SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 794

MOBILNA APLIKACIJA ZA PRAĆENJE ZALIHA NAMIRNICA I PLANIRANJE OBROKA S UKLJUČENIM SUSTAVOM PREPORUČIVANJA RECEPATA

Marela Arambašić

Zagreb, srpanj 2025.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 794

MOBILNA APLIKACIJA ZA PRAĆENJE ZALIHA NAMIRNICA I PLANIRANJE OBROKA S UKLJUČENIM SUSTAVOM PREPORUČIVANJA RECEPATA

Marela Arambašić

Zagreb, srpanj 2025.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 3. ožujka 2025.

DIPLOMSKI ZADATAK br. 794

Pristupnica:	**Marela Arambašić (0036523688)**
Studij:	Računarstvo
Profil:	Programsko inženjerstvo i informacijski sustavi
Mentor:	izv. prof. dr. sc. Alan Jović

Zadatak: **Mobilna aplikacija za praćenje zaliha namirnica i planiranje obroka s uključenim sustavom preporučivanja recepata**

Opis zadatka:

U užurbanom suvremenom životu, upravljanje zalihama namirnica i planiranje obroka postaje sve veći izazov, što često rezultira lošijom prehranom i prekomjernim bacanjem hrane. U ovom diplomskom radu potrebno je osmisliti i razviti mobilnu aplikaciju za platformu Android koja će korisnicima omogućiti praćenje zaliha namirnica, planiranje obroka te preporučivanje recepata temeljeno na dostupnim namirnicama. Podatke o receptima potrebno je dohvaćati iz javno dostupne baze (npr. Spoonacular Food API) pomoću REST API-ja. Korisničko sučelje treba omogućiti pregled svih dostupnih recepata, praćenje zaliha namirnica uz mogućnosti izmjena količina, planiranje obroka te sustav za preporuku recepata. Za implementaciju sustava preporuke recepata koristit će se tehnike strojnog učenja, uzimajući u obzir različite faktore, poput korisničkih interakcija (npr. praćenje klikova), dostupnih namirnica i označenih omiljenih recepata, kako bi se generirali personalizirani prijedlozi recepata. Izbor programskog jezika, razvojnog okruženja i baze podataka je proizvoljan, ali izbor treba opravdati iz perspektive zahtjeva na sustav.

Rok za predaju rada: 4. srpnja 2025.

Ovim putem želim zahvaliti svojim roditeljima, obitelji i prijateljima na podršci, razumijevanju i poticajima tijekom cijelog trajanja mog studija. Posebno zahvaljujem svom mentoru, izv. prof. dr. sc. Alanu Joviću, na stručnom vodstvu, pomoći pri izradi ovog rada i usmjeravanju tijekom fakultetskog obrazovanja. Bez njihove podrške ovaj rad ne bi bio moguć.

Sadržaj

Uvod

U današnjem ubrzanom tempu modernog života, briga o vođenju zaliha namirnica, planiranju prehrane, odabiru recepata i racionalnim korištenjem postojećih zaliha hrane postaje sve veći izazov. Nedostatak strukturiranih načina za upravljanjem namirnicama u kućanstvu dovodi do situacija u kojima sve više hrane u smočnicama i hladnjacima ostaje neiskorišteno, pokvari se i završi kao otpad. Takva situacija nije samo financijski loša za kućanstvo, već predstavlja i ozbiljan ekološki izazov. Prema procjeni Švedskog instituta za hranu i biotehnologiju, godišnje se baci više od 30 posto ukupno proizvedene hrane u svijetu [1].

Stoga je cilj ovog rada istražiti i razviti mobilnu aplikaciju koja omogućuje učinkovito upravljanje zalihama namirnica, planiranja obroka te preporuke recepata na temelju dostupnih sastojaka, uzimajući u obzir podatke o prehrambenim preferencijama korisnika.

Na tržištu postoje aplikacije koje djelomično rješavaju navedene probleme, kao što su [2] [3]:

- *Paprika*, aplikacija koja omogućuje organizaciju recepata, kreiranje planova prehrane i izradu lista za kupovinu, dostupna na nekoliko platformi [4]
- *Mealime*, aplikacija s fokusom na planiranje obroka te automatskom generiranju popisa za kupovinu s ciljem smanjenja bacanja hrane [5]
- *Whisk*, aplikacija koja koristi umjetnu inteligenciju za personalizaciju recepata i planova prehrane [6]
- *Yummly*, jedna od najpopularnijih aplikacija za upravljanje receptima s velikom bazom podataka i funkcionalnostima povezivanja s aplikacijama za naručivanje namirnica, no koja je od 2024. godine prestala s radom nakon promjene vlasnika [7] [8]

