Mobilna Android aplikacija za potporu prilagodljivoj pripremi hrane

Medak, Zvonimir

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni preddiplomski studij računarstva

MOBILNA ANDROID APLIKACIJA ZA POTPORU PRILAGODLJIVOJ PRIPREMI HRANE

Završni rad

Zvonimir Medak

Osijek, 2020.

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

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

Osijek, 18.07.2020.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime studenta:	Zvonimir Medak
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4100, 17.09.2019.
OIB studenta:	46411366911
Mentor:	Prof.dr.sc. Goran Martinović
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Mobilna Android aplikacija za potporu prilagodljivoj pripremi hrane
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:	18.07.2020.
Datum potvrde ocjene Odbora:	09.09.2020.

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, 15.09.2020.

Ime i prezime studenta:	Zvonimir Medak
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4100, 17.09.2019.
Turnitin podudaranje [%]:	6

Ovom izjavom izjavljujem da je rad pod nazivom**: Mobilna Android aplikacija za potporu prilagodljivoj pripremi hrane**

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

i sumentora

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

Potpis studenta:

SADRŽAJ

1. UVOD

Tema ovoga završnog rada je izrada mobilne Android aplikacije koja će služiti kao pomoć pri odabiru recepata primarno za osobe koje imaju netolerancije na neke oblike hrane ili namirnice, ali će podržavati i neka ograničenja koja osobe imaju. Osobe s netolerancijama na neke oblike hrane ili namirnice nailaze na svakodnevne probleme pri pripremi hrane. Osobe u takvim situacijama znaju koje im namirnice nisu pogodne, ali moraju pretraživati cijeli recept da provjere sadrži li recept određenu namirnicu. Cilj ovoga završnog rada je dati mogućnost osobama u takvim situacijama, ako posjeduju Android pametni mobitel, da na lakši način mogu dobiti recepte koji njima odgovaraju obzirom na njihove netolerancije i ograničenja.

Da bi se teorijska pozadina teme uspjela ostvariti, u sklopu izrade rada bilo je potrebno proučiti koje se netolerancije često javljaju te proučiti i opisati zašto se događaju. Potrebna je teorijska podloga za mobilnu implementaciju, jer mora posjedovati parametre po kojima će pratiti je li recept prikladan za danu osobu. Pri tome, bilo je potrebno voditi se medicinskim i nutricionističkim načelima koja govore o netolerancijama ljudi na pojedinu hranu i razne namirnice.

Za implementaciju teme na mobilne Android uređaje potrebno je poznavanje razvojnog okruženja Android Studio koje podržava nekoliko objektno-orijentiranih jezika, ali je u radu korišten Kotlin za razradu aplikacije. Također, potrebno je poznavanje i jezika SQLite koji se koristio u sklopu Room Database-a za izradu baze podataka recepata i XML-a koji se koristi za opisivanje izgleda zaslona.

U drugom poglavlju rada opisan je teorijski dio u kojem su prikazani izazovi problema netolerancije hrane. Ideja i građa mobilne aplikacije opisana je u trećem poglavlju završnog rada, a implementacija programskog koda u četvrtom poglavlju. Peto poglavlje daje pregled načina korištenja aplikacije i testiranje.

1.1. Zadatak završnog rada

U radu je potrebno proučiti i opisati izazove postupaka pripreme hrane prilagodljive različitim zahtjevima i ograničenjima korisnika, te mogućnosti potpore mobilnih aplikacija tim postupcima. Vodeći se medicinskim i nutricionističkim preporukama, treba razviti odgovarajući model jelovnika, namirnica i recepata za pripremu hrane. Također, treba razviti model izbora jelovnika, namirnica i recepata za pripremu hrane. Također treba razviti model mobilne aplikacije s bazom podataka koja će omogućiti unos željenih parametara i ograničenja korisnika, pohranu i analizu unesenih podataka i

zahtjeva, stvaranje preporuka za izbor prikladne hrane, ponudu recepata za njenu pripremu, te unos, analizu i ispis dojmova korisnika. Također, treba opisati programatske tehnologije i razvojnu okolinu za razvoj mobilne aplikacije. U praktičnom dijelu, treba razviti mobilnu aplikaciju s bazom podataka i implementirati razvijeni model izbora jelovnika i model aplikacije na prikladnom predlošku arhitekture. Programsko rješenje treba ispitati i analizirati na odgovarajućem skupu podataka.

2. IZAZOVI PROBLEMA NETOLERANCIJE HRANE I APLIKACIJE ZA POTPORU

Osobe koje imaju nuspojave pri konzumiranju nekih namirnica nazivaju se netolerantnim osobama na tu namirnicu. Postoji mnoštvo ljudskih netolerancija namirnica, ali u ovoj temi obradit će se samo neke. Također, neki ljudi preferiraju ne jesti meso ili uopće životinjske proizvode pa se i njima treba prilagoditi. U svrhu obuhvaćanja slučajeva netolerancije i nekih od ograničenja prehrane, ova aplikacija će sadržavati neke od parametara netolerancije i neke od parametara ograničenja prehrane.

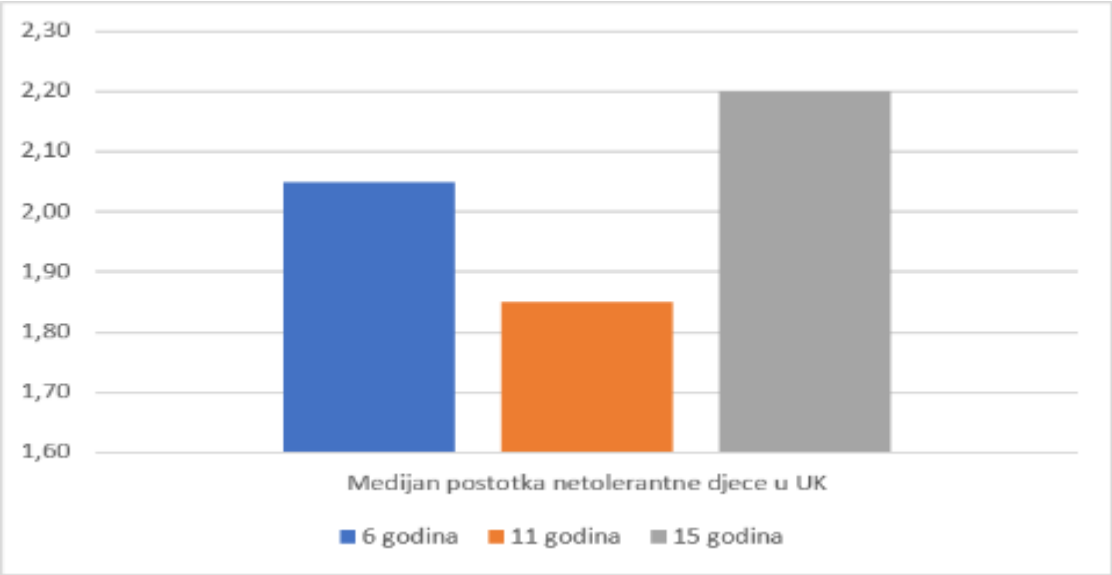
Postoji mnoštvo aplikacija koje služe kao pripomoć pri pripremi hrane, ali u ovome radu će se spomenuti njih pet. Većina takvih aplikacija podržava samo filtriranje s obzirom na ograničenja prehrane ili ovisno o tipu jela, a ne prema netolerancijama osoba.

2.1. Ljudska netolerancija hrane i ograničenja

Netolerancijom hrane smatra se nemogućnost probave nekog oblika hrane ili namirnice koja može dovesti do raznih simptoma. Neke netolerancije mogu biti recesivne, što znači da ne moraju nužno izazivati simptome, ali ih i dalje genetski prenosimo. S druge strane, postoje netolerancije koje su dominantne. Dominantne netolerancije u svakom slučaju pri nasljeđivanju zadaju mnoge poteškoće. Osobe koje imaju neku od netolerancija, moraju biti iznimno pažljive tijekom kupovine namirnica ili naručivanja jela. Nemarom ili nedovoljnom pažnjom osobe se mogu dovesti u lošu situaciju pri konzumiranju neke hrane koja sadrži namirnicu na koju su netolerantni. Djeca su osobito rizična skupina jer teško mogu pratiti kada i što jedu [1].

U [2] se prema [3] navode reakcije koje nisu posredovane imunološkim sustavom nazivaju se netolerancijama hrane. Netolerancije hrane javljaju se zbog različitih mehanizama kao što su nedostatak enzima i farmakoloških utjecaja na određene pojedince. Simptomi se razlikuju prema mehanizmu i po jačini od blagih do ozbiljnih. Mogu zahvaćati jedan ili više organa, a najčešće zahvaćaju kožu i probavni sustav.

Također, navode da prema istraživanju u svrhu dokazivanja učestalosti netolerancija hrane kod djece u Ujedinjenom Kraljevstvu dobili su rezultate koji su prikazani na slici 2.1. [2].



SlikaError! No text of specified style in document. **2.1.** Grafički prikaz netolerancije hrane kod djece u Ujedinjenom Kraljevstvu

2.1.1. Laktoza

Prema [4], konzumiranje mlijeka ili mliječnih proizvoda često izaziva štetne probavne, ali i ostale simptome. Neki od tih simptoma su grčevi i nadutost. Postoji učestalo razmišljanje da laktoza u mlijeku uzrokuje simptome, ali postoji još mnogo tvari u mlijeku koji ih mogu uzrokovati. Također, učestalo mišljenje je da je netolerancija laktoze urođena, ali postoji i mnoštvo neurođenih faktora koji mogu izazvati netoleranciju laktoze. Očekuje se da je najučestaliji razlog simptoma netolerancije nemogućnost probave laktoze u mlijeku i mliječnim proizvodima.

Kao što je navedeno, netolerancija laktoze može biti urođena, ali prema [4] navodi se da u ranom djetinjstvu, nezrelost laktaze može dovesti do sumnje na simptome netolerancije laktoze. Zato je potrebno temeljito provjeriti uzroke nepravilnog probavljanja laktoze, pogotovo kod osoba mlađe dobi.

2.1.2 Kofein

Prema [5], kofein je najrasprostranjenija psihoaktivna droga na svijetu. Srednja količina konzumiranog kofeina je 193 mg dnevno. Kofein je prirodni sastojak u preko 60 različitih vrsta

biljaka, uključujući kavu, čaj i kakao kao najpoznatije. U prirodnom obliku, kofein je kiseli bijeli prah koji je osrednje topljiv u vodi. Najčešće se konzumira u kavi i čaju. Također, povećala se koncentracija kofeina u ostalim pićima. Primjerice, povećala se popularnost energetskih pića koja sadrže od 50 mg do 375 mg kofeina po pakiranju.

U većim dozama kofeina dolazi do negativnih posljedica. Kao posljedica može doći do povećanja anksioznosti, nervoze, loše volje, čak i do nadraženog želudca. Prevelikom konzumacijom kofeina može doći do trovanja. Neki od simptoma trovanja su neumornost, nervoza, nesanica, titranje mišića i aritmija. Ovakve simptome najviše imaju osobe koje su netolerantne na kofein i djeca. Povremeni potrošači kofeina također mogu osjetiti simptome netolerancije pri unosu velikih količina kofeina. Nekoliko istraživanja pokazuje da konzumiranje kofeina kod pojedinaca može izazvati halucinacije, pogotovo kod onih koji su izloženi stresu [5].

2.1.3. Fruktoza

Prema [6], fruktoza je monosaharid koji se učestalo nalazi u voću i koristi u mnoštvu hranjivih proizvoda kao zaslađivač. Prevelikim unosom fruktoze može doći do nepravilnosti u organizmu, a netolerancija može biti urođena. Tijekom povijesti, fruktoza se koristila kao zaslađivač kod dijabetičara jer je slabo utjecala na porast glukoze u krvi za razliku od ostalih zaslađivača.

Postoje tri tipa netolerancije fruktoze. To su malapsorpcija fruktoze, nedostatak jetrene fruktokininaze i urođena netolerancija fruktoze. Malapsorpcija fruktoze je tip netolerancije hrane koja zahvaća 40% osoba u zapadnoj zemljinoj polutci. Uzroci mogu biti razni kao što je genetika, izloženost fruktozi, a i općenito zdravlje osobe. Osobe koje imaju malapsorpciju ne mogu probaviti fruktozu. Kao rezultat, fruktoza samo prolazi kroz debelo crijevo i stvara vjetrove i bolno probavljanje hrane. S druge strane, osoba može imati nedostatak jetrene fruktokininaze, a da toga ne bude ni svjesna. To se događa jer je bolest recesivna. Dobivanjem gena samo od jednog roditelja, dijete bude samo prenositelj, a bolest ne djeluje. Osobama koje su naslijedile oba gena nedostaje enzima koji u jetri koji služe za razgradnju fruktoze. Posljednja, urođena netolerancija fruktoze ozbiljan je oblik netolerancije. Događa se kada osoba ne može probaviti fruktozu niti njene prethodnike, kao što je smeđi šećer. Osobama s ovom netolerancijom nedostaje enzima koji pomažu pri razgradnji fruktoze. Neprobavljena fruktoza gomila se na jetri i bubrezima što može dovesti do kobnih situacija [7].

2.1.4. Gluten

Netolerancija glutena učestali je problem koji nastaje kao štetna reakcija na gluten iz žitarica. Najteži oblik netolerancije je celijakijska bolest koja zahvaća oko 1% svjetske populacije. S druge strane, necelijakijska netolerancija glutena zahvaća do 13% populacije, ali blaži je oblik netolerancije. Bez obzira na to što je blaži oblik, i dalje uspijeva izazvati probleme. Oba oblika mogu izazvati mnoštvo simptoma koji ne moraju nužno imati veze s probavom [8].

Glutenska ograničenja pojedinaca s necelijakijskom osjetljivosti na žito mogu biti učinkovit prehrambeni plan. Probavljanjem žitnih proizvoda, pojedinci mogu osjetiti probavne simptome koje s ovim planom izbjegavaju. U slučaju sekundarne hipolaktazije koju prouzroči celijakijska bolest, prehrana bez glutena može potaknuti povratak laktaze u probavni sustav. To je spor proces, ali može odlično završiti [4].

2.1.5. Sulfiti

Sulfiti su tvari koje se uglavnom nalaze u hrani ili kao dodaci hrani. Netolerancija na sulfite nije toliko učestala, ali ipak postoji. Kao i za većinu netolerancija, vrijedi da simptomi i reakcije variraju od blagih pa sve do opasnih po život. Sulfiti se uglavnom ne koriste na svježim namirnicama, ali se i dalje koriste u kuhanim i obrađenim namirnicama. Također, sulfiti mogu nastati prirodnim putem. Primjer nastajanja sulfita je pravljenje vina ili piva. Zato se predlaže izbjegavanje takvih namirnica, ako postoji slučaj netolerancije sulfita [9].

