretraživanje

Budući da aplikacija zahtjeva rad s velikim brojem komponenata iz praktičnog razloga napravljen je sustav za pretragu komponenata kako korisnik ne bi morao ručno prolaziti kroz veliki popis. Dodavanjem filtera implementira se metoda *performFiltering()* kojoj se predaje znakovni niz pomoću kojega se filtriraju same komponente. Kreira se pomoćna lista *filteredList* u koju se dodaju filtrirani rezultati pretrage. Prvo predani znakovni niz pretvara se u String, prebacuje u mala slova te se ukljanjaju nepotrebne praznine kako bi filter za pretragu što bolje funkcionirao. Prolazi se kroz cijelu listu komponenata, a zatim jednostavnim *if* uvjetom provjerava se dolazi li do podudaranja u modelu ili brendu komponente sa zadanim filterom. Ako je to istina, ta komponenta se dodaje na pomoćnu listu. Kada se prođe kroz cijelu listu komponenata, pomoćna lista sa rezultatima pretrage predaje se adapteru i on sada umjesto svih komponenata prikazuje samo one koje zadovoljavaju filter pretrage. Opisani postupak može se vidjeti i na slici 3.15.

```
@Override
public Filter getFilter() { return searchFilter; }

private Filter searchFilter = new Filter() {
    @Override
    protected FilterResults performFiltering(CharSequence constraint) {
        List<Component> filteredList=new ArrayList<>();

        if (constraint==null || constraint.length()==0){
            filteredList.addAll(componentsFull);
        }
        else {
            String filterPattern=constraint.toString().toLowerCase().trim();

            for (Component item : componentsFull){
                if (item.getBrand().toLowerCase().contains(filterPattern) || item.getModel().toLowerCase().contains(filterPattern)){
                    filteredList.add(item);
                }
            }
        }
        FilterResults results= new FilterResults();
        results.values=filteredList;

        return results;
    }

    @Override
    protected void publishResults(CharSequence constraint, FilterResults results) {
        components.clear();
        components.addAll((List)results.values);
        notifyDataSetChanged();
    }
};
```

Slika 3.15. Implementacija filtera pretrage

Slika 3.16. prikazuje konkretan primjer primjene filtera pretrage. Pretražuje se pojam „nvidia“ među grafičkim kraticama i kao što je vidljivo na slici, prikazane su samo one grafičke kartice koje taj pojam sadrže u svom imenu.



Slika 3.16. Prikaz zaslona aplikacije

3.4.4. Prikupljanje podataka

U aktivnosti *SpecificComponentActivity* osim generalnih informacija o odabranoj komponenti, korisnik ima izbor dva gumba koji ga vode na stranice jeftinije.hr i nabava.net s pretragom cijene te iste komponente. Budući da link pretrage na obje ove stranice ima fiksni dio, na vrlo jednostavan način može se obaviti pretraga željene komponente direktno iz aplikacije tako što se kombinira taj fiksni dio linka s modelom i brendom komponente. Na oba gumba postavlja se metoda setOnClickListener() koja prilikom pritiska na gumb prebacuje korisnika na odgovarajuću stranicu. Prikaz ovog dijela može se vidjeti na slikama 3.17. i 3.18.

```
jeftinijeURL="https://www.jeftinije.hr/Trazenje/Proizvodi?q="+proizvod;
nabavaURL="https://www.nabava.net/search.php?q="+proizvod+"&cod=&cdo=";

btnJeftinije.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) { goToUrl(jeftinijeURL); }
});

btnNabava.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) { goToUrl(nabavaURL); }
});
```

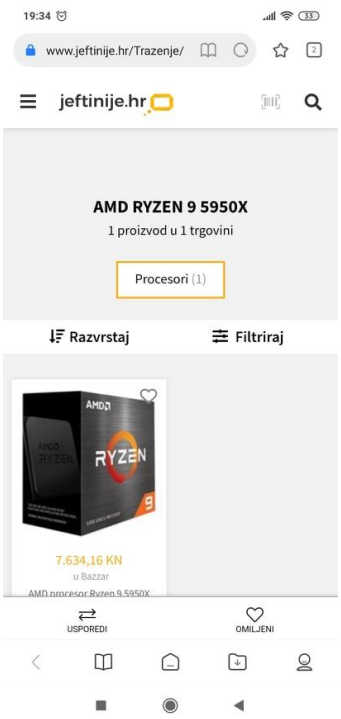
Slika 3.17. Prikaz koda za pretragu na stranicama

```
private void goToUrl(String s) {
    Uri uri=Uri.parse(s);
    startActivity(new Intent(Intent.ACTION_VIEW,uri));
}
```

Slika 3.18. Metoda *goToUrl()*

Slike 3.19. i 3.20. prikazuju rad navedenog programskog koga. Na slici 3.19. vidi se aktivnost odabrane specifične komponente, a prilikom pritiska na gumb jeftinije.hr korisnika se prebacuje na sliku 3.20. gdje se vidi ispravna pretraga na stranici jeftinije.hr za željenu komponentu.