Za razliku od navedenih rješenja, ovaj rad nastoji razvojem aplikacije pružiti sveobuhvatno rješenje koje objedinjuje upravljanje zalihama namirnica, planiranje prehrane i preporuke recepata. U prvom dijelu rada predstavit će se vrste sustava preporuke te analiza implementacije primijenjenog modela. Zatim slijedi prikaz programske potpore i arhitekture mobilne aplikacije, kao i detaljna implementacija razvijena za Android platformu. Na kraju rada analizirat će se mogućnosti budućih poboljšanja i proširenja funkcionalnosti aplikacije.

1. Sustavi preporuka

Sustavi preporuka (engl. *recommender systems*) su algoritamski sustavi za filtriranje velike količine informacija čija je svrha personalizirano predviđanje ocjena ili preferencija korisnika za određeni sadržaj. Algoritmi obrađuju podatke na osnovi prijašnjih kupovina, ocjena, povijesti pregledavanja, korisničkih klikova i sličnih aktivnosti. Cilj je identificirati obrasce i korisničke navike te, na temelju toga, predložiti sadržaj koji bi korisnika mogao najviše zanimati.

Ovi sustavi široko su rasprostranjeni u današnjem digitalnom okruženju, posebno na društvenim mrežama i platformama za distribuciju digitalnih sadržaja kao što su Netflix, Amazon i YouTube. Ključni su u pomaganju korisnicima da pronađu relevantan sadržaj u moru informacija.

Postoje tri osnova pristupa izradi sustava preporuke [9]:

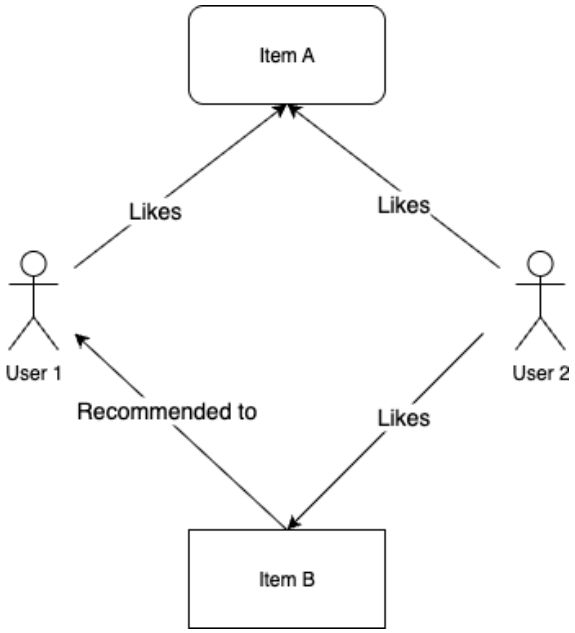
1. Kolaborativno filtriranje
2. Filtriranje sadržaja
3. Hibridni sustavi preporuke

1.1 Kolaborativno filtriranje

Kolaborativno filtriranje je pristup koji preporuke temelji na obrascima ponašanja korisnika i njihovim karakteristikama. Algoritam polazi od pretpostavke da su korisnicima zanimljivi sadržaji koje su pozitivno ocijenili ili koji su pozitivno ocijenjeni od korisnika sa sličnim interesima.

Postoje dvije glavne vrste kolaborativnog filtriranja [10]:

1. Korisnik – korisnik *(user-user)*: Sustav povezuje korisnike sličnih ponašanja ili ocjena te na temelju toga ciljanom korisniku preporučuje sadržaj koji sam korisnik još nije isprobao. Na slici 1.1 se nalazi primjer ovog postupka.
2. Proizvod – proizvod *(item-item)*: Sustav preporučuje korisniku sadržaje koji su slični onima s kojima je već imao pozitivnu interakciju