2.1.6. Vegetarijanstvo

Prema [10], vegetarijanstvo je niz prehrambenih načina, koji isključuju konzumiranje nekih ili svih mesnih proizvoda ovisno o razini vegetarijanstva. Moguća su i dodatna prehrambena ograničenja koja ovise o pojedinoj osobi. Bitno je odrediti detalje nutrijenata koji se unose vegetarijanskom prehranom zbog pravilnog unosa nutrijenata za zdrav život. Tradicionalne vegetarijanske prehrane imale su nizak unos energije, proteina i raznih vitamina. Danas je dostupan niz biljne hrane tijekom cijele godine kojima se može regulirati unos nutrijenata.

Prema [10], navodi se da postoji nekoliko nutritivnih prednosti vegetarijanske prehrane na zdravlje pojedinca. Biljna hrana u većini slučajeva bogatija je nutrijentima s obzirom na broj kalorija koji se njima unese. Bogate su raznim vitaminima i kompleksnim ugljikohidratima što pozitivno utječe na zdravlje. Neke od njih bogate su proteinima, kao što su mahunarke. Neki proteini koji se dobivaju iz biljnih izvora su "nepotpuni". Unosom male količine životinjskih proteina ili drugih biljnih izvora proteina može se upotpuniti ljudska potreba za svim aminokiselinama koje se nalaze u proteinima.

2.1.7. Veganstvo

Prema [10], veganstvom se smatra izbjegavanje svih životinjskih proizvoda, kao što je meso, riba, perad, jaja i mliječni proizvodi. Neki povećavaju svoja ograničenja ne korištenjem životinjskih proizvoda, kao što je koža ili krzno. Veganstvo zahtjeva detaljno praćenje unosa nutrijenata jer ih je teško unositi ovakvom prehranom. Uglavnom se moraju unositi razni izvori nutrijenata da bi se dobila dovoljna količina nutrijenata za zdrav život.

Veganska prehrana predstavlja više problema pri unosu dovoljne količine mikronutrijenata nego druge prehrane kod djece. Veganskom dijetom isključena je nekolicina grupacija hrane koje su bogate mikronutrijentima. Veganske dijete mogu imati nizak unos kalcija, željeza, cinka i ostalih oblika ovih minerala koji nisu dostupni u velikoj količini u prirodnom okruženju [10].

2.2. Recepti za pripremu hrane, njihova ograničenja i mogućnosti

Receptom se smatra tekst koji obuhvaća namirnice i način pripreme tih namirnica da bi se dobio rezultat u obliku jela. Svaki recept mora sadržavati namirnice, a time može doći i do problema netolerancije i ograničenja koji su opisani. Recept je „dinamičan“ u smislu da se može mijenjati po potrebi. Neke namirnice jednostavno nije moguće zamijeniti, gdje su vidljivi nedostatci. Svakom prilagodbom recepta stvara se novi recept koji nekome možda više odgovara i tim načinom stvara se više recepata koji odgovaraju većem broju ljudi. Nekim osobama se možda sviđaju namirnice i njihova kombinacija, ali se možda ne sviđa način pripreme nekih pa se odluče napraviti novi s izmijenjenim načinom pripreme. Glavni nedostatak recepata je što ne vole ili ne mogu podnijeti svi iste namirnice. Tko ima netolerancije prema nekim oblicima hrane može eksperimentirati s namirnicama koje njemu odgovaraju i time stvoriti recept koji se može svidjeti i ostatku svijeta. Svatko ima svoj ukus i želje, a recepti pri tome mogu zadovoljiti svako nepce.

2.3. Izazovi u području prilagodljive pripreme hrane

Prema [11], prehrana igra veliku ulogu u razvoju, održavanju i pravilnom radu stanica imunološkog sustava. Tradicionalna razmišljanja govore da bi se kod djece trebalo izbjegavati alergene koji bi potencijalno mogli izazvati netoleranciju na neku hranu, dok neki govore da je to idealno vrijeme za unos takvih alergena. Ni danas nema znanstvenih dokaza koji mogu opovrgnuti ijedan od tih navoda. Postoje dokazi koji u nekim slučajevima smanjuju šansu za netolerancijom kada se dijete izloži takvoj prehrani, a u drugim slučajevima se smanji kada se ne izloži. Potrebno je djetetu postaviti što raznovrsniju prehranu jer postoje dokazi da unos raznih vrsta hrane pozitivno utječe na imunološki sustav djeteta.

Temelj plana prehrane kod netolerancija hrane je individualizirani plan prehrane za svaku osobu kojim se izbjegava hrana koja osobi smeta. Kod djece, potrebno ih je ne izlagati hrani koja im može izazvati akutne ili kronične simptome. S druge strane, potrebno im je ponuditi zdravu i nutritivno uravnoteženu prehranu da se ne bi sputavao njihov razvoj. Kod nekih osoba u potpunosti je potrebno izbjegavati alergene, ali u novijim istraživanjima pronalazi se da netolerantne osobe mogu probaviti hranu na koju su netolerantni ukoliko je ona pripremljena na drugačiji način. Uz neugodne posljedice koje dolaze pri izboru hrane kod netolerantnih osoba, može doći i do pojave anksioznosti u društvenim okupljanjima. U takvim slučajevima trebalo bi se polako uvoditi netolerantne namirnice u prehranu da bi se pokušao poboljšati imunološki sustav osobe [11].

2.4. Mobilne aplikacije za pomoć u prilagodbi hrane

2.4.1. Easy Recipes

Easy Recipes je mobilna aplikacija koju je napravio DIL. Aplikacija se sastoji od minimalno pet zaslona, ovisno koliko su se koristili u više slučajeva isti zasloni. Aplikacije je podijeljena na četiri glavna zaslona koji uključuju početni zaslon, nove recepte, favorite i košaricu. Na početnom zaslonu vidljivi su recepti grupirani po tipu jela. Na vrhu zaslona prikazanih recepata nalazi se pretraživač u koji možemo unijeti željeno ime recepta da se pretraži u bazi podataka. Zaslon koji prikazuje nove recepte sastoji se od prikazanih recepata koje su korisnici nedavno dodali u bazu podataka. Zaslon s favoritima sastoji se od dva dijela, omiljenih recepata i onih koje je objavio korisnik. Oba dijela zaslona mogu se izmjenjivati običnim povlačenjem po zaslonu lijevo ili desno. Zaslon za košaricu prikazuje recept i namirnicu koju korisnik postavlja u košaricu. Omogućeno je i brisanje namirnica.

Odabirom nekog recepta otvara se zaslon na kojem se prikazuju detalji o receptu, sastojci koji se mogu dodati u košaricu i upute za pripremu recepta.

2.4.2. Tasty

Tasty je mobilna aplikacija koju je napravio BuzzFeed. Za razliku od Easy Recipes-a, Tasty pri izradi profila pita želi li korisnik da mu se prikazuju recepte s mesom ili ne. Aplikacija se sastoji od četiri zaslona. To su početni zaslon, favoriti, postavke i detalji o receptima. Početni zaslon sastoji se od prikazanih recepata koji su grupirani po raznim kriterijima. Na vrhu zaslona nalazi se pretraživač preko kojega možemo upisati željeno ime recepta i provjeriti postoji li isti u bazi podataka. Drugi zaslon, zaslon s favoritima podijeljen je na dva dijela, nedavno pregledane recepte i kuharice. U nedavno pregledanom dijelu vidimo recepte koje je korisnik nedavno pregledao, dok u kuharicama vidimo recepte koje je korisnik označio favoritima sortirane u kuharice. S ovoga zaslona može se otići na zaslon s postavkama za aplikaciju. Na tom zaslonu može se postaviti jezik aplikacije i ostale osnovne postavke za prikaz aplikacije. Zadnji zaslon služi za prikaz detalja recepta nakon što se odabere recept. U detaljnom prikazu recepta možemo vidjeti potrebne sastojke, preporuke i način pripreme.

2.4.3. Yummly

Yummly je mobilna aplikacija koju je napravio Yummly. Pokretanjem aplikacije treba se izraditi profil. Nakon što korisnik izradi profil, aplikacija nudi različite upitnike koji za svrhu imaju prikazivanje odmah dostupnih recepata ovisno o dostupnosti namirnica ili filtracija onih recepata čiji su sastojci nepoželjni. Aplikacija se sastoji od pet zaslona. To su zaslon s preporučenim receptima, zaslon za istraživanje recepata, pro, košarica i detaljniji prikaz recepata. Na zaslonu s preporučenim receptima dobivamo recepte koji su preporučeni s obzirom na parametre koje je korisnik odabrao pri izradi profila. Odlaskom na zaslon za istraživanje korisnik mogu se pronaći grupe recepata. Odabirom grupe prikazuju se recepti iz te grupe. Pro zaslon služi za korisnike koji imaju plaćeni račun da bi vidjeli recepte koje su pripremili profesionalni kuhari. Odlaskom na košaricu vidljivi su sastojci koje je korisnik stavio za kupovinu ili u virtualnu kuhinju. Odabirom nekog od ponuđenih recepata

odlazimo na zaslon koji prikazuje detalje o receptu. Na zaslonu su prikazani sastojci, nutritivne vrijednosti, komentari i slični recepti.

2.4.4. SideChef

SideChef je mobilna aplikacija koju je napravio SideChef. SideChef pruža korisniku opciju prijave kojom se dobiva dodatna mogućnost odabira nekoliko osobnih preferencija kao što su ograničenja prehranom ili netolerancije. Aplikacija se sastoji od šest zaslona. Nakon stvaranja profila otvara se zaslon koji prikazuje recepte preporučene za korisnika. Zaslon omogućuje pretraživanje recepata u bazi podataka, a na njemu se nalaze i preporučeni recepti u raznim grupama. Postoji jedan napredniji način pretraživanja recepata za razliku od ostalih opisanih aplikacija, a to je pretraživanje preko sastojaka. Sljedeći zaslon sadrži nove recepte za korisnika, a prikazani su i profesionalni kuhari gdje se preporučuju njihovi recepti. Sve recepte moguće je spremiti u zbirke recepata koje su prikazane na jednom od zaslona. Moguće ih je stvarati i izmjenjivati. SideChef na jednom od svojih zaslona omogućuje stvaranje planova prehrane po danima i obrocima. Aplikacija podržava upisivanje košarice za kupovinu namirnica koje su potrebne. One su prikazane na zasebnom zaslonu. Odabirom recepta ili kuhara prikazuje se zaslon na kojem je moguće vidjeti detalje o receptu i njegovoj pripremi.

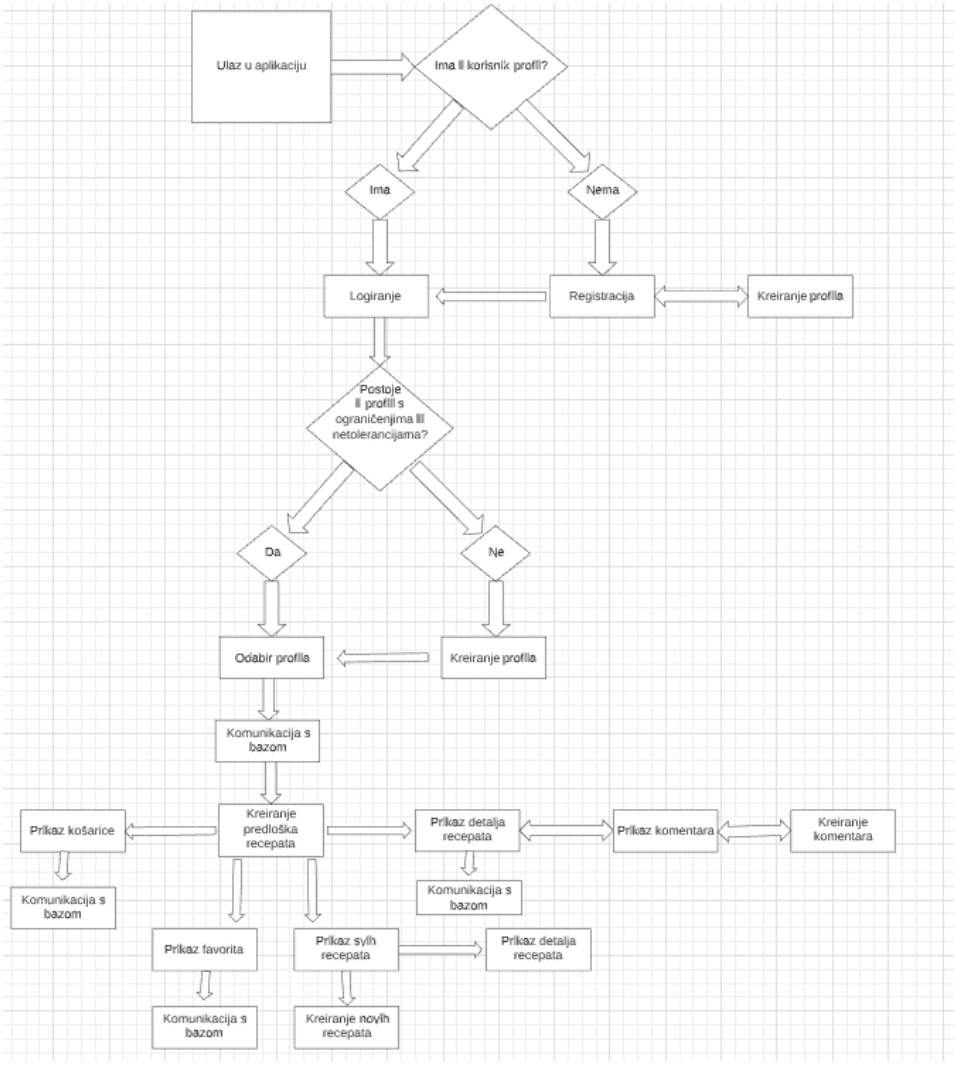
2.4.5. My Cookbook

My Cookbook je mobilna aplikacija koju je napravio Maadinfo Services. Ulaskom u aplikaciju omogućuje se prijava korisnika, koja nije nužna, ali dobiva se više mogućnosti. Aplikacija se sastoji od osam zaslona. Na početnom zaslonu prikazuju se korisnikovi recepti. Odabirom nekog recepta ulazi se na zaslon o detaljima tog recepta, gdje je moguće vidjeti sastojke i upute za pripremu. Postoji sličan zaslon koji omogućuje pretragu recepata prijatelja. My Cookbook podržava stvaranje recepata koji će se dodati u korisnikove recepte, a to se odvija na posebnom zaslonu. Aplikacija nudi opciju dodavanja recepata s interneta. Točnije, aplikacija pretražuje nekoliko stranica s receptima da bi pronašla recept koji se slaže s ključnom riječi ili imenom unesenim u tražilicu. Moguće je kao i u aplikaciji opisanoj u poglavlju 2.3.4. imati planove prehrane po danima s manje mogućnosti nego što to nudi spomenuta aplikacija. My Cookbook prati u stopu i ostale aplikacije sa stvaranjem zaslona za

prikazivanje košarice za kupovinu sastojaka. Na jednom od zaslona moguće je postavljati oznake na recepte i pretražiti ih po istim.