Slika 3.19. Prikaz zaslona aplikacije



Slika 3.20. Prikaz zaslona aplikacije

3.4.5. Popis komponenata

Kao što je već rečeno, korisnik može spremiti komponente koje mu se svide te ih naknadno pogledati. U aktivnosti *FavoritiActivity* poziva se instanca baze podataka i popunjava se lista *favoriteComponents* tako što se preko DAO-a pozove metoda *favoriteComponents()* koja iz baze podataka vraća komponente spremljene u favorite. Nakon toga sve što je potrebno napraviti je dodati svaku komponentu s te liste u adapter *RecyclerView*-a kako bi ih on ispravno mogao prikazati. Prikaz koda moguće je vidjeti na slici 3.21.

```
favoriteComponents=db.componentDao().favoriteComponents();

for (int i=0;i<favoriteComponents.size();i++){
    Component component= new Component(favoriteComponents.get(i).getType(),favoriteComponents.get(i).getPartNumber(),favoriteComponents.get(i)
            .getBrand(),favoriteComponents.get(i).getModel(),favoriteComponents.get(i).getRank(),favoriteComponents.get(i).getBenchmark(),
            favoriteComponents.get(i).getSamples(),favoriteComponents.get(i).getUrl());
    adapter.addNewCell(component,adapter.getItemCount());
}
```

Slika 3.21. Lista omiljenih komponenata

Slika 3.22. pruža prikaz liste omiljenih komponenata u aplikaciji. U svrhe što boljeg prikaza same liste, dodane su 3 komponente u listu.

Slika 3.22. Prikaz zaslona aplikacije

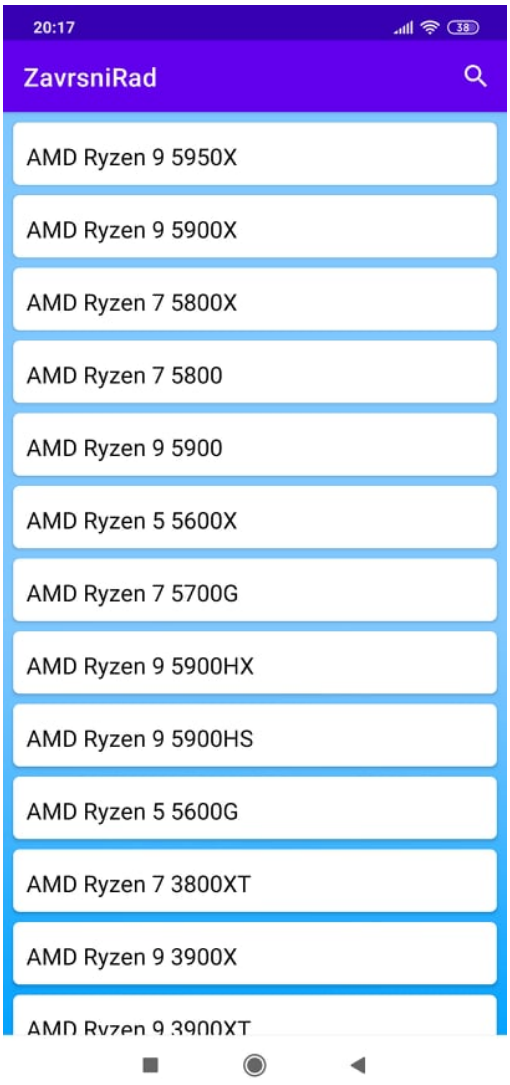
4. ISPTIVANJE FUNKCIONALNOSTI

U posljednjem poglavlju ovog rada prikazan je praktičan način rada aplikacije, ispitan je rad aplikacije na 2 različita slučaja te su opisane moguće nadogradnje same aplikacije.

4.1. Opis načina korištenja aplikacije

Pri ulasku u aplikaciju korisniku se na izbor prikazuju tipovi komponenata i gumb koji vodi na listu omiljenih komponenata kao što je prikazano na slici 4.1.