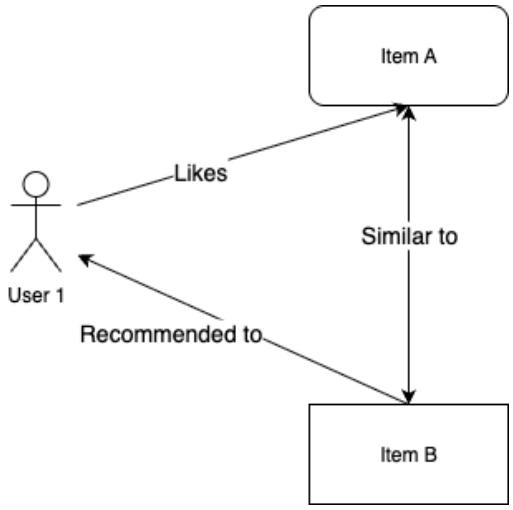


Slika 1.1 Primjer kolaborativnog filtriranja na temelju sličnosti korisnika

Glavni nedostatak kolaborativnog filtriranja je tzv. problem hladnog starta, odnosno situacija kada sustav nema dovoljno podataka o novim korisnicima ili novim sadržajima. U takvim slučajevima sustav nije sposoban generirati personalizirane preporuke dok se ne prikupi dovoljan broj podataka. Ovaj problem posebno dolazi do izražaja kod novih proizvoda ili tehnologija, što može rezultirati sniženom kvalitetom u početnoj fazi korištenja.

1.2 Filtriranje sadržaja

Algoritam filtriranja sadržaja (engl. *content-based filtering*) temelji se na analiziranju značajki i atributa sadržaja koji se preporučuje. Ovaj pristup analizira atribute i ključne riječi povezane s predmetima, kao što su naslovi, opisi i značajke, te identificira sličnosti između tih sadržaja i novih predmeta koji bi korisnika mogli zanimati. Primjer primjene algoritma nalazi se na slici 1.2.



Slika 1.2 Primjer filtriranja sadržaja

Glavna prednost ovog pristupa je da nije ovisan o ponašanju drugih korisnika, što ga čini pogodnim za primjenu u ranim fazama razvoja aplikacija s malim brojem korisnika ili interakcija.

Neke od najčešće korištenih načina mjerenja sličnosti su [11]:

- Kosinusna sličnost: mjeri kut između dva vektora
- Euklidova udaljenost: mjeri udaljenosti između dva vektora u prostoru
- TD-IDF (Term Frequency–Inverse Document Frequency): tehnika za određivanje važnosti riječi u tekstu

1.3 Hibridni sustavi preporuke

Hibridni sustavi preporuke kombiniraju metode kolaborativnog i sadržajnog filtriranja kako bi se povećala preciznosti i relevantnosti preporuka. Ovakav pristup nastoji iskoristiti prednosti oba pristupa te riješiti se problema poput hladnog starta i situacija u kojoj sustavi predlažu uvijek iste ili slične sadržaje (problem zatvaranja u krug).

Neki od načina kombiniranja metoda su [12]:

- *Weighted:* sustav implementira dva ili više modela te njihovim rezultatima pridoda određenu težinu čime se dobiva konačna lista preporuka

- *Switching*: sustav dinamički bira metodu preporuke koju će primijeniti ovisno o situaciji, primjerice koristi sadržajno filtriranje kod novih korisnika
- *Mixed*: sustav kombinira preporuke različitih modela u jedan skup prijedloga
- *Cascade*: sustav prvo primjenjuje jednu metodu, zatim se rezultati dodatno obrađuju i rangiraju drugom metodom
- Itd.

Iako hibridni sustavi daju bolje rezultate, njihova implementacija je složenija te zahtijeva veću podatkovnu i računalnu infrastrukturu.

1.4 Izbor algoritma za preporuku

U tablici 1.1 prikazana je usporedba tri osnovna pristupa izradi sustava preporuke: kolaborativno filtriranje, filtriranje sadržaja i hibridni sustavi. Svaki od navedenih pristupa ima svoje prednosti i mane.

Tablica 1.1 Usporedba algoritama za preporuku

Kolaborativno filtriranje	Filtriranje sadržaja	Hibridni sustavi
Analiza ponašanja i ocjena korisnika	Analiza značajki i atributa sadržaja	Kombinacija metoda kolaborativnog filtriranja i filtriranja sadržaja
Problem hladnog starta za nove korisnike	Ograničenost na sličnost prema postojećim atributima	Složenija i skuplja implementacija
Potrebna velika količina podataka o korisnicima	Potrebni podatci o značajkama stavki	Potrebna velika količina podataka o korisnicima i o značajkama
Može otkriti neočekivane veze između korisnika i sadržaja	Nije ovisan o ponašanju drugih korisnika	Bolja preciznost

Pogodan za velike platforme s mnogo korisnika	Pogodan za primjenu u ranijim fazama razvoja	Pogodan za kompleksne sustave s mnogo podataka

Kolaborativno filtriranje temelji se na analizi ponašanja i ocjena korisnika s ciljem identificiranja sličnih obrazaca interesa među korisnicima. Ovaj pristup može otkriti neočekivane veze između korisnika i sadržaja, što doprinosi kvaliteti preporuka, osobito na velikim platformama s mnogo korisnika. Međutim, nedostatak mu je potreba za velikom količinom korisničkih podataka te problem hladnog starta, koji otežava stvaranje preporuke za nove korisnike.