2.5. Idejno rješenje mobilne apilkacije

Idejno rješenje, tijek korištenja i ponašanje mobilne aplikacije može se pojednostavljeno prikazati dijagramom toka na slici 2.2.

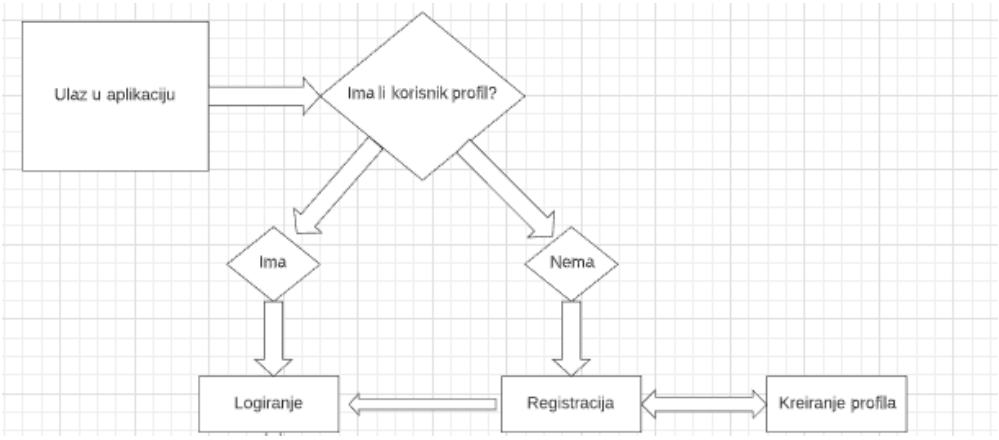


Slika 1.2. Ponašanje aplikacije preko dijagrama toka

Ponašanje aplikacije koje je prikazano na slici 2.1. rastavit će se na više cjelina da bi se odijelile cjeline koje imaju zasebne funkcionalnosti. Idejno rješenje cjelina obradit će se u sljedećim potpoglavljima, a programsko rješenje obradit će se u četvrtom poglavlju završnog rada.

2.5.1. Prijava i registracija korisnika

Rastavljanjem dijagrama toka sa slike 2.2. na dio koji se tiče prijave korisnika i registracije, dobije se dijagram toka na slici 2.3.

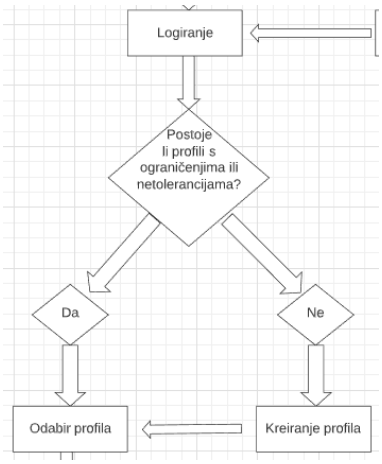


Slika 2.3. Ponašanje aplikacije pri prijavi i registraciji korisnika

Ulaskom u aplikaciju otvara se zaslon na kojem se korisnik može prijaviti ukoliko već ima profil ili registrirati u slučaju da ga nema. U slučaju da korisnik nema profil, odlazi na zaslon za registraciju gdje se od korisnika traže podaci za izradu profila. Nakon uspješne izrade profila i pohrane profila u bazu podataka, odlazi se na početni zaslon za prijavu. Korisnik bi sada trebao imati svoj profil s kojim se prijavi unošenjem korisničkog imena i lozinke. Provjerom unesenog korisničkog imena i lozinke s bazom podataka, korisniku se odobrava ili ograničava daljnji pristup aplikaciji.

2.5.2. Stvaranje i odabir profila

Rastavljanjem dijagrama toka sa slike 2.2. na dio koji se tiče stvaranja profila s netolerancijama i ograničenjima, dobije se dijagram toka prikazan na slici 2.4.

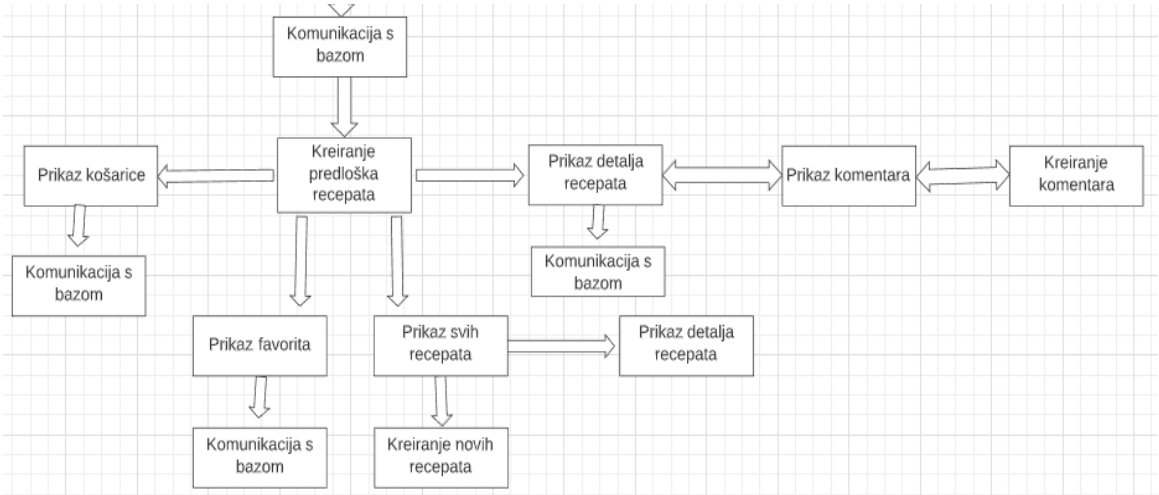


Slika 2.4. Ponašanje aplikacije pri rukovanju s profilima

Kao što je prikazano na slici iznad, prijavom korisnika dolazi se na zaslon s profilima. Ako je korisnik prethodno koristio aplikaciju, već ima postavljen profil s netolerancijama i ograničenjima. U slučaju da je novi korisnik, potrebno je izraditi novi profil. Izrada profila odvija se na zasebnom zaslonu u kojem može odabrati koje netolerancije i ograničenja prehrane taj profil ima te tako ga pohraniti u bazu podataka. Nakon uspješno stvorenog profila, korisnik može odabrati profil i dobiti preporučene recepte za odabrani profil.

2.5.3. Prikaz recepata i komentara

Daljnjim rastavljanjem dijagrama toka sa slike 2.2., na dio koji se tiče recepata i komentiranja na recepte, dobiva se dijagram toka prikazan na slici 2.5.



Slika 2.5. Ponašanje recepata i komentiranja

Nakon uspješnog odabira profila događa se komunikacija s bazom podataka, gdje se na temelju netolerancija i ograničenja profila prikazuju prikladni recepti. Svi recepti koji su prikazani na tom zaslonu su prikladni za korisnika s obzirom na njegovo zdravstveno stanje.

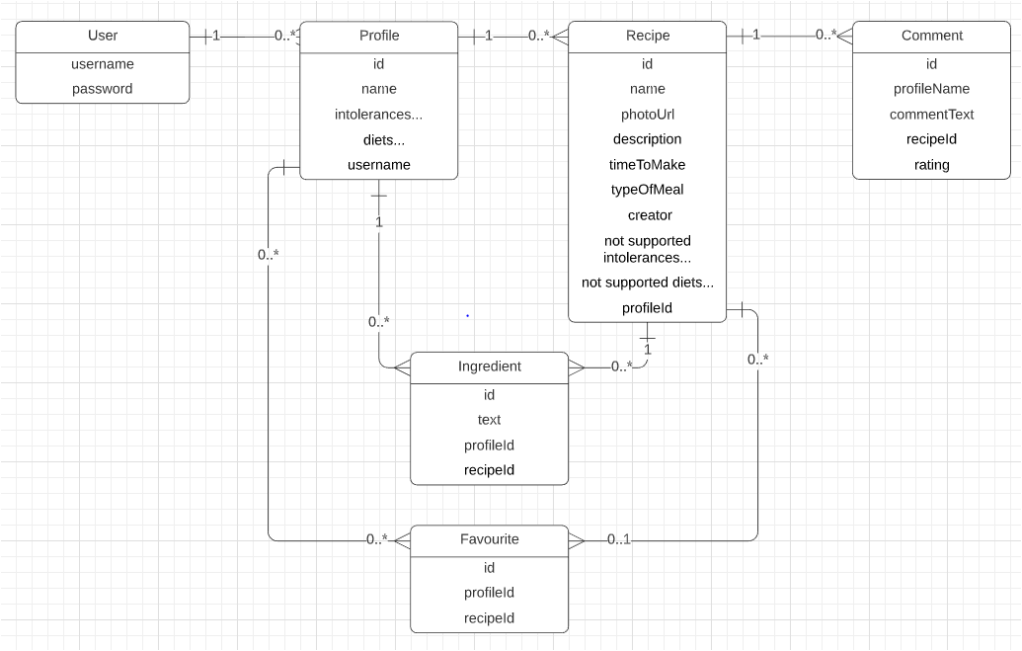
Sa zaslona predloženih recepata može se otići na košaricu namirnica koja služi kao popis sastojaka koje treba kupiti ili koje imamo. Sve namirnice i dohvaćanje namirnica događa se preko baze podataka i vezane su za korisnikove profile, a ne za korisnika. Mogu se pogledati recepti koji su označeni kao omiljeni jer su oni kao takvi već pohranjeni u bazu podataka pa se mogu tako i dohvatiti. Mogu se prikazati i svi recepti, a to se odvija na zasebnom zaslonu. Tako bi se prikazali i recepti koji nisu prikladni za korisnikov profil, ali su oni označeni crvenom bojom, a prikladni zelenom. S tog zaslona moguće je otići na sljedeći zaslon za stvaranje novih recepata pomoću formulara za stvaranje novog recepta, koji će se pohraniti u bazu podataka.

Odabirom bilo kojeg recepta, na bilo kojem zaslonu, može se otvoriti zaslon za prikazivanje detalja o receptu. Sa zaslona na kojem su prikazani detalji moguće je obrisati recept, ako je to recept koji je stvorio taj korisnički profil. Ako nije, moguće je otvoriti zaslon s komentarima i komentirati ga.

Na istom zaslonu recept se može označiti kao omiljeni, čime se prikazuje i na zaslonu s omiljenim receptima. Također, moguće je urediti recept ako je to recept korisničkog profila. U suprotnom je moguće stvoriti novi korisnički recept na temelju tog odabranog recepta.

2.5.4. Struktura baze podataka

Idejni prikaz baze podataka je prikazan na slici 2.6.



Slika 2.6. Prikaz idejnog rješenja baze podataka

Na slici 2.6. vidljivo je da se baza podataka sastoji od korisnika, profila, recepta, komentara, sastojaka i omiljenih. Komponenta korisnika je početna komponenta. Kao što je prikazano, jedan korisnik može imati više profila. Svaki profil pohranjuje se u bazu podataka i dohvaća se preko korisničkog imena. Svaki profil može stvoriti više sastojaka koje stavlja u košaricu ili više recepata. Recepti su vezani direktno za profile da bi se mogle prikazati osobe koje su stvorile taj recept. Svaki recept može imati više sastojaka koji će se prikazivati pri pregledu recepta, a može imati više komentara koji se vežu za recept. Komponenta omiljenih povezuje recepte i profile. Jedan recept može biti povezan samo s jednom komponentom omiljenih, ali profil može imati više omiljenih.

3. PRIJEDLOG MODELA MOBILNE APLIKACIJE

3.1. Parametri koji opisuju netoleriranje hrane

Parametri koji su opisani u potpoglavlju 2.1. koristit će se u aplikaciji za odabire mogućih netolerancija i ograničenja u prehrani. Kao što je prikazano na slici 2.1., prosječno 2% djece u Ujedinjenom Kraljevstvu netolerantno je na neku hranu. Parametri netolerancija opisani pod tim poglavljem uzeti su zbog njihovog učestalog pojavljivanja kod ljudi, a tako su odabrana i ograničenja prehrane. Iako netolerancija na gluten nije jednako česta kao netolerancija na fruktozu, ona i dalje postoji. Budući da se gluten često pronalazi u pšeničnim proizvodima, bitno je znati nalazi li se u nekom receptu. Konzumacija kofeina ljudima nije strana. Kofein se, kao što je ranije navedeno, nalazi u preko 60 različitih biljnih vrsta. Netolerancijom na fruktozu zahvaćeno je 40% osoba zapadne zemljine polutke. Osobe netolerantne na fruktozu moraju paziti pri konzumaciji voća, a voće je potrebno za raznoliku i pravilnu prehranu. Laktoza je netolerancija mliječnih proizvoda. Osobe zahvaćene netolerancijom laktoze moraju paziti pri konzumaciji takvih proizvoda. Sulfiti su za osobe koje se ne bave proučavanjem nutricionizma i medicinskih preporuka u vezi hrane nepoznata stvar. Nepoznati su jer netolerancija na sulfite ne zahvaća veliki broj ljudi, ali ipak se nalazi u ovom radu zbog svojih reakcija koje mogu biti kobne za život. S druge strane, postoje osobe koje svojom voljom ne žele konzumirati neke namirnice. U ovome radu spominju se vegetarijanstvo i veganstvo kao ograničenja u prehrani. Osobe iz osobnih razloga biraju ne konzumirati meso ili čak ne konzumirati bilo kakve životinjske proizvode, ovisno o ograničenju koje si odaberu.