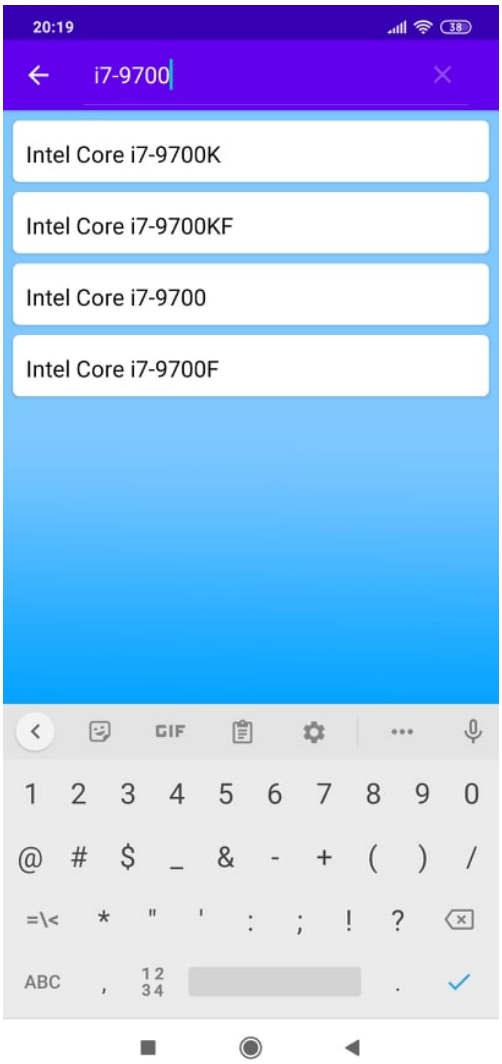
Slika 4.1. Početni zaslon



Slika 4.2. Aktivnost s komponentama

Nakon što korisnik odabere željeni tip odvodi ga se na novu aktivnost popunjenu odgovarajućim komponentama. Na slici 4.2 vidi se ta aktivnost te popis komponenata nakon što korisnik odabere procesori kao tip željene komponente.

Nadalje, korisnik odabire konkretan procesor koji želi. To može napravit prolaskom kroz listu ili pretragom pomoću *Search bara* kao što je prikazano na slici 4.3.



Slika 4.3. Prikaz Search bara

Slika 4.4. Prikaz aktivnosti s detaljima o komponenti

Nakon odabira konkretne komponente, korisnika se prebacuje na novu aktivnost koja sadrži detalje o komponenti (slika 4.4.). Također, ukoliko korisnik želi proučiti potpune specifikacije određene komponente, to može učiniti pritiskom na ponuđeni link.

Korisniku se na izbor također nudi pregled ponude za konkretnu komponentu na domaćem tržištu. Pritiskom na jedan od 2 gumba odvodi ga se na odgovarajuću stranicu sa pretragom za odabranu komponentu kao što se može vidjeti na slici 4.5.

Slika 4.5. Jeftinije.hr

Slika 4.6. Prikaz aktivnosti s listom omiljenih komponenata

Ako se korisniku određena komponenta svidi i poželi ju spremiti za kasnije to može učiniti pritiskom na gumb dodaj u favorite. Na slici 4.6. može se vidjeti izgled aktivnosti koja sadrži listu omiljenih komponenata nakon što je dodano nekoliko komponenata u nju. Ako se pritisne na ćeliju neke od ovih komponenata aplikacija ponovno odvodi korisnika na aktivnost sa detaljima o toj komponenti. Također, korisnik u svakom trenutku može maknuti komponentu s ove liste pritiskom na x.

4.2. Testiranje rada aplikacije

Za testiranje ove aplikacije promatrat će se ukupna cijena komponenata nakon odabira u aplikaciji i usporedit će ju se s cijenama tih istih komponenata u jednoj od najpoznatijih trgovina računalnom opremom, Links.hr. Bit će definirane dvije računalne konfiguracije te će se provjeriti efikasnost aplikacije za svaku od njih. Očekivani rezultat za oba slučaja su niže ukupne cijene komponenata odabirom iz aplikacije.

4.2.1. Prvi slučaj

Za prvi slučaj testiranja odabire se konfiguracija računala s Intel-ovim procesorom. Zadane komponente računala su:

- Procesor: Intel Core i5-10600K
- Grafička kartica: GeForce RTX 2060
- RAM memorija: Corsair Vengeance 2x8GB DDR4
- Napajanje: Corsair CX650
- SSD: Kingston A400 240 GB
- Matična ploča: ASUS ROG Maximus XII

U trgovini Links.hr cijene zadanih komponenata su sljedeće: procesor 1804,05 kn, grafička kartica 4199,00 kn, RAM memorija 854,05 kn, napajanje 711,55 kn, ssd 255,55 kn, matična ploča 3989,05 kn. Ukupna cijena svih komponenata tako iznosi 11.813,25 kn.