S druge strane, filtriranje sadržaja analizira značajke i atribute samog sadržaja, poput sastojaka i vrste kuhinje. Prednost ovog pristupa jest što nije ovisan o povijesti korisničkih interakcija, čime je pogodan za rane faze razvoja aplikacije ili sustava s ograničenim korisničkim podacima. Ograničenje mu je što može preporučivati samo sadržaje slične onima koje je korisnik prethodno pretraživao ili ocijenio.

Hibridni sustavi pokušavaju iskoristiti prednosti oba pristupa kombiniranjem kolaborativnog i sadržajnog filtriranja, čime nastoje poboljšati preciznost preporuka i smanjiti probleme poput hladnog starta i zatvaranja u krug. Iako hibridni sustavi daju povoljnije rezultate, njihova implementacija je složenija i računalno zahtjevnija.

S obzirom na navedene karakteristike sustava preporuka i ciljeve projekta, odabrana je metoda filtriranja sadržaja kao osnovna metoda u ovom radu. Ovaj pristup omogućuje iskorištavanje konkretnih atributa recepata, poput sastojaka i vrste kuhinje, pružajući personalizirane preporuke bez potrebe za opsežnim podacima o korisnicima. Izbor također pomaže u prevladavanju problema hladnog starta, koji je česta prepreka pri korištenju kolaborativnog filtriranja.

2. Implementacija sustava za preporuku

U ovom poglavlju opisana je implementacija sustava za preporuku, uključujući proces prikupljanja i obrade podataka, izgradnje, učenja i ispitivanja modela za preporuku te implementacije na platformi Railway.

Sustav je razvijen kao samostalna poslužiteljska usluga koja koristi arhitekturu REST, pri čemu je za izgradnju web sučelja korištena knjižnica *Flask* u programskom jeziku Python. Tako je s pomoću zahtjeva HTTP POST aplikaciji omogućen pristup modelu.

Model je naučen koristeći knjižnicu *scikit-learn* koja se koristi za strojno učenje.

2.1 Prikupljanje i priprema podataka

Za učenje modela potreban je veliki skup recepata. Za prikupljanje recepata korišten je javno dostupan *endpoint* Spoonacular API-ja, *"recipes/random"*.

Zbog ograničenja vanjske usluge, koja u jednom pozivu može vratiti maksimalno 100 recepata, skripta za dohvaćanje podataka šalje niz uzastopnih zahtjeva kako bi se prikupila što veća količina recepata. Svi dohvaćeni recepti se zatim parsiraju, kao što je vidljivo na slici 2.1, te spremaju u JSON datoteku.

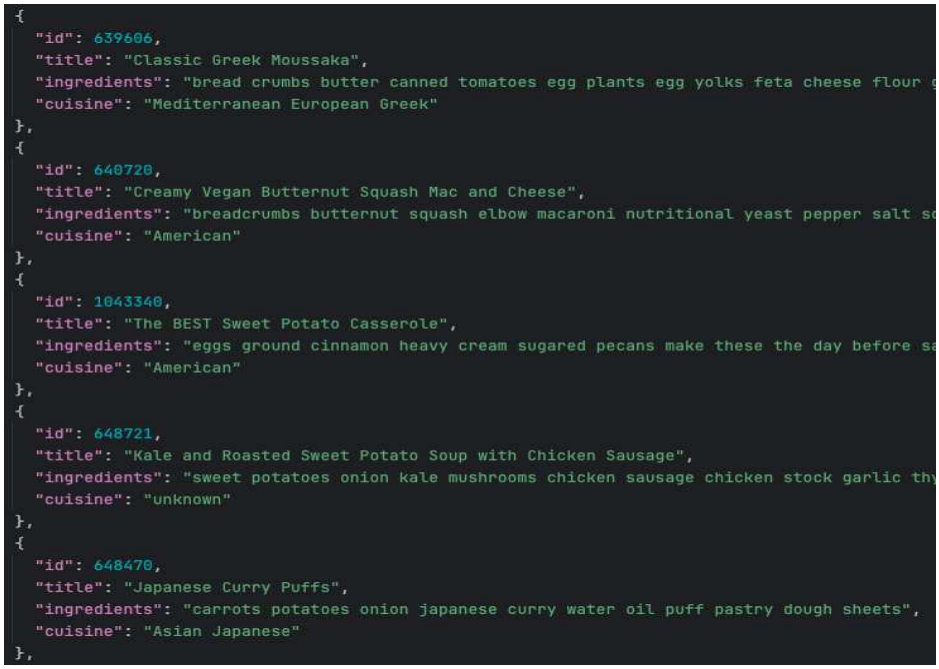
```
def parse_recipe(recipe):
    return {
        'id': recipe['id'],
        'title': recipe['title'],
        'ingredients': ' '.join([ing['nameClean'] for ing in recipe.get('extendedIngredients', [])]),
        'cuisine': ' '.join(recipe.get('cuisines', [])) or 'unknown',
    }
```