3.2. Parametri recepata

Recepti, kao što je navedeno u potpoglavlju 2.2., lako su izmjenjivi i prilagodljivi. Aplikacija će se sastojati od recepata koji će biti predloženi korisnicima. Ovisno o netolerancijama profila i netolerancijama recepta, predložit će se prikladni recepti. U ovakvoj aplikaciji recepti su korisni jer ih je lako izmijeniti onako kako je korisniku potrebno. Recepte u aplikaciji bit će moguće pregledavati i označavati kao omiljene. Svatko tko stvori recept moći će vršiti izmjene nad tim receptom, ali ako netko želi sebi prilagoditi recept imat će mogućnost i za to. Kopiranjem parametara postojećeg recepta moći će se napraviti željene izmjene i spremiti u novi recept. Stvaranjem potpuno novog recepta morat će se popuniti formular u kojem se nalaze stvari kao što su ime, sastojci i upute. U svakom stvorenom

receptu potrebno je označiti netolerancije i ograničenja recepta. Podržane netolerancije i ograničenja navedene su u potpoglavlju 2.1. Netolerantnim osobama neće se predlagati takvi recepti i bit će označeni crvenom bojom. Svaki recept moći će se komentirati i ocijeniti. Takve komentare moći će vidjeti svi korisnici.

3.3. Funkcionalni zahtjevi mobilne aplikacije

3.3.1. Registriranje i prijava korisnika

Tijekom prvog ulaska u aplikaciju korisnik neće imati svoj korisnički račun. Korisnički račun može napraviti u aplikaciji odlaskom na zaslon za registraciju. Na zaslonu za registraciju potrebno je unijeti svoje željeno korisničko ime i lozinku, a ako već postoji takvo korisničko ime zatražit će se promjena. Nakon uspješnog odabira parametara registracije pohranjuje se korisnički račun u bazu podataka. Povratkom na zaslon za prijavu korisnik se može prijaviti u aplikaciju. U pozadini se događa komunikacija s bazom podataka i provjera ispravnosti podataka.

3.3.2. Stvaranje i odabir profila

Uspješnom prijavom dolazi se na zaslon koji prikazuje sve korisničke profile. Novonastali korisnici neće imati profil pa moraju stvoriti novi odlaskom na zaslon za stvaranje profila. Na zaslonu za stvaranje profila korisnik mora unijeti ime, netolerancije i ograničenja prehrane za taj profil, ako ih ima. U pozadini se provjerava postoji li već takvo ime na korisničkom računu. Nakon uspješnog stvaranja profila, korisnik može odabrati stvoreni profil ili ga izbrisati.

3.3.3. Prikaz preporučenih recepata

Odabirom nekog od korisničkih profila dolazi se na zaslon s preporučenim receptima. Tijekom prikaza preporučenih recepata u pozadini se događa komunikacija s bazom podataka, gdje se parametri korisničkog profila uspoređuju s parametrima recepata i tako se pronalaze prikladni recepti. Prikazivanjem preporučenih recepata olakšava se korisniku pretraga prikladnih recepata. Ako neki

recept ne podržava neku od netolerancija koje korisnički profil ima, taj recept se neće prikazati korisniku.

3.3.4. Stvaranje novih recepata

Svaki prijavljeni korisnik može stvoriti svoje recepte na kojima se označava da ih je on stvorio. Pri stvaranju recepta, korisnik mora odabrati prikladno ime recepta koje ne smije biti zauzeto, može unijeti URL adresu slike, opisati postupak pripreme, koliko traje priprema, sastojke i treba označiti sadržava li recept neke sastojke koji bi narušili netolerancijama ili ograničenjima drugih korisnika.

3.3.5. Stvaranje recepata na temelju postojećih

Kao što je navedeno u potpoglavljima 2.2. i 3.2. da su recepti prilagodljivi, aplikacija omogućuje prilagođavanje recepata kopiranjem postojećeg recept i omogućavanjem uređivanja. Nakon uređivanja korisnik mora odabrati novi naziv recepta jer se stvara novi recept u bazi podataka. Ovakav način stvaranja recepta omogućen je korisnicima koji nisu napravili originalni recept. Stvaranjem recepata na temelju postojećih recepata, korisnici mogu prilagoditi recept po svojim željama.

3.3.6. Dodavanje komentara na recepte

Komentari na receptima mogu biti dobri i loši, ali dodavanjem komentara na recepte omogućuje se korisniku recepta pregled uspješnosti recepta kod ostalih korisnika ili čak naputke za poboljšanje recepta. Komentiranje recepata mogu vršiti samo korisnici koji nisu napravili originalni recept. Pri komentiranju se mora staviti i ocjena od jedan do pet radi lakšeg pregleda općenite ocjene recepta. Time se omogućava da u kratkom roku vidimo kako se svidio recept ostalim korisnicima.

3.3.7. Dodavanje omiljenih recepata

Pri pregledu recepata korisnicima se omogućuje označavanje recepta kao omiljeni. Označavanjem recepta kao omiljeni recept pohranjuje se u bazu podataka. Pohranjene omiljene recepte moguće je

pregledati na zaslonu za omiljene recepte. Označavanjem nekog recepta omiljenim, korisnik si može spremiti recept koji mu se svidio ili koji želi napraviti kasnije bez potrebe za ponovnom pretragom. Označavanjem recepta omiljenim, stvara se nova veza u bazi podataka koja govori da je neki recept označen kao omiljeni.

3.3.8. Dodavanje sastojaka u košaricu

Postojanjem košarice korisnik dobiva način zapisivanja sastojaka koje treba kupiti ili koje već posjeduje. Košarica se nalazi na zasebnom zaslonu na kojem se omogućuje dodavanje sastojaka. Dodavanjem sastojka u košaricu od korisnika se traži naziv sastojka gdje može napisati što god poželi. Slobodnim upisom sastojaka daje se sloboda korisniku da napiše i koliko takvih sastojaka mu treba ili koliko posjeduje. Svaki sastojak koji korisnik spremi na nekom od svojih korisničkih profila veže se na korisnički profil, a ne za korisnički račun i na takav način se pohranjuje u bazu podataka.

3.3.9. Struktura baze

Struktura baze sastoji se od komponenata opisanih u potpoglavlju 2.5.4. Komponenta korisnika služi za prijavu korisnika u aplikaciju zbog čega korisnici mogu komentirati i stvoriti nove recepte. Svaki registrirani korisnik pohranjuje se kao komponenta korisnika u bazu podataka. Komponenta se sastoji od *username* parametra koji označava korisničko ime i *password* parametra za zaporku. Komponenta profila služi za pohranu korisničkih profila da bi se mogli prikazati prikladni recepti svakom korisničkom profilu. Iz tog razloga, profil kao parametre ima netolerancije i ograničenja korisnika. Profili se pronalaze preko korisničkog imena korisnika. Recepti se pohranjuju preko profila koji ih naprave, a tako se vidi tko može uređivati recept i brisati ga. Omiljeni su poveznica između recepata i korisničkih profila čime se mogu pronaći svi omiljeni recepti preko profila. Sastojci su povezani s profilom i receptom jer koriste istu komponentu za spremanje recepata u košaricu i recepte.

4. PROGRAMSKO RJEŠENJE APLIKACIJE ZA PRILAGODLJIVU PREHRANU

4.1. Programske tehnologije, okoline i jezici korišteni za izradu aplikacije

4.1.1. Android studio

Android studio je razvojno okruženje koje je razvio Google u suradnji s JetBrains-om. Android studio, kao što mu i samo ime govori, služi za razvoj Android aplikacija bez obzira na platformu na kojoj se koristi. Za lakše razvijanje aplikacija omogućuje vizualno oblikovanje zaslona aplikacija i pametno pisanje programskog koda koje prepoznaje i daje prijedloge što se htjelo napisati. Za testiranje aplikacija pruža emulator koji omogućuje ispitivanje aplikacije.

4.1.2. Kotlin

Kotlin je relativno novi open source objektno-orijentirani programski jezik. Kotlin je došao kao nasljednik Java programskog jezika i unaprijedio ga u nekim aspektima. Kotlin znatno reducira programski kod, riješio je problem s nepostojećim objektima i moguće ga je koristiti za razvoj aplikacija na više platformi. U zadnje vrijeme širi se i na web programiranje.

Prema [12] Kotlin je open source, besplatan, općenito primjenjivi programatski programski jezik koji je originalno dizajniran za JVM i Android. Temelji se na sigurnosti, jasnoći i podržavanju raznih alata. Kotlin ima sve značajke i prednosti funkcijskih jezika.

4.1.3. XML

XML ili Extensible Markup Language je jezik koji je stvoren u svrhu ljudskog čitanja datoteka koje stvori računalo. Prije stvaranja takvih jezika za prikazivanje čitljivog teksta unutar datoteka bilo je teško prepoznati što piše u datoteci. Najpopularniji su JSON i XML koji pomažu pri razvoju aplikacija. XML u okvirima Android studija koristi se kao jezik za opisivanje zaslona na aplikacijama i postavljanje elemenata na zaslon.

4.1.4. Biblioteka Room Database

Room Database je biblioteka koja omogućuje postavljanje lokalne SQLite baze podataka za Android aplikacije. Implementiranjem dobiva se baza podataka koju je moguće oblikovati po želji. Pri stvaranju objekata koristi se programski jezik za stvaranje mobilne aplikacije, a za pristup objektima i bazi podataka koristi se SQLite jezik.

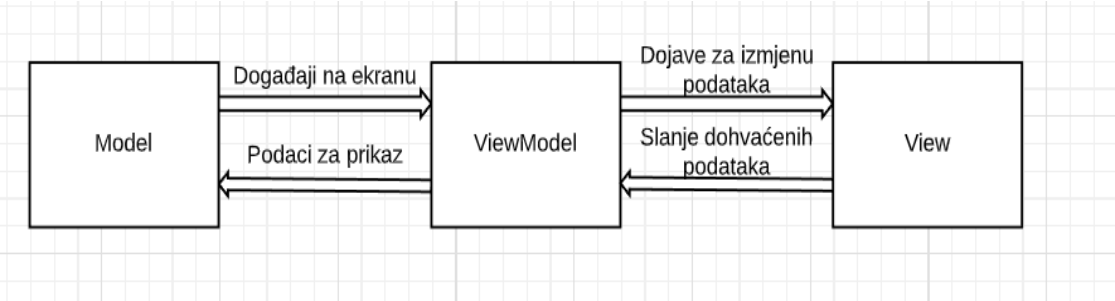
Prema [13] Room je cijela biblioteka koja omogućuje lakše stvaranje i manipuliranje SQLite bazom podataka. Koriste se oznake kojima definiramo svoje baze podataka, komponente baze podataka i operacije. Room biblioteka sama prepoznaje te oznake i pretvara ih u SQLite programski kod koji se krije iza oznake. Biblioteka pruža sloj apstrakcije na SQLite i zbog toga je cijelo dohvaćanje objekata iz baze podataka jednostavnije i brže.

4.2. **Programski predložak arhitekture MVVM**

4.2.1. Arhitektura MVVM

Arhitektura MVVM, kao i svaka druga arhitektura, označava način grupiranja programskog koda u aplikacijama. Cilj MVVM arhitekture je podijeliti programski kod na tri glavna dijela. To su *Model, View* i *ViewModel*.*Model* je dio programskog koda koji se odnosi na izvorne podatke i njihovo dohvaćanje. To su klase koje obavljaju internetske pozive ili pozive za dohvaćanje objekata iz lokalne baze podataka. S *Modelom* komunicira samo *ViewModel*. *ViewModel* sadržava sav programski kod koji je vezan za promjenu oblika podataka ili izvršavanje nekih logičkih radnji da bi se kasnije mogle pravilno prikazati na zaslonu. *ViewModel* treba, uz *Model,* komunicirati i sa zaslonom, što se naziva *View*.*View* je programski kod koji je potreban za postavljanje i funkcioniranje zaslona. *View* smije primati podatke samo s *ViewModel*a te ih prikazivati kako je postavljeno u programskom kodu. Prikaz komunikacije i što se događa tijekom komunikacije prikazano je na slici 4.1.

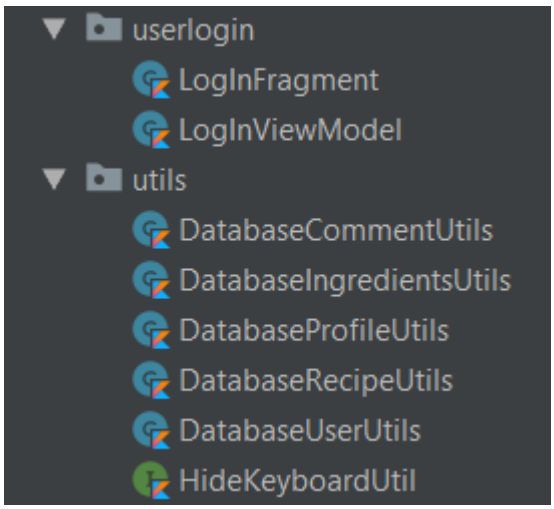
Prema [14] glavni dijelovi MVVM arhitekture su *View, ViewModel i Model. View* obavještava *ViewModel* o događanjima na zaslonu od strane korisnika. *ViewModel* omogućava *Viewu* pristup podacima koji su mu potrebni za pravilan prikaz zaslona. *Model* služi za pohranu i dohvaćanje podataka.



Slika 4.1. Prikaz komunikacije podjele MVVM arhitekture

4.2.2. Primjenjena arhitektura MVVM

Primjenom arhitekture MVVM dobiva se na čitljivosti i organiziranosti programskog koda. U aplikaciji ovoga završnog rada primijenjena je MVVM arhitektura. Struktura datoteka vidljiva je na slici 4.2.



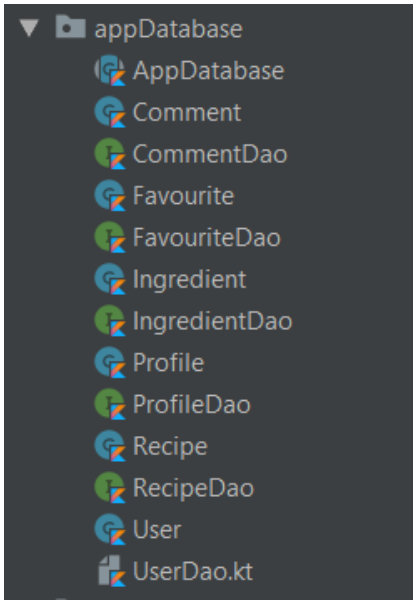
Slika 4.2. Prikaz MVVM strukture datoteka

View dio MVVM arhitekture i programski kod koji je potreban za prikazivanje zaslona za prijavu korisnika nalazi se u datoteci *LogInFragment*. Fragmenti imaju svoje *ViewModele* za pozadinsku logiku. *LogInViewModel* koristi se za spremanje cijele logike dohvaćanja podataka od *Modela*, navigacije, prerade podataka i općenite logike koju je potrebno obaviti na zaslonima.

Na slici 4.2. vidljivo je da u *utils* postoje datoteke koje vrše apstrakciju dohvaćanja podataka iz baze podataka. Ovaj princip naziva se *Repository Pattern*.