Ako se iste te komponente pronađu koristeći aplikaciju dobivaju se sljedeće cijene: procesor 1622,88 kn (nabava.net – iponcomp.hr), grafička kartica 4199,00 kn (jeftinije.hr – Links.hr), RAM memorija 756,00 kn (nabava.net - uzishop.hr), napajanje 668,00 kn (jeftinije.hr - thm.hr), ssd 213,13 kn (jeftinije.hr – bazzar.hr), matična ploča 3298,76 kn (nabava.net – megabajt.hr). Ukupna cijena svih komponenata tako iznosi 10.757,77 kn

4.2.2. Drugi slučaj

Za drugi slučaj testiranja promatra se konfiguracija s AMD-ovim procesorom. U ovom slučaju, zadane komponente su sljedeće:

- Procesor: AMD Ryzen 5 5600X
- Grafička kartica: Radeon RX 6700XT

- RAM memorija: G.SKILL Trident Z 2x8GB DDR4
- Napajanje: Corsair CX650
- SSD: Crucial MX500 1TB
- Matična ploča: ASUS TUF B550M-PLUS

U trgovini Links.hr cijene zadanih komponenata su sljedeće: procesor 2469,05 kn, grafička kartica 6649,00 kn, RAM memorija 949,00 kn, napajanje 711,55 kn, ssd 892,05 kn, matična ploča 1310,05 kn. Ukupna cijena svih komponenata tada iznosi 12.980,70 kn.

Ako se iste te komponente pronađu koristeći aplikaciju dobivaju se sljedeće cijene: procesor 2260,00 kn (nabava.net – uzishop.hr), grafička kartica 6199,00 kn (nabava.net – adm.hr), RAM memorija 879,00 kn (jeftinije.hr - mall.hr), napajanje 668,00 kn (jeftinije.hr - thm.hr), ssd 736,59 kn (nabava.net – iponcomp.hr), matična ploča 1139,00 kn (jeftinije.hr - thm.hr). Ukupna cijena svih komponenata tada iznosi 11.881,59 kn.

4.2.3. Interpretacija rezultata testiranja

Bitno je napomenuti da su prikazane cijene komponenata izračunate na dan pisanja ovog dijela rada te je moguće da će se one u budućnosti promijeniti. Vidljivo je da je korisnik korištenjem aplikacije uštedio čak 1055,48 kn u prvom slučaju, dok je u drugom slučaju uštedio 1099,11 kn. U oba slučaja korisnik je uštedio približno 10% ukupne cijene, što svakako nije mala stvar. Ovim rezultatima dokazana je ispravnost rada aplikacije ali isto tako i njezina efikasnost te isplativost.

5. ZAKLJUČAK

U posljednje vrijeme raste zainteresiranost ljudi za samostalnom izradom osobnog računala. Sve više ljudi pomoć traži na internetu po raznoraznim forumima i stranicama za pomoć. Nakon što se napokon dovoljno informiraju i krenu u plan izrade svoga računala, javlja se problem gdje pronaći i kupiti potrebne komponente. Također u pitanje dolazi i dostupnost te cijena tih istih komponenata. Cilj ovog završnog rada bio je olakšati upravo te probleme, a rješenje je pronađeno u obliku mobilne aplikacije za Android platformu.

Nakon analize već postojećih programskih rješenja i gore navedenih problema osmišljeno je rješenje te su postavljeni funkcionalni zahtjevi koje aplikacija mora zadovoljiti. Prije izrade same aplikacije izrađen je dijagram tijeka te aplikacije koji je uvelike pomogao u rješavanju svih zahtjeva te u snalažljivosti kroz aplikaciju.

Aplikacija je realizirana u programskom okruženju Android studio te je cijela napisana u programskom jeziku Java. Nakon proučene teorijske podloge odlučene su programske tehnologije koje će biti korištene kako bi olakšale izradu same aplikacije. Zbog potreba čuvanja podataka u aplikaciji, korištena je Room baza podataka. Baza je popunjena informacijama o komponentama preuzetim sa stranice *Userbenchmark* koja nudi besplatne resurse za razvojne programere.

Konačna aplikacija korisniku nudi uvid u ponude računalnih komponenata na domaćem tržištu. Na brz i lak način korisnik može odabrati i sačuvati željene komponente te im kasnije pristupiti i provjeriti cijenu i dostupnost.