Slika 2.1 Parsiranje recepata

Za svaki dohvaćeni recept, spremaju se samo polja potrebna za učenje modela, kao što je vidljivo na slici 2.2:

- *Id* -> jedinstveni identifikator recepta
- *Title* -> Naslov

- *Ingredients* -> Povezana lista naziva sastojaka
- *Cuisine* -> Vrsta kuhinje



Slika 2.2 Pregled prikupljenih recepata

2.2 Model

Za stvaranje i učenje modela za preporuku recepata korištena je popularna knjižnica za strojno učenje *scikit-learn*. Model je naučen koristeći tekstualne podatke o sastojcima i vrstama kuhinja recepata.

Za pretvaranje teksta u numerički oblik korišten je vektorizator TD-IDF (*Term Frequency – Inverse Document Frequency),* dok je za klasifikaciju odabrana logistička regresija, kao što je prikazano na slici 2.3.

```
model = Pipeline(
    [
        ('vectorizer', TfidfVectorizer(max_features=5000)),
        ('classifier', LogisticRegression(max_iter=1000, class_weight='balanced'))
    ]
)
```

Slika 2.3 Izgradnja modela za preporuku

2.2.1 TD-IDF

TD-IDF je statistička metoda koja mjeri važnost riječi unutar jednog dokumenta u odnosu na cijeli skup dokumenata. Time se smanjuje utjecaj čestih, manje informativnih riječi, a naglašavaju se značajni izrazi specifični za određeni recept.

Maksimalan broj značajki postavljen je na 5000 zbog specifične domene i ograničenog vokabulara, što dodatno doprinosi učinkovitosti modela.

2.2.2 Logistička regresija

Za klasifikaciju je odabrana logistička regresija, model koji je prikladan za prediktivne zadatke jer daje vjerojatnosti pripadnosti uzorku određenoj klasi. Parametar *max_iter* postavljen je na 1000 kako bi se osigurao dovoljan broj iteracija bez prekomjernog opterećivanja resursa.

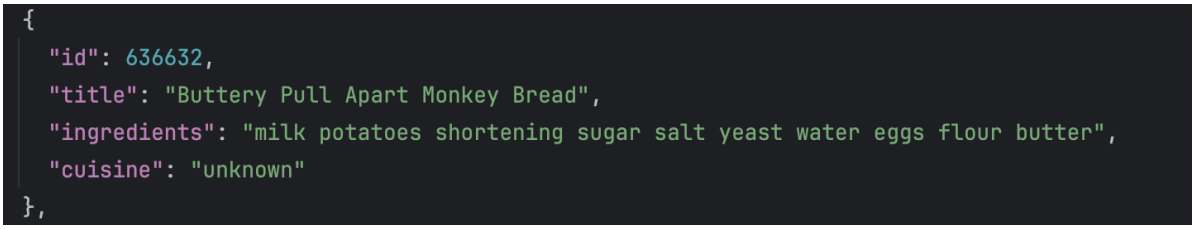
Također, primijenjena je opcija balansiranja klasa zbog neravnomjerne distribucije korisničkih oznaka, što omogućuje bolju točnost i kod manje zastupljenih primjera.

2.3 Učenje

Učenje modela se provelo koristeći prethodno prikupljene recepte spremljene u datoteku *spoonacular_recipes.json.* Zbog nedostupnosti stvarnih podataka o korisnicima i njihovim interakcijama, potrebno je simulirati korisničke profile radi stvaranja dovoljnog skupa podataka za učenje i ispitivanje. U simulaciji je generirano 5000 korisnika, pri čemu je svakom korisniku nasumično dodijeljeno 5 označenih recepata.

Nasumični odabir koristi se kao praktičan kompromis za simulaciju različitih korisničkih preferencija, koje u stvarnosti variraju od sklonosti receptima slične vrste kuhinje i sastojaka

do privrženosti receptima bez izražene povezanosti. Primjer tako odabranog recepta vidljiv je na slici 2.4.