Na slici 4.3. vidljiva je struktura datoteka za *Model* dio MVVM arhitekture. Pod *Modelom*, kao što je već navedeno, smatraju se klase koje dohvaćaju čiste podatke bez ikakvih promjena nad njima. Zbog toga je *appDatabase* odvojen i služi kao *Model* dio cijele aplikacije.

Na slici 4.3. vidljive su sve komponente baze podataka kao i njihove pristupne točke koje se nazivaju isto kao i komponenta baze podataka uz prefiks „Dao“.



Slika 4.3. Prikaz strukture datoteka za Model dio MVVM arhitekture

4.3. Prikaz glavnih dijelova programskog koda

4.3.1. Registracija i prijava korisnika

Glavni problem prijave korisnika svodi se na dohvaćanje korisnika iz baze podataka te pretraživanje postojanja korisnika u bazi podataka.

```
fun onVerificationClicked(username: String, password: String, rootLayout : View) {
    coroutineScope.launch { this: CoroutineScope
        val currentUser :User?  = databaseUserUtils.getUserByName(username)
        if (currentUser != null) {
            if (passwordCheck(currentUser, password)) {
                withContext(Dispatchers.Main){ this: CoroutineScope
                    navigateToProfileFragment()
                }
            }else{
                withContext(Dispatchers.Main) { this: CoroutineScope
                    popupView.error_text.text = "Wrong password"
                    popupWindow.showAtLocation(
                        rootLayout,
                        Gravity.CENTER,
                        x: 0,
                        y: 0
                    )
                }
            }
        }else{
            withContext(Dispatchers.Main) { this: CoroutineScope
                popupView.error_text.text = "User with that username doesn't exist or the fields are empty"
                popupWindow.showAtLocation(rootLayout, Gravity.CENTER, x: 0, y: 0)
            }
        }
    }
}
private fun passwordCheck(user: User?, password: String) = user!!.password == password
```

Programski kod 4.1. Provjera korisnika i njihova prijava

Iz programskog koda 4.1 vidljivo je da *onVerificationClicked* prima kao parametre *username, password* i *rootLayout*. Uvođenjem *rootLayouta* preko parametara dobiva se na nepovezanosti *View* i *ViewModel* strane čime se dobiva mogućnost izbacivanja iskočnih prozora. *ViewModel* komunicira s bazom podataka preko *databaseUserUtilsa* da dohvati sve korisnike s tim korisničkim imenom što se pohranjuje u konstantu *currentUser*. U slučaju da ne postoji, izbacuje se iskočni prozor s tekstom da takav korisnik ne postoji ili je unos bio prazan. Postojanjem takvog korisnika prolazi se daljnja provjera unesene lozinke funkcijom *passwordCheck.* Ako je lozinka ispravna korisnika se šalje na idući zaslon funkcijom *navigateToProfileFragment*, a u suprotnom izbacuje se iskočni prozor s tekstom o pogrešnoj lozinki.

Glavni problem registracije je stvaranje novih korisnika, a uz to potrebno je paziti da se ne stvore dva ista korisnika.

```
fun insertNewUser(name: String, password: String, confirmation: String, rootLayout : View) {
    if (verifyPassword(password, confirmation) && name != "") {
        coroutineScope.launch { this: CoroutineScope
            if (checkUser(name) == null) {
                databaseUserUtils.insertUser(
                    User(
                        name,
                        password
                    )
                )
                withContext(Dispatchers.Main){ this: CoroutineScope
                    navigateToLogin()
                }

            }else{
                withContext(Dispatchers.Main){ this: CoroutineScope
                    popupView.error_text.text = "User with that username already exists"
                    popupWindow.showAtLocation(rootLayout, Gravity.CENTER, x: 0, y: 0)
                }
            }
        }
    }else{
        popupView.error_text.text = "Username empty or wrong password"
        popupWindow.showAtLocation(rootLayout, Gravity.CENTER,  x: 0, y: 0)
    }
}
private fun verifyPassword(password: String, confirmation: String) = password == confirmation
```

Programski kod 4.2. Registiracija novih korisnika

Na slici iznad prikazana je funkcija *insertNewUser* koja prima parametre sa zaslona. To su *name*, *password*, *confirmation*, *rootLayout*. Odmah pri pokretanju funkcije poziva se druga funkcija *verifyPassword* koja provjerava obje lozinke koje je korisnik upisao. U slučaju da nisu jednake izbacuje se iskočni prozor s tekstom da je korisničko ime prazno ili da se lozinke ne poklapaju što je moguće preko *rootLayout* parametra.Daljnjom provjerom pregledava se postoji li takvo korisničko ime u bazi podataka pomoću *databaseUserUtils* funkcije *insertUser* koja unosi korisnika u bazu podataka, ako ne postoji stvara se novi korisnik s unesenim imenom i lozinkom, a ako postoji izbacuje se iskočni prozor s tekstom da korisnik postoji.

4.3.2. Stvaranje i odabir profila

Glavni problem pri stvaranju profila je uzimanje svih parametara netolerancija i ograničenja prehrane sa zaslona i prosljeđivanje istih da bi se stvorio novi profil. Pri stvaranju profila potrebno je provjeriti posjeduje li korisnik već takvo ime profila.

```
fun inputProfile(name: String, list: ArrayList<Boolean>, username: String, rootLayout: View) {
    coroutineScope.launch { this: CoroutineScope
        if(!name.isEmpty()){
            if (checkIfProfileAvailable(name, username) == null) {
                databaseProfileUtils.insertProfile(
                    Profile(
                        profileId: 0,
                        name,
                        list[0],
                        list[1],
                        list[2],
                        list[3],
                        list[4],
                        list[5],
                        list[6],
                        username
                    )
                )
                withContext(Dispatchers.Main){ this: CoroutineScope
                    navigateToProfileFragment()
                }
            }else{
                withContext(Dispatchers.Main){ this: CoroutineScope
                    popupView.error_text.text = "Profile already exists"
                    popupWindow.showAtLocation(rootLayout, Gravity.CENTER, x: 0, y: 0)
                }
            }

        }else{
            withContext(Dispatchers.Main){ this: CoroutineScope
                popupView.error_text.text = "Profile name mustn't be empty"
                popupWindow.showAtLocation(rootLayout, Gravity.CENTER, x: 0, y: 0)
            }
        }
    }
}
```

Programski kod 4.3. Stvaranje novih profila

Funkcija *inputProfile* preko parametara prima *name, list, username, rootLayout.* Parametar *rootLayout* ima jednaku namjenu kao i u prijašnjim objašnjenjima. Parametri *name* i *list* koriste se pri stvaranju profila nakon što se provjeri postoji li takav profil funkcijom *checkIfProfileAvailable*, ako u jednoj od provjera je li *name* prazan ili ako profil već postoji izbacuje se iskočni prozor. U slučaju da su sve provjere uspješno prošle dolazi se do poziva *databaseProfileUtils* funkcije *insertProfile* kojom se pohranjuje profil u bazi podataka s unesenim imenom i listom netolerancija i ograničenja.

Parametar *username* u funkciji *inputProfile* potreban je jer se profil povezuje s korisnikom. Nakon stvaranja profila korisnik se vraća na zaslon s profilima gdje se dohvaćaju i prikazuju svi profili pomoću funkcije *submitNewList*. *SubmitNewList* prima kao parametar *adapter* na kojem će se podaci prikazati i *username* s kojim se profili povezuju. Pomoću *databaseProfileUtils* funkcije *getAllProfiles*, koja kao parametar prima *username* kako bi mogla dohvatiti prikladne profile za određenog korisnika, dohvaća se lista profila koja se sprema u varijablu *list*. Funkcijom koju adapter posjeduje, *submitList*, šalju se podaci adapteru koje treba prikazati.

```
fun submitNewList(adapter: ProfileRecyclerAdapter, username: String) {
    coroutineScope.launch { this: CoroutineScope
        val list : List<Profile>? = databaseProfileUtils.getAllProfiles(username)
        withContext(Dispatchers.Main) { this: CoroutineScope
            adapter.submitList(list)
        }
    }
}
```

Programski kod 4.4. Dohvaćanje i prikazivanje profila

Odabir profila svodi se na problem dohvaćanja profila iz baze podataka nakon odabira profila. Obavlja se funkcijom *getProfileFromDatabase* koja kao parametar prima identifikator profila tipa *Long* čime je svaki profil jedinstven. Dohvaćanje profila odvija se preko funkcije *databaseProfileUtilsa getProfile* koja koristi isti identifikator kako bi pronašla traženi profil u bazi podataka. Pronađeni profil postavlja se u *MutableLiveData* konstantu *_currentProfile* kako bi se moglo na *View* strani vidjeti da se odabrao neki profil što kasnije treba napraviti.

```
fun getProfileFromDatabase(profileId: Long) {
    coroutineScope.launch { this: CoroutineScope
        val profile : Profile? = databaseProfileUtils.getProfile(profileId)
        withContext(Dispatchers.Main) { this: CoroutineScope
            _currentProfile.value = profile
        }
    }
}
```

Programski kod 4.5. Odabir profila

4.3.3. Prikaz preporučenih recepata

Glavni problem s prikazivanjem preporučenih recepata nakon odabira nekog profila svodi se na provjeravanje netolerancija i ograničenja profila i receptima. Uočavanjem da profil ne podržava nešto što recept ima, recept se ne prikazuje. Funkcija *submitNewRecommendedList* prima kao parametar *adapter* i *profileId*. Parametar *adapter* koristi se kao i *rootLayout* u prijašnjim slučajevima, kako bi se moglo iskoristiti na više mjesta, a da ne bude vezano za točno jedan adapter. Prvo, potrebno je dohvatiti sve recepte preko *databaseRecipeUtils* funkcije *getRecipes* i spremiti ih u konstantu *recipes*. Potrebno je pronaći i profil s identifikatorom *profileId* koji je dobiven preko parametra korištenjem funkcije *databaseRecipeUtilsa getProfile*. Nakon toga, potrebno je stvoriti praznu listu *recommendedRecipes* koja će sadržavati sve recepte koji su prikladni za odabrani profil. Prolaskom kroz sve recepte iz konstante *recipes* provjeravaju se parametri koji mogu utjecati na prikladnost recepta za profil. Provjere se odvijaju preko funkcija *checkCaffeine* koja radi na isti princip kao i ostale gdje kao parametar primaju profil i recept te provjeravaju slažu li se parametri netolerancije na određene stvari.

```
fun submitNewRecommendedList(adapter: AllRecipeRecyclerAdapter, profileId: Long) {
    coroutineScope.launch { this: CoroutineScope
        val recipes : List<Recipe>? = databaseRecipeUtils.getRecipes()
        val profile : Profile? = databaseRecipeUtils.getProfile(profileId)
        val recommendedRecipes : MutableList<Recipe> = mutableListOf<Recipe>()
        for (recipe : Recipe in recipes!!) {
            if (!checkCaffeine(recipe, profile!!) && !checkFructose(
                    recipe,
                    profile
                ) && !checkGluten(recipe, profile) && !checkLactose(recipe, profile)
                && !checkSulfite(recipe,profile) && !checkVegan(recipe, profile) && !checkVegetarian(recipe, profile)
            ) {
                recommendedRecipes.add(recipe)
            }
        }
        withContext(Dispatchers.Main) { this: CoroutineScope
            adapter.submitList(recommendedRecipes)
        }

    }
}

fun checkLactose(recipe: Recipe, profile: Profile): Boolean {
    if (recipe.lactose) {
        return recipe.lactose == profile.lactose_intolerance
    }
    return false
}
```

Programski kod 4.6. Filtriranje preporučenih recepata

4.3.4. Stvaranje novih recepata

Stvaranje novih recepata složenije je od ostalih dijelova aplikacije jer se sastoji od više komponenata baze podataka koje je potrebno povezati u jednu cjelinu. Potrebno je spojiti profil, recept i sastojke. Kao što je prikazano na slici programskog koda 4.7., potrebna je funkcija *insertRecipe* koja kao parametre prima *recipe* i *rootLayout.* Kao početna provjera poziva se funkcija *checkExistence* koja provjerava postoji li recept s takvim imenom u bazi podataka. U slučaju da ne postoji poziva se funkcija *databaseRecipeUtils*-a *insertRecipe* koja preko parametra prima recept koji želimo stvoriti u bazi podataka. Zatim, potrebno je dohvatiti sve sastojke koje je korisnik unio za određeni recept i stvoreni recept iz baze podataka kako bi mogli pronaći pravi identifikator. Poziva se funkcija *createRecipeIngredientsWithDelete* koja prima recept i listu sastojaka kao parametre. Funkcija služi za dodavanje sastojaka u bazu podataka s poveznicom na određeni recept što se odvija preko *databaseIngetientsUtils* funkcije *insertIngredient.* Potrebno je proći kroz cijelu listu sastojaka i unijeti ih s poveznicom na pravilan recept. Funkcija još i briše sastojke koji se nalaze s identifikatorom nula jer se taj identifikator koristi za recepte koji se još prave.

```
fun insertRecipe(recipe: Recipe, rootLayout: View){

    coroutineScope.launch { this: CoroutineScope
        if (!checkExistence(recipe.name)) {
            databaseRecipeUtils.insertRecipe(recipe)
            var list : List<Ingredient>? = databaseIngredientsUtils.getRecipeIngredients( recipeId: 0)
            val foundRecipe : Recipe? = databaseRecipeUtils.getRecipeByName(recipe.name)
            createRecipeIngredientsWithDelete(foundRecipe!!, list!!)
            withContext(Dispatchers.Main) { this: CoroutineScope
                navigateToAllRecipes()
            }
        }
        else{
            withContext(Dispatchers.Main){ this: CoroutineScope
                popupView.error_text.text = "Recipe with that name already exists or the name is empty"
                popupWindow.showAtLocation(rootLayout, Gravity.CENTER, x: 0, y: 0)
            }
        }
    }
}
private fun createRecipeIngredientsWithDelete(recipe: Recipe, ingredients: List<Ingredient>) {
    for (ingredient :Ingredient in ingredients) {
        databaseIngredientsUtils.insertIngredient(
            Ingredient(
                recipeId = recipe.recipeId,
                ingredientText = ingredient.ingredientText
            )
        )
    }
    databaseIngredientsUtils.deleteRecipeIngredients()

}
```