Ispitivanjem rada aplikacije i testiranjem na realnim slučajevima dokazano je kako korisnik korištenjem ove aplikacije lako dolazi do saznanja o trenutnim najboljim ponudama za određenu računalnu komponentu. Upravo tako korisnik štedi i vrijeme i novac. Na testiranim slučajevima dokazana je ne samo isplativost aplikacije u smislu uštede novca, već i njezina efikasnost.

Kao što je već ranije spomenuto, dio podataka o komponentama je preuzet iz baze podataka *UserBenchmark Resources for Developers* a dio je ručno napisan iz raznih izvora. Razlog tome je nedostatak besplatnog API-ja koji sadržava veći broj podataka o svim komponentama na jednom mjestu. Ukoliko bi se u budućnosti pojavio jedan takav API vrlo lako bi bilo zamijeniti stare podatke iz baze s novim, poboljšanim podacima te bi to bila izvrsna nadogradnja za ovu aplikaciju.

Jedno od mogućih proširenja aplikacije bilo bi dodavanje nove funkcionalnosti koja bi omogućila korisniku sortiranje komponenata po cijeni ili dobavljaču. Za ovo proširenje bilo bi potrebno

metodom *web scrapinga* doći do traženih podataka o cijeni i dostavljaču za svaku pojedinačnu komponentu u bazi.

LITERATURA

[1] PC Builder Part picker & Generate Builds, https://pc-builder-codewaster.en.aptoide.com/app [5.7.2021.]

[2] Upgrade PC - build or upgrade your computer, https://play.google.com/store/apps/details?id=com.futuredevelop.podborpk&hl=hr&gl=US [5.7.2021.]

[3] About PCPartPicker, https://pcpartpicker.com/about/ [5.7.2021.]

[4] O jeftinije.hr, https://www.jeftinije.hr/predstavljanje [5.7.2021.]

[5] How to Assemble a desktop PC, https://en.wikibooks.org/wiki/How_To_Assemble_A_Desktop_PC/Choosing_the_parts [5.7.2021.]

[6] The Ultimate Compatibility Guide, https://www.youtube.com/watch?v=AiVWQthb-20&ab_channel=Bitwit [5.7.2021.]

[7] Uvod u Android Studio, https://developer.android.com/studio/intro

[8.7.2021.]

[8] Java, https://www.w3schools.com/java/java_intro.asp [8.7.2021.]

[9] JSON, https://www.json.org/json-en.html [8.7.2021.]

[10] Room Database, https://developer.android.com/training/data-storage/room [9.7.2021.]

[11] UserBenchmark Developer Resources, https://www.userbenchmark.com/page/developer [9.7.2021.]

SAŽETAK

U Ovom radu bilo je potrebno osmisliti i implementirati vlastiti sustav za pomoć korisniku u odabiru računalnih komponenata. Budući da je danas uvelike zastupljeno korištenje Android pametnih uređaja, aplikacija je upravo iz tog razloga razvijena za tu platformu u programskom okruženju Android studio. Aplikacija korisniku pruža veliki popis računalnih komponenata, preuzetih s interneta, koje može odabrati i dobiti uvid u cijenu i dostupnost istih na domaćem tržištu. Aplikacija je zadovoljila sve postavljene funkcionalne zahtjeve, a testiranjem rada dokazana je njezina učinkovitost, isplativost i efikasnost.

Ključne riječi: baza podataka, mobilna Android aplikacija, računalne komponente, sustav za pomoć

ABSTRACT

SYSTEM FOR ASSISTANCE IN SELECTION OF COMPUTER COMPONENTS

In this paper, it was necessary to design and implement our own system to help the user in choosing computer components. Since the use of Android smart devices is widely represented today, the application was developed for this exact platform in the Android studio integrated development environment for this very reason. The application provides the user with a large list of computer components, downloaded from the Internet, which can then be selected and give user an insight into the price and availability of those same components in the domestic market. The application met all the set functional requirements, and by testing the work, its cost-effectiveness and efficiency were proven.

Key words: database, mobile Android application, computer components, help system

ŽIVOTOPIS

Domagoj Dragić rođen je 4.12.1999. godine u Đakovu. Završio je Osnovnu školu Vladimir Nazor u Đakovu i opću gimnaziju Antun Gustav Matoš u Đakovu. Po završetku gimnazije upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. Poznaje engleski i njemački jezik, a od programskih jezika najbolje poznaje Javu. Najviše voli izrađivati mobilne Android aplikacije.