Slika 2.4 Primjer nasumično odabranog recepta

Zatim su nakon odabira recepata iterirani svi ostali recepti kao kandidati za preporuku kako bi se procijenila sličnost između sastojaka i vrste kuhinje označenih recepata i recepata kandidata.

Sličnost se izračunava funkcijom *ingredient_overlap* koja mjeri preklapanje sastojaka, dok je dodatno uvezen bonus od 0.25 bodova za isti tip kuhinje. Ako ukupni rezultat sličnosti premaši prag od 0.45, kandidat je označen kao pozitivan primjer (oznaka 1), u protivnom kao negativan primjer (oznaka 0).

Prag od 0.45 odabran je kao optimalni kompromis između preniske vrijednosti, koja bi rezultirala velikim brojem lažno pozitivnih rezultata (prevelik broj recepata smatrao bi se sličnima na temelju malog broja preklapanja sastojaka) te previsoke vrijednosti, koja bi rezultirala premalim brojem sličnih recepata (zahtijevala bi preveliko podudaranje sastojaka). Kao vrijednosti praga su, osim 0.45, ispitane i vrijednosti 0.2, 0.3 i 0.5.

Bonus za isti tip kuhinje dodan je kako bi se poboljšala kvaliteta sličnosti, osobito u slučajevima kada sama usporedba sastojaka nije dostatna za prelazak praga.

Navedena funkcionalnost je prikazana na slici 2.5.

```python
for _ in range(NUM_USERS):
    bookmarks = random.sample(all_recipe_ids, BOOKMARKS_PER_USER)
    bookmark_ingredients = [id_to_ingredients[rid] for rid in bookmarks]
    bookmark_cuisines = [id_to_cuisine[rid] for rid in bookmarks]

    for rid in all_recipe_ids:
        if rid in bookmarks:
            continue

        candidate_ing = id_to_ingredients[rid]
        candidate_cuisine = id_to_cuisine[rid]

        max_similarity = 0
        for i, b_ing in enumerate(bookmark_ingredients):
            sim = ingredient_overlap(candidate_ing, b_ing)
            if candidate_cuisine == bookmark_cuisines[i]:
                sim += CUISINE_BONUS
            max_similarity = max(max_similarity, sim)

        label = 1 if max_similarity >= SIMILARITY_THRESHOLD else 0

        X.append(id_to_text[rid])
        y.append(label)
```

Slika 2.5 Prikupljanje podataka o sličnosti recepata

Dobiveni podatkovni skup čine tekstni prikazi recepata (x) i pripadajuće oznake (y) za sve korisnike. Podaci su podijeljeni na skup za učenje (80 %) i skup za ispitivanje (20 %) koristeći funkciju *train_test_split.*

Nakon učenja, model je spremljen lokalno u datoteku *recipe_recommender.pkl* koristeći knjižnicu *joblib*.

Tijekom učenja generiran je izvještaj o provedenom učenju i evaluaciji. Na slici izvještaja 2.6 prikazani su sljedeći podatci:

- Ukupno je generirano je 3835000 uzoraka, od čega je 463502 pozitivnih primjera, odnosno primjera s oznakom 1
- Preciznost modela za klasu nerelevantnih recepata (0) je vrlo visoka (0.97), što znači da modeli rijetko označavaju nerelevantne recepte kao relevantne.
- Za klasu relevantnih recepata (1) preciznost je niža (0.30), ali model ima visok odziv (engl. *recall*) od 0.82, što znači da pronalazi većinu relevantnih recepata unutar svih stvarno relevantnih.

- Ukupna točnost modela je 75 %

Model pokazuje jaku konzervativnost prema negativnoj klasi, smanjujući broj lažno pozitivnih rezultata. Odnosno, sustav izbjegava preporučivati recepte koji imaju mali stupanj preklapanja sastojaka i ne dijele vrstu kuhinje, te stoga nisu relevantni korisniku. Ovakav pristup doprinosi poboljšanju korisničkog iskustva i jačanju povjerenja u sustav.

Visoki odziv za pozitivnu klasu osigurava da većina relevantnih recepata bude prepoznata, što je ključno za dobru funkcionalnost sustava preporuka.

```
Loaded 1000 recipes
Generated dataset with 3835000 samples. Positive examples: 463502
Training model...
Evaluating model...
              precision    recall  f1-score   support

           0       0.97      0.74      0.84    674500
           1       0.30      0.82      0.44     92500

    accuracy                           0.75    767000
   macro avg       0.63      0.78      0.64    767000
weighted avg       0.89      0.75      0.79    767000
```

Slika 2.6 Izvještaj o provedenom treningu i evaluaciji

2.4 Ispitivanje

Ispitivanje prethodno generiranog modela izvršilo se s pomoću Python skripte koja učitava model i potom koristi listu recepata kako bi izvršila i ispisala predikcije.