Programski kod 4.7. Stvaranje novog recepta

Problem stvaranja i brisanja sastojaka u receptu svodi se na brisanje ili stvaranje i pravilno postavljanje nove liste sastojaka na zaslon. Stvaranje sastojaka za recept vrši se preko funkcije *createRecipeIngredient* koja funkcionira na način kao i *createRecipeIngredientWithDelete*, ali bez brisanja. Brisanje sastojaka svodi se na pozivanje funkcije *deleteRecipeIngredient* kojoj je potreban identifikator sastojka koji se dobije odabirom nekog sastojaka i identifikator recepta kako bi se znalo s kojeg recepta se treba sastojak obrisati. Sastojak se briše pomoću *databaseIngredientUtils* funkcije *deleteIngredient* koja prima identifikator sastojka kao parametar i briše ga iz baze podataka. Nakon toga treba se pozvati funkcija *submitRecipeList* kako bi se osvježio zaslon bez obrisanog sastojka.

```
fun createRecipeIngredient(adapter: IngredientsRecyclerAdapter,ingredient: Ingredient, recipeId: Long = 0) {
    coroutineScope.launch { this: CoroutineScope
        databaseIngredientsUtils.insertIngredient(
            Ingredient(
                ingredientText = ingredient.ingredientText,
                recipeId = recipeId
            )
        )
        submitRecipeList(adapter,recipeId)
    }
}
fun deleteRecipeIngredient(ingredientId: Long, recipeId: Long = 0, adapter: IngredientsRecyclerAdapter,rootLayout : View) {
    deletePopup.showAtLocation(rootLayout, Gravity.CENTER, x: 0, y: 0)
    deletePopupView.yes_button.setOnClickListener { it: View!
        coroutineScope.launch { this: CoroutineScope
            withContext(Dispatchers.IO) { this: CoroutineScope
                databaseIngredientsUtils.deleteIngredient(ingredientId)
                submitRecipeList(adapter, recipeId)
            }
        }

        deletePopup.dismiss()

    }
    deletePopupView.no_button.setOnClickListener { it: View!
        deletePopup.dismiss()
    }
}
```

Programski kod 4.8. Brisanje i stvaranje sastojaka u receptu

4.3.5. Stvaranje novih recepata na temelju postojećih

Glavni problemi stvaranja recepta su objašnjeni u potpoglavlju 4.3.4. Postoje dva manja problema koja je potrebno riješiti. Prvi bi bio postavljanje zaslona da prikazuje točne parametre recepta koji se želi prilagoditi, a to se rješava funkcijom *setupUI* koja kao parametar prima *recipe* i *binding*.

Binding služi kao poveznica XML-a i fragmenta. Pomoću *bindinga* pristupa se svim elementima korisničkog sučelja i postavljaju se pravilne vrijednosti preko parametarskog recepta.

```
private fun setupUI(recipe: Recipe?, binding : CreateRecipeBinding) {
    binding.recipeNameEdit.setText(recipe!!.name)
    binding.typeOfMealEdit.setText(recipe.typeOfMeal)
    binding.timeToMakeEdit.setText(recipe.timeToMake)
    binding.descriptionEdit.setText(recipe.description)
    binding.photoUrlEdit.setText(recipe.photoUrl)
    binding.glutenCheck.isChecked = recipe.gluten
    binding.lactoseCheck.isChecked = recipe.lactose
    binding.caffeineCheck.isChecked = recipe.caffeine
    binding.fructoseCheck.isChecked = recipe.fructose
    binding.sulfiteCheck.isChecked = recipe.sulfite
    binding.veganCreateCheck.isChecked = recipe.vegan
    binding.vegetarianCreateCheck.isChecked = recipe.vegetarian
    binding.insertRecipeButton.setText("Update")
}
```

Programski kod 4.9. Postavljanje parametara zaslona

Funkcija *updateRecipe* prima kao parametar recept i profil koji se želi izmijeniti i profil koji radi izmjenu. Prvo, provjerava se je li profil isti kao i onaj koji je izvorno napravio recept da bi se znalo trebali li osvježiti odabrani recept ili stvoriti novi na temelju odabranog. Ako se pravi novi recept poziva se funkcija *insertRecipe* koja je ranije objašnjena, ako se osvježava stari poziva se *updateRecipe* funkcija koja osvježava postojeći recept s novim podacima.

```
fun updateRecipe(recipe : Recipe, profile: Profile){
    if(recipe.creator == currentRecipe.creator && recipe.name == currentRecipe.name){
        coroutineScope.launch { this: CoroutineScope
            databaseRecipeUtils.updateRecipe(recipe.name, recipe.photoUrl, timeToMake = recipe.timeToMake,
                typeOfMeal = recipe.typeOfMeal, description = recipe.description,
                gluten = recipe.gluten, fructose = recipe.fructose,
                lactose = recipe.lactose, caffeine = recipe.caffeine
                , sulfite = recipe.sulfite, vegan = recipe.vegan,
                vegetarian = recipe.vegetarian,recipeId = recipe.recipeId)
            withContext(Dispatchers.Main){ this: CoroutineScope
                navigateToDetailRecipe(recipe)
            }
        }
    }
    else{
                createRecipeViewModel.insertRecipe(Recipe(
                    name = recipe.name,
                    description = recipe.description,
                    timeToMake = recipe.timeToMake,
                    typeOfMeal = recipe.typeOfMeal,
                    gluten = recipe.gluten,
                    fructose = recipe.fructose,
                    lactose = recipe.lactose,
                    caffeine = recipe.caffeine,
                    sulfite = recipe.sulfite,
                    vegan = recipe.vegan,
                    vegetarian = recipe.vegetarian,
                    profileId = profile.profileId,
                    creator = recipe.creator
                ), view)

    }

}
```

Programski kod 4.10. Osvježavanje postojećeg recepta i stvaranje novog na temelju postojećeg

4.3.6. Dodavanje komentara na recepte

Komentari na receptima razdvojeni su na četiri problema. Dohvaćanje svih komentara na receptu odvija se pozivom funkcije *getAllComments* koja kao parametar prima identifikator recepta i dohvaća sve komentare pomoću *databaseCommentUtils* funkcije *getComments* vezane za određeni recept. Stvaranje novog komentara koje se odvija pozivanjem funkcije *insertComment* koja prima komentar kao parametar i provjerava je li tekst unutar komentara prazan, ako nije, stvara se komentar u bazi podataka.

```
fun getAllComments(recipeId: Long){
    coroutineScope.launch { this: CoroutineScope
        val list :List<Comment>? = databaseCommentUtils.getComments(recipeId)
        withContext(Dispatchers.Main){ this: CoroutineScope
            submitList(list)
        }
    }
}
fun insertComment(comment: Comment){
    if(comment.commentText != "") {
        coroutineScope.launch { this: CoroutineScope
            databaseCommentUtils.insertComment(
                Comment(
                    profileName = profile.profileName,
                    recipeId = comment.recipeId,
                    commentText = comment.commentText,
                    rating =  comment.rating
                )
            )
            val list :List<Comment>? = databaseCommentUtils.getComments(comment.recipeId)
            withContext(Dispatchers.Main) { this: CoroutineScope
                submitList(list)
            }
        }
    }
}
```

Programski kod 4.11. Prikaz dohvaćanja i stvaranja komentara

Uz mogućnost dodavanja komentara potrebna je mogućnost i brisanja komentara, a tako i osvježavanja starih. Brisanje i osvježavanje starih komentara omogućeno je samo onim profilima koji su stvorili komentar. Brisanje komentara odvija se pozivom funkcije *deleteComment* koja prima kao parametar odabrani komentar i preko *databaseCommentUtis*-ove *deleteComment* funkcije briše komentar iz baze podataka i osvježava listu adaptera. Problem osvježavanja teksta na postojećem komentaru rješava se pokretanjem funkcije *editComment* koja prima *commentId* i *text*. Funkcija preko *databaseCommentUtils*-a poziva funkciju *updateComment* koja prima iste parametre kao i glavna funkcija kako bi mogla pravilno pronaći komentar u bazi podataka i osvježiti ga s novim podacima.

```
fun deleteComment(comment: Comment){
    coroutineScope.launch { this: CoroutineScope
        databaseCommentUtils.deleteComment(comment.commentId)
        val list :List<Comment>? = databaseCommentUtils.getComments(recipe.recipeId)
        withContext(Dispatchers.Main){ this: CoroutineScope
            submitList(list)
        }
    }
}

private fun editComment(commentId: Long, text: String){
    coroutineScope.launch{ this: CoroutineScope
        databaseCommentUtils.updateComment(commentId, text)
        val list :List<Comment>? = databaseCommentUtils.getComments(recipe.recipeId)
        withContext(Dispatchers.Main){ this: CoroutineScope
            submitList(list)
        }
    }
}
```

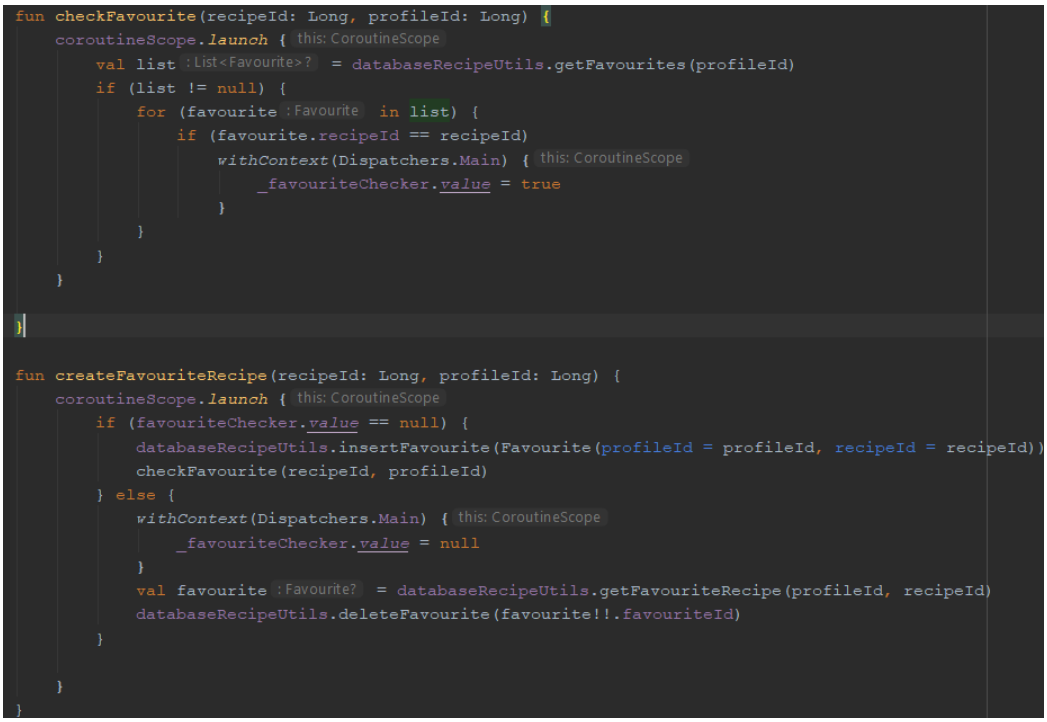
Programski kod 4.12. Osvježavanje i brisanje komentara

4.3.7. Dodavanje omiljenih recepata

Kao što je spomenuto u potpoglavlju 2.5.4., omiljeni je komponenta baze podataka što ovaj problem dodavanja omiljenih recepata svodi na problem stvaranja nove omiljene komponente koja sadržava poveznicu profila i recepta.

Pozivom funkcije *createFavouriteRecipe* kojoj se predaju identifikatori profila i recepta stvara se nova komponenta omiljenog preko *databaseRecipeUtils* funkcije *createFavourite*, a ako već postoji u bazi podataka takva komponenta, briše se preko *databaseRecipeUtils*-ove funkcije *deleteFavourite*.

Pomoć pri provjeri je li recept omiljen je *checkFavourite* funkcija kojom se dohvaća omiljena komponenta iz baze podataka i postavlja vrijednost *_favouriteChecker* tipa *MutableLiveData* na pravilnu vrijednost.



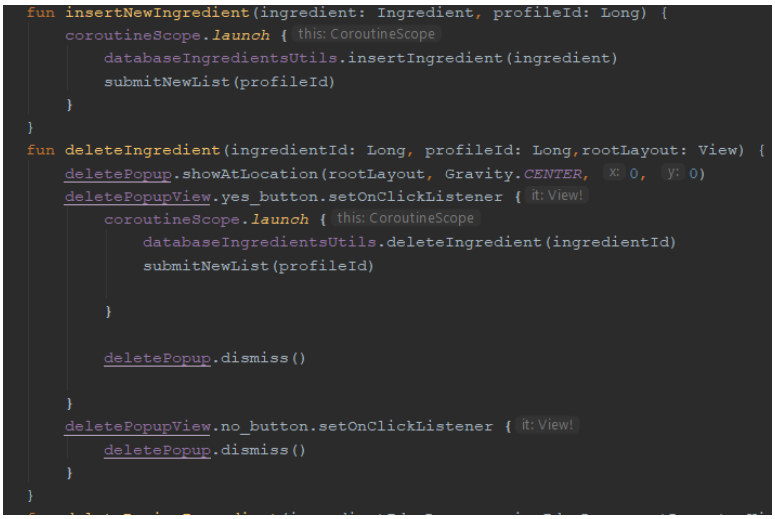
Programski kod 4.13. Dodavanje omiljenih recepata

4.3.8. Dodavanje sastojaka u košaricu

Problem dodavanja sastojaka u košaricu rješava se slično kao i problem s dodavanjem sastojaka recepta u potpoglavlju 4.3.4. Razlikuju se po tome što se sastojci ne moraju vezati samo za recept, nego se mogu vezati za profil kako je prikazano u potpoglavlju 2.5.4. Povezivanjem sastojaka za profil može se omogućiti da svaki profil ima svoju košaricu sastojaka.

S dodavanjem sastojaka mora se implementirati i brisanje sastojaka iz košarice. Stvaranje sastojaka izvodi se pokretanjem funkcije *insertNewIngerdient* koja preko *databaseIngredientsUtils*-a poziva funkciju *insertIngredient* koja pohranjuje sastojak u bazi podataka.

Brisanje se odvija pokretanjem funkcije *deleteIngredient* koja izbacuje iskočni prozor na kojem se može odabrati „da“ ili „ne“. Odabirom odgovora „da“ poziva se funkcija *deleteIngredient* koja prima identifikator sastojka na koji je korisnik odabrao i briše se iz baze podataka.