Skripta prvo učitava datoteku s listom recepata, te zatim nasumično odabire 3 recepta kao korisnikove označene recepte. Potom odabire 20 nasumičnih recepata (isključujući označene recepte) i tekstno kombinira sastojke i vrste kuhinja recepata sa svakim od 20 izabranih recepata kandidata.

Učitanom modelu se prosljeđuju tekstni kandidati za predikciju, koji se naposljetku sortiraju po vjerojatnosti koju je model predvidio.

```
User bookmarks:
- 716379 Chocolate Coconut Banana Bread (unknown)
- 715562 Loaded Baked Potato Soup (unknown)
- 652433 Moroccan Lemon Shish Kebabs (unknown)

Model predictions (sorted by confidence):
729530 Upside-Down Blueberry Puffs Brunch (unknown): Likely liked (confidence: 0.99)
647210 Honey Dijon Roasted Brussels Sprout (unknown): Likely liked (confidence: 0.99)
645419 Green Beans With Roasted Walnuts and Sweet Cranberries (unknown): Likely liked (confidence: 0.99)
632660 Apricot Glazed Apple Tart (unknown): Likely liked (confidence: 0.99)
661531 Steak with lemon and capers (unknown): Likely liked (confidence: 0.99)
1017374 Easy Weekday Breakfast Muffins (unknown): Likely liked (confidence: 0.99)
636199 Broccoli Oatmeal Breakfast Casserole (unknown): Likely liked (confidence: 0.99)
663833 Triple Chocolate Whoppers (unknown): Likely liked (confidence: 0.99)
631785 Impossible" Coconut Pie (unknown): Likely liked (confidence: 0.99)
637638 Cheesy Cauliflower (unknown): Likely liked (confidence: 0.99)
649690 Lemon Mint Sorbet (unknown): Likely liked (confidence: 0.99)
660670 Sourdough Stuffing with Sage Sausage and Apples (unknown): Likely liked (confidence: 0.98)
715397 Cheesy Chicken and Rice Casserole (unknown): Likely liked (confidence: 0.98)
634389 Basic Hummus (Middle Eastern): Likely liked (confidence: 0.98)
665029 Watermelon Popsicles with Mint, Basil & Lime (unknown): Likely liked (confidence: 0.98)
646512 Salmon Caesar Salad (American): Likely liked (confidence: 0.95)
661741 Strawberry and Chocolate Chip Panini (unknown): Likely liked (confidence: 0.95)
631747 Dutch Oven Paella (Spanish European): Likely liked (confidence: 0.92)
735820 Crock-Pot: Asian-Style Country Ribs with Black Bean Garlic Sauce (Asian): Likely liked (confidence: 0.91)
663151 Thai Shrimp (Thai Asian): Likely liked (confidence: 0.80)
```

Slika 2.7 Rezultat ispitivanja modela

2.5 Sustav Railway

Web aplikacija za sustav preporuke implementirana je koristeći radni okvir *Flask* u programskom jeziku Python. U glavnoj implementacijskoj datoteci aplikacija učitava prethodno naučen model, spremljen u datoteku *recipe_recommender.pkl*, i izlaže ga putem metode HTTP POST na REST *endpoint* imena "/*predict*".

Tijelo zahtjeva sadrži podatke o kandidatima za preporuku i označenim receptima koji sadrže pripadajuće sastojke i vrste kuhinja kao što je prikazano na slikama 2.8 i 2.9. Ovi podaci se prosljeđuju modelu koji izračunava i vraća preporuke sortirane po vjerojatnosti da će se korisnicima svidjeti.

```
@InternalSerializationApi @Serializable  6 Usages
data class SuggestRecipesRequest(
    val bookmarks: List<SuggestRecipeRequest>?,
    val candidates: List<SuggestRecipeRequest>?
)
```

Slika 2.8 Tijelo zahtjeva za predikciju recepata

```
@InternalSerializationApi @Serializable  15 Usages
data class SuggestRecipeRequest(
    @SerialName( value = "cuisines")
    val cuisines: List<String>?,
    @SerialName( value = "extendedIngredients")
    val extendedIngredients: List<ExtendedIngredientsResponse>?,
)
```

Slika 2.9 Prikaz detalja o receptima koji se šalju u tijelu zahtjeva za predikciju

Za *deployment* web aplikacije i omogućavanje njenog rada u oblaku odabrana je platforma Railway. Railway omogućuje brzo i jednostavno postavljanje i upravljanje web aplikacijama pisanima u programskom jeziku Python bez potrebe za kompleksnom infrastrukturom.