Programski kod 4.14. Prikaz dodavanja i brisanja sastojaka iz košarice

5. NAČIN KORIŠTENJA I ISPITIVANJA RADA APLIKACIJE

5.1. Način korištenja aplikacije

Aplikacija je podijeljena na pet glavnih dijelova. Dio s korisnicima, profilima, receptima, komentarima i košaricom. Dio s korisnicima podrazumijeva stvaranja novih korisnika i prijavu postojećih. Svaki korisnik pohranjuje se u lokalnu bazu podataka Sva logika koja se tiče stvaranja, spremanja i prijave korisnika nalazi se u tom dijelu aplikacije. Korisnički dio sastoji se od dva zaslona. To su zaslon za prijavu i zaslon za registraciju korisnika. Dio s profilima podrazumijeva stvaranje novih profila, prikaz postojećih i spremanje netolerancija i ograničenja profila. Svako ograničenje ili netolerancija koja je odabrana pri stvaranju profila pohranit će se u lokalnu bazu podataka Room. Dio s profilima sastoji se od tri zaslona. To su zaslon za stvaranje novog profila, prikaz postojećih i detalji o profilu. Time se veže za korisnički račun i prikazuje samo vlasniku profila. Odabirom nekog od postojećih profila prelazi se na dio s receptima koji obuhvaća cijeli proces stvaranja, osvježavanja, stvaranja novih na temelju postojećih recepata i predlaganje recepata korisničkom profilu. Logika za predlaganje recepata temelji se na provjeri što recept sadrži i ima li profil netoleranciju na nešto što recept sadrži. Dio s receptima sastoji se od pet zaslona. To su zaslon za omiljene recepte, svi recepti, predloženi recepti, zaslon za stvaranje novog recepta koji se koristi za osvježavanje starog recepta i stvaranje novog na temelju postojećeg te zaslon za detaljni prikaz recepta. S detalja o receptu moguće je prijeći na dio s komentarima koji obuhvaća sve komentare koji se nalaze na točno određenom receptu. Komentari se vežu preko recepata i korisničkih profila da bi se moglo znati tko je stvorio komentare i za koji recept. Time se omogućuje ispravljanje komentara te njihovo brisanje. Dio s komentarima sastoji se od jednog zaslona za prikaz komentara. Košarica kao dio aplikacije odnosi se na zaslon koji prikazuje sve sastojke vezane uz korisnički profil. Korisnik može stvoriti popis namirnica za kupovinu i brisati sastojke. Detaljniji opisi ovih dijelova i načina rada nalaze se u potpoglavlju 4.3., a prikaz zaslona i nekih slučajeva bit će prikazan u idućem potpoglavlju ovoga završnog rada.

5.2. Ispitivanje mobilne aplikacije

5.2.1. Postavke ispitivanja mobilne aplikacije

Prije samog ispitivanja mobilne aplikacije dodano je nekoliko recepata u bazu podataka da bi se moglo odmah testirati predlaganje recepata. Aplikaciju će se ispitati kroz dva testna slučaja koji se nalaze u sljedećim potpoglavljima. Baza podataka za korisnike je prazna, dok dio baze podataka za recepte sadrži nekoliko recepata. Cilj prvog testnog primjera je stvaranje korisničkog računa, profila s netolerancijom na kofein i laktozu, stvaranje recepta s greškom i osvježavanje recepta. Cilj drugog testnog primjera je na postojeći korisnički profil stvoriti profil koji je netolerantan na fruktozu, ispitati predlaže li se recept dodan u prvom testnom primjeru i nakon što se predloži iskomentirati ga, postaviti u omiljene i u košaricu dodati par sastojaka. Korisničko ime će se sastojati od korisničkog imena „zvonimir“ i lozinke „zvonimir“.

5.2.2. Ispitivanje prvog testnog slučaja

Prvo će se korisnik prijaviti s parametrima koji su spomenuti u potpoglavlju 5.2.1. Očekivano ponašanje aplikacije je da će se izbaciti iskočni prozor koji će reći da takav korisnik ne postoji.

Slika 5.1. Pokušaj prijave nepostojećeg korisnika

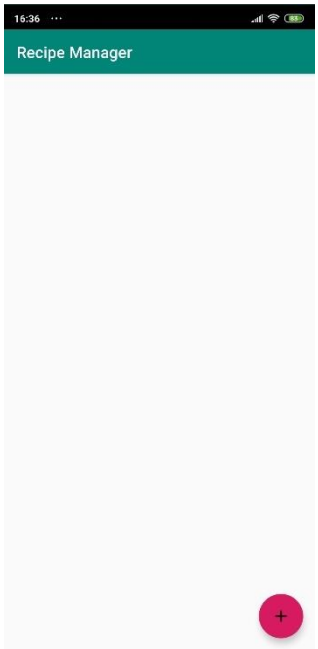
Nakon neuspjelog pokušaja prijave bez prijašnje registracije korisnik se treba registrirati. Registrirat će se korisnik s istim parametrima kao i za prijavu što bi trebalo proći po planu.

Slika 5.2. Registracija korisnika

Ponovnim pokušavanjem registriranje korisnika s takvim parametrima očekuje se izbacivanje iskočnog prozora da korisnik već postoji.

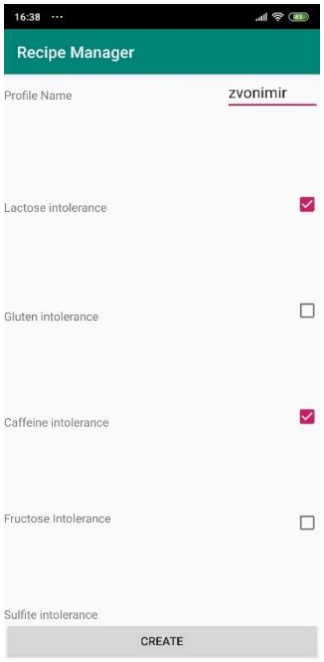
Slika 5.3. Pokušaj registiranja već registriranog korisnika

Nakon uspješne prijave korisnika se odvodi na zaslon s profilima koji treba biti prazan jer korisnik nema stvorenih profila.



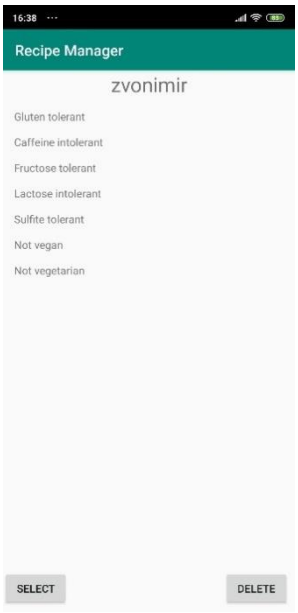
Slika 5.4. Prikaz zaslona nakon uspješne prijave korisnika

Odabirom plusa na zaslonu dolazi se na zaslon za stvaranje profila gdje će se stvoriti profil pod imenom „zvonimir“ koji će biti netolerantan na kofein i laktozu.



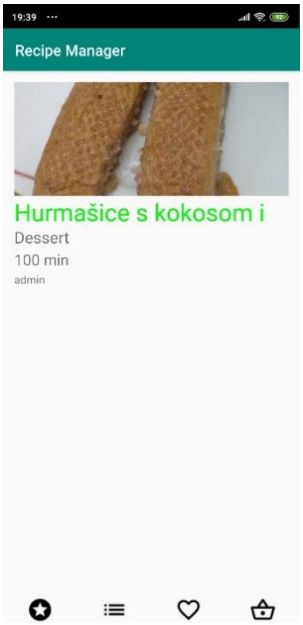
Slika 5.5. Stvaranje profila s netolerancijama i ograničenjima

Stvaranjem tog profila i povratkom na odabir može se vidjeti stvoreni profil. Odabirom profila otvara se novi zaslon na kojem se trebaju pravilno prikazati odabrani parametri tijekom stvaranja profila.



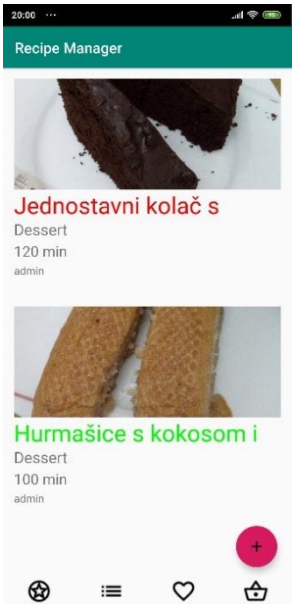
Slika 5.6. Profil sa svim potrebnim opisima netolerancija i ograničenja

Nakon uspješnog odabira profila korisnika se odvodi na zaslon s predloženim receptima za odabrani profil. Kao što je navedeno u potpoglavlju 5.2.1. već postoji nekoliko recepata pa se na predloženom zaslonu oni mogu pojaviti.



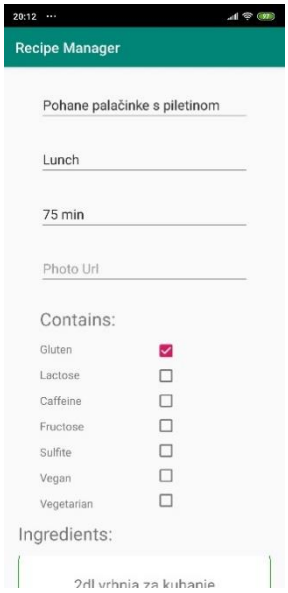
Slika 5.7. Prikaz zaslona s predloženim receptima

Odlaskom na zaslon sa svim receptima mogu se vidjeti i recepti koji nisu kompatibilni za korisnika i oni su označeni crvenom bojom.

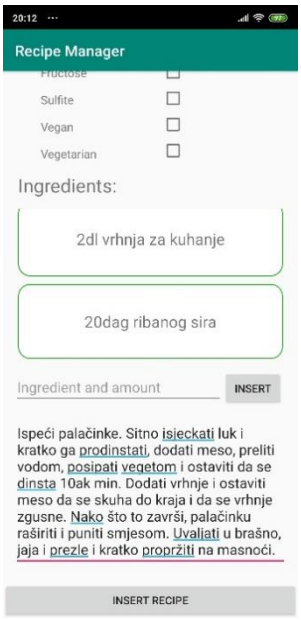


Slika 5.8. Prikaz zaslona sa svim receptima

Odabirom plusa odlazi se na zaslon za stvaranje novih recepata gdje je jedino obvezno polje ime recepta jer se ono provjerava u slučaju da već postoji takvo ime. U slučaju ispitivanja rada neće se unijeti slika da se pokaže postojanje slike kada se ne unese URL adresa slike recepta.



Slika 5.9. Prikaz dijela zaslona za stvaranje novog recepta



Slika 5.10. Prikaz ostatka zaslona za stvaranje novog recepta

Nakon uspješnog stvaranja recepta i odabirom stvorenog recepta dobiva se zaslon s detaljima recepta. Recept treba imati sliku koja je ugrađena u aplikaciju i označavati da slika nije unesena tijekom stvaranja recepta.

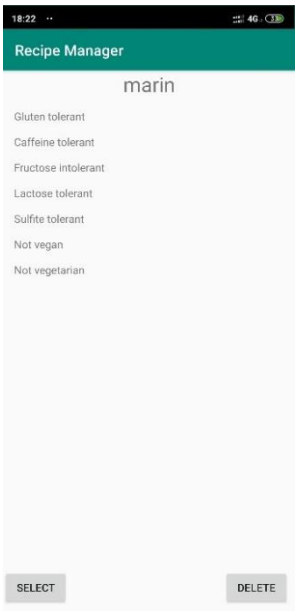
Slika 5.11. Prikaz zaslona detalja recepta

Ako je korisnik, kao što je u ovom slučaju, stvorio recept koji je odabrao, ima mogućnost brisanja recepta odabirom slike koša za smeće, mogućnost spremanja u omiljene recepte odabirom ikonice srca te ima mogućnost komentirati recept odabirom oznake za komentar. Ako korisnik želi urediti recept odabire olovku. U ovom slučaju korisnik želi urediti recept i postaviti sliku na recept.

Slika 5.12. Osvježeni recept sa slikom

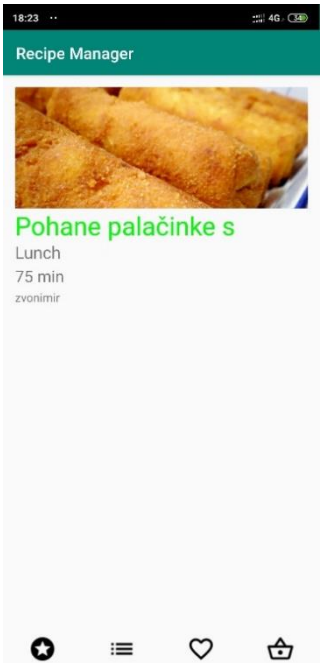
5.2.3. Ispitivanje drugog testnog slučaja

Kao što je već spomenuto u potpoglavlju 5.2.1., stvorit će se još jedan profil na prvotnom korisničkom računu koji je netolerantan na fruktozu.



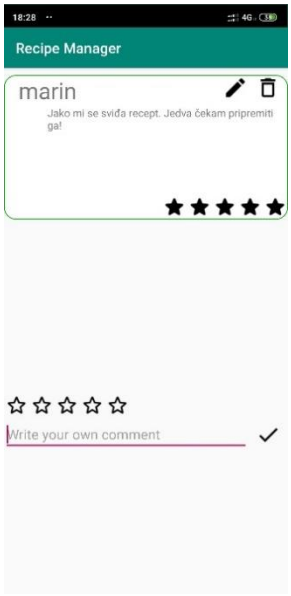
Slika 5.13. Drugi profil sa svim potrebnim opisima

Recept koji je dodan u prvom testnom primjeru predlaže se profilu iz drugog testnog primjera jer sadrži samo gluten, a u profilu „marin“ se navodi da podnosi gluten.



Slika Error! No text of specified style in document.**5.14.** Predloženi recept

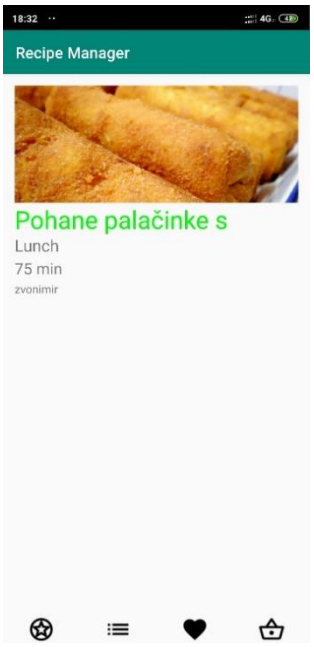
Odabirom predloženog recepta, dolazi se na zaslon s detaljima o receptu i potrebno je odabrati ikonicu srca kako bi se recept dodao u omiljene recepte. Odabirom ikonice komentara korisnika se odvodi na zaslon za komentiranje gdje ostavlja komentar na recept.



Slika 5.15. Komentiranje na recept

Povratkom na početni zaslon s receptima može se na navigacijskoj traci odabrati ikonica sa srcem koja odvodi na zaslon s omiljenim receptima.

Na zaslonu se prikazuje recept na kojem je odabrana ikonica srca kada se komentiralo na recept.



Slika 5.16. Omiljeni recepti

Korisnik si želi zapisati potrebne namirnice koje mu nedostaju za odabrani recept pa odabire košaricu u navigacijskoj traci i odlazi na zaslon s namirnicama. Na zaslonu s namirnicama korisnik odabirom plusa dodaje vrhnje za kuhanje i mlijeko kao namirnice koje mu nedostaju.

Slika 5.17. Košarica s dodanim namirnicama

5.3. Utjecaj predloška arhitekture MVVM na razvoj aplikacije

Prema [15], MVVM omogućava strukturiranje programskog koda na takav način da se podijeli odgovornost klasa i funkcija na pravilan način. Jedna od glavnih prednosti je čitljivost koda i lakše ispravljanje grešaka. Korištenjem MVVM predloška arhitekture dijelove koda puno je lakše testirati, upravo zbog podjele odgovornosti klasa. Odvajanjem *View* dijela od *ViewModel* dijela dobiva se na mogućnosti ponovnog korištenja *ViewModela* ili *Viewova* što uvelike olakšava proces stvaranja aplikacije.

Korišteni predložak arhitekture MVVM je znatno pridonio čitljivosti programskog koda i kvalitetnog organiziranja u datoteke, kao što je navedeno i u [17]. Primjenom arhitekture MVVM točno se zna koji dio programskog koda što radi i ne smije raditi stvari za koje nije namijenjen. Primjer toga je da se u *View* dijelu koriste stvari koje trebaju biti u *Model* dijelu, što se ne smije događati jer *ViewModel* treba biti posrednik između *Model* dijela i *View* dijela. Ispreplitanjem programskog koda *Model* dijela u *View* dijelu dolazi do komplikacije programskog koda, učestalog ponavljanja programskog koda i mnogo drugih loših navika. Korištenjem bilo koje arhitekture slijede se barem neka pravila programiranja što znatno poboljšava programski kod, njegovu čitljivost i funkcionalnost.

5.4. Korisničko iskustvo

Prema [16], korisničko iskustvo je iskustvo osobe nekog korisnika s aplikacijom. Često se UX poistovjećuje s UI-jem, ali postoji velika razlika gdje UX dizajnera zanima kako nešto funkcionira, a UI dizajnera kako nešto izgleda. To govori da će UX dizajner radije imati bolju funkcionalnost nego izgled. Glavni kriteriji korisničkog iskustva su intuitivnost i dizajn. Intuitivnost označava koliko je lako koristiti aplikaciju, koliko je prilagođena starijim osobama i osobama s lošijom opremom. Dizajn je bitna stavka svake aplikacije, ali ne može biti zamjena za intuitivnost korištenja.

Prema [17], korisničko iskustvo odnosi se na osjećaje i stavove neke osobe o određenom proizvodu, sustavu ili usluzi. Uključuje praktične, eksperimentalne, značajne i vrijedne poglede ljudskog korištenja računala. Korisničko iskustvo najviše se očituje u korisnosti proizvoda. Korisnost se može podijeliti na korisničko zadovoljstvo, testiranje proizvoda i prirodan UX. Korisničko zadovoljstvo očituje se izgledom aplikacije, ugodnosti koju pruža korisniku tijekom korištenja i brzine. Testiranje proizvoda bitan je dio UX-a jer se time dobivaju stvarne osobe koje testiraju proizvod i moguće je

dobiti povratne informacije. Prirodan UX označava pravilan slijed događaja na zaslonu pri čemu korisnik ne mora previše razmišljati gdje i kako treba otići.

Za povratnu informaciju oko korisničkog iskustva najbolje je dati osobama da isprobaju aplikaciju, te su iz tog razloga aplikaciju koristile tri osobe. Testiranje je trajalo 4 dana. Osobama su dane napomene da najviše pozornosti obrate na dizajn aplikacije, intuitivnost te da razmisle o sveukupnom zadovoljstvu korištenja aplikacije.

Tablica 5.1. Podaci o korisničkom iskustvu nakon testiranja

Godine	Spol	Dizajn	Intuitivnost	Komentari	Zadovoljstvo
25	M	3.5/5	4.5/5	„Dizajn je podbacio, ali funkcionalnosti su to nadoknadile.“	4/5
56	Ž	4/5	4/5	„Aplikacija mi je pomogla pri biranju hrane jer nisam morala toliko razmišljati kome što odgovara.“	4/5
31	M	4/5	5/5	„Intuitivna aplikacija koja osobama s netolerancijama kao što sam ja uvelike pomaže pri pronalasku prikladnih recepata.“	4.5/5

Prema podacima iz tablice 5.1. vidljivo je kako su najniže ocjene dobivene na temelju dizajna aplikacije što označava da bi se u idućim projektima na tome trebalo poraditi. Po ocjenama moguće je zaključiti da je aplikacija intuitivnija za osobe mlađe životne dobi nego za starije osobe koje možda nisu imale toliko doticaja s mobilnim aplikacijama. Sveukupno zadovoljstvo korištenja aplikacije je čvrsta četvorka. Korisnici su zadovoljni onime što aplikacija nudi i kako im može pomoći u daljnjem životu.

6. ZAKLJUČAK

Cilj ovoga završnog rada je istraživanje učestalih netolerancija i ograničenja prehrane kod ljudi, istraživanje recepata i digitalizacija navedene ideje u mobilnu Android aplikaciju. Istraživanjem literature koja se tiče netolerancija i ograničenja u prehrani kod ljudi došlo se do temeljnih spoznaja i poznavanja netolerancijskih problema ili ljudskih prehrambenih odluka što je omogućilo njihovo implementiranje u Android mobilnu aplikaciju. Da bi aplikacija što bolje izgledala i imala što bolje funkcionalnosti, bilo je potrebno istražiti trenutno dostupne mobilne aplikacije koje se temelje na istim ili sličnim principima. Nakon istraživanja predloženo je idejno rješenje aplikacije i način rada svih dijelova aplikacije. Implementacija ideje prikazana je u najbitnijim dijelovima programskog koda, dok je u posljednjem dijelu ovoga završnog rada prikazan način korištenja i testiranje aplikacije.

Ostvarena mobilna aplikacija omogućuje digitalizaciju svih recepata korisnika te lakše snalaženje među svim receptima preko profila kojima se predlažu recepti ovisno o ograničenjima i netolerancijama korisnika. Također, aplikacija korisniku omogućuje lakše zapisivanje potrebnih namirnica i detaljniji pregled recepata. Korisnik si može prilagođavati recepte drugih profila, ostavljati komentare na receptima što uvelike može pomoći osobi koja je stvorila recept ili drugim profilima koji su naišli na recept i žele vidjeti je li recept uistinu dobar.

Aplikacija ima za cilj omogućiti pomoć osobama s netolerancijama i ograničenjima u prehrani kako bi se lakše mogli snalaziti u svojim receptima i kako bi ih mogli imati u digitalnom obliku. Analiza načina korištenja i korisničko iskustvo pokazuju da je aplikacija uspjela u svojem cilju olakšavanja pripreme hrane testnim osobama te da je postigla zadovoljavajuću razinu intuitivnosti za razne dobne skupine. Aplikaciju je moguće unaprijediti s gledišta povezivanja s nekom bazom podataka koja nije lokalna tako da se omogući pregled recepata drugih mobilnih uređaja i omogući komunikacija između korisnika. Moguće ju je unaprijediti da bude reaktivnija, korištenjem RxKotlina, čime bi se mogle dodati i još neke funkcionalnosti, i dizajnom. Aplikacija je postavila dobru osnovu za stvaranje aplikacije koja bi mogla pomoći svim ljudima koji posjeduju Android pametni telefon i imaju ograničenja ili netolerancije koje su obrađene u radu.

LITERATURA

[1] M. Ispano, *Definition and classification of food allergy and intolerance,* 1998., https://www.sciencedirect.com/science/article/abs/pii/S0335745798800968, pristupljeno: 11.4.2020.

[2] H. Mackenzie, T. Dean, *Food allergies and food intolerances in children,* 2011., https://www.sciencedirect.com/science/article/pii/B9781845694319500057, pristupljeno: 11.4.2020.

[3] C. Ortolani, E. A. Pastorello, *Food allergies and food intolerances*, 2006., https://www.sciencedirect.com/science/article/abs/pii/S1521691805001836, pristupljeno: 11.4.2020.

[4] A. Szilagyil, C. Walker and M. G. Thomas, *Lactose Intolerance and Other Related Food Sensitivities,* 2019., https://www.sciencedirect.com/science/article/abs/pii/S1521691805001836, pristupljeno: 11.4.2020.

[5] D. P. Evatt and R. R. Griffiths, *Encyclopedia of Human Nutrition, 3rd edition: Caffeine,* str. 222. - 228., 2013., https://www.sciencedirect.com/referencework/9780123848857/encyclopedia-of-human-nutrition, pristupljeno 11.4.2020.

[6] N. L. Keim, P. J. Havel, *Encyclopedia of Human Nutrition, 3rd edition: Fructose*, str. 809. - 813., 2013., https://www.sciencedirect.com/referencework/9780123848857/encyclopedia-of-human-nutrition, pristupljeno: 11.4.2020.

[7] Z. Villines, *Fructose Intolerance*, 2020., https://www.medicalnewstoday.com/articles/fructose-intolerance, pristupljeno: 20.4.2020.

[8] A. Bjarnadottir, *The 14 Most Common Signs of Gluten Intolerance,* 2016., https://www.sciencedirect.com/science/article/abs/pii/S1521691805001836, pristupljeno: 20.4.2020.

[9] *Sulfite Sensitivity*, WebMD, 2018., https://www.webmd.com/allergies/sulfite-sensitivity, pristupljeno: 20.4.2020.

[10] J. Dwyer, *Encyclopedia of Human Nutrition, 3rd edition: Vegeterian Diets*, str. 829. - 834., 2013., https://www.sciencedirect.com/referencework/9780123848857/encyclopedia-of-human-nutrition, pristupljeno: 11.4.2020.

[11] A. Mazzocchi, C. Venter, K. Maslin, C. Agostoni, *The Role of Nutritional Aspects in Food Allergy: Prevention and Management*, 2017., https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5579643/, pristupljeno; 4.7.2020.

[12] M. Heller, *What is Kotlin? The Java Alternative Explained,* 2020., https://www.infoworld.com/article/3224868/what-is-kotlin-the-java-alternative-explained.html, pristupljeno: 7.6.2020.

[13] M. Sandoval, *Room – Kotlin, Android Architecture Components,* 2018., https://medium.com/mindorks/room-kotlin-android-architecture-components-71cad5a1bb35, pristupljeno: 7.6.2020.

[14] F. Muntenescu, *Android Architecture Patterns Part 3: Model – View- View Model*, 2016., https://medium.com/upday-devs/android-architecture-patterns-part-3-model-viewviewmodel-e7eeee76b73b, pristupljeno: 7.6.2020.

[15] A. Tewari, *Why You Should Use MVVM for Small Apps*, 2013., https://www.appliedis.com/whyyou-should-use-mvvm-for-smallapps/#:~:text=Model%2DView%2DViewModel%20(MVVM,the%20functionality%20of%20your%20application., pristupljeno 4.7.2020.

[16] W. Belk, *How do we evaluate User Experience (UX)?*, 2014., https://medium.com/@wbelk/how-do-we-evaluate-user-experience-ux-2fd3dbeabe17, pristupljeno: 30.6.2020.

[17] A. Smith, *What is User Experience? What Makes a Good UX Design?,* 2017., https://blog.prototypr.io/what-is-user-experience-what-makes-a-good-ux-designb404bb933bd0, pristupljeno: 30.6.2020.

SAŽETAK

Zadatak ovoga završnog rada je proučavanje teorijske podloge netolerancija i ograničenja prehrane kod ljudi te programsko ostvarenje mobilne Android aplikacije. Implementiranje mobilne aplikacije omogućuje stvaranje korisničkih profila koji korisnicima omogućuju odabir netolerancija i ograničenja prehrane korisnika i tako filtriranje recepata po prikladnosti za korisnika. U radu je opisano idejno rješenje aplikacije, model i građa aplikacije, te okruženje i jezici koji su korišteni pri izradi aplikacije s naglaskom na programski jezik Kotlin. Također, pobliže je opisan MVVM predložak arhitekture i njegov utjecaj na izradu aplikacije, objašnjen je način korištenja, te su analizirani rezultati ispitivanja rada mobilne aplikacije za različite testne slučajeve. Ta analiza pokazuje da je korisnicima olakšan proces svakodnevne pripreme hrane korištenjem mobilne aplikacije.

Ključne riječi: mobilna Android aplikacija, MVVM, netoleriranje hrane, prilagodljiva prehrana, recepti,

ABSTRACT

The aim of this final paper is to study the theoretical basis of food intolerance and dietary restrictions in humans and the implementation of the mobile Android application. Implementing a mobile application allows the creation of user profiles that allow users to select intolerances and dietary restrictions of users and thus filter recipes by user suitability. The paper describes the conceptual design of the application, the model and structure of the application, and the environment and languages used in the development of the application with an emphasis on the programming language Kotlin. Also, the MVVM architecture template and its influence on the application development are described in detail, the method of use is explained, and the results of testing the operation of the mobile application for different test cases are analyzed. This analysis shows that the mobile application has eased the process of daily food preparation for its users.

Key words: mobile Android application, MVVM, dietary restrictions, adaptive diet, recipes

ŽIVOTOPIS

Zvonimir Medak rođen je 25.8.1998. u Slavonskom Brodu. Pohađao je OŠ „Blaž Tadijanović“ u Slavnoskom Brodu, a srednju školu Gimnaziju „Matija Mesić“, također u Slavonskom Brodu. Trenutno je treća godina Računarstva na FERIT Osijek-u. Ima iskustva u programskom svijetu s raznim programskim jezicima kao što su C, C++, C#, Java, Kotlin i Swift. Glavni dio Android znanja postigao je kao samouk uz Udacity programe, a mali dio kolegijem na fakultetu. Posjeduje diplomu iOS akamdemije koju je organizirao Factory.hr gdje je dobio napredna znanja iOS-a i Swift-a. Timski rad mu nije stran, a ni pričanje na engleskom jeziku. Brzo se prilagodi novim okolinama i programskim jezicima jer ih ima nekolicinu iza sebe.

PRILOZI

Prilog 1. Završni rad u docx formatu

Prilog 2. Završni rad u pdf formatu

Prilog 3. Programski kod mobilne aplikacije