Sustav Railway zahtijeva poveznicu na javno dostupan repozitorij na platformi GitHub koji pored glavnog koda ima sve potrebne datoteke za pokretanje usluge, uključujući *recipe_recommender.pkl te requirements.txt* datoteku koja sadrži popis svih paketa koje aplikacija koristi, i datoteku *Procfile* koja definira naredbu za pokretanje aplikacije u okruženju *Railway*.

3. Specifikacija programske potpore

Ovo poglavlje opisuje funkcionalne zahtjeve mobilne aplikacije te sukladna ograničenja kako bi se definirali ciljevi i granice sustava.

3.1 Funkcionalni zahtjevi

Aplikacija *PantryMeals* razvijena je kao mobilna aplikacija za Android za upravljanje kućnim zalihama namirnica, pregled recepata te personaliziranu preporuku recepata na temelju dostupnih sastojaka i korisničkih preferencija.

Ključne funkcionalnosti aplikacije su:

F1. Pregled svih dostupnih recepata

- Korisnik mora imati pristup potpunoj bazi recepata dohvaćenoj preko integriranog Spoonacular API-ja
- Omogućena je funkcionalnost pretraživanja recepata po nazivima

F2. Pregled detalja recepta

- Za svaki recept aplikacija prikazuje sljedeće detalje, ako su dostupni:
    - Naslov recepta
    - Vrijeme potrebno za pripremu
    - Sliku gotovog jela
    - Vrstu kuhinje
    - Listu sastojaka
    - Instrukcije za pripremu, podijeljene u jasno označene korake za lakše praćenje
    - Prikaz liste sličnih recepata u bazi

F3. Upravljanje zalihama namirnica

- Dodavanje novih namirnica
    - Korisnik može unijeti nove sastojke u digitalnu smočnicu uz naznačenu količinu dostupnog sastojka
- Ažuriranje postojećih namirnica
    - Korisnik može ažurirati dostupnu količinu zaliha
- Uklanjanje namirnica

        - Korisnik može obrisati potrošene ili neupotrebljive namirnice
    - Pregled svih trenutačno dostupnih namirnica
        - Prikaz zaliha kroz pregledni popis s naznakom dostupnih količina

F4. Upravljanje označenim (omiljenim) receptima

- Označavanje recepta kao favorit
    - Korisnik može označiti recept kao omiljeni za brži pristup
- Uklanjanje recepta iz favorita
    - Korisnik može ukloniti recept iz liste favorita
- Pregled svih favorita
    - Prikaz svih prethodno označenih recepata s mogućnosti pretraživanja po nazivu

F5. Pregled preporučenih recepata

- Prikaz svih preporučenih recepata generiranih na temelju prethodno opisanog modela sustava za preporuke

3.2 Ograničenja

- Ovisnost o vanjskim uslugama
    - Aplikacija koristi uslugu *Spoonacular API* za dohvat recepata, koja ima ograničenje broja dnevnih zahtjeva prema usluzi ovisno o tipu pretplate i načinu plaćanja. Vrijeme odziva prilikom dohvaćanja podataka ovisi o performansama vanjske usluge i nije moguće dodatno optimirati unutar aplikacije. U aplikaciji su dostupni samo oni recepti koji su trenutačno prisutni u bazi podataka Spoonacular *API*-ja. Nadalje, nije moguće garantirati stabilnost strukture podataka jer promjene za vanjskom *API*-ju (npr. promjena modela recepata) mogu uzrokovati probleme u radu aplikacije dok se kod ne prilagodi novoj verziji.
- Ovisnost o internetskoj vezi
    - Za korištenje svih funkcionalnosti aplikacije nužna je dostupnost stalne internetske veze, budući da aplikacija vrši sve pozive prema vanjskim uslugama *online*.
- Platformska ograničenja

- Aplikacija je razvijena isključivo za operativni sustav Android te nije podržana na iOS-u ili drugim platformama.

4. Arhitektura sustava

Aplikacija *PantryMeals* temelji se na arhitekturi klijent-poslužitelj gdje mobilna aplikacija predstavlja klijenta koji putem HTTPS poziva komunicira s vanjskim uslugama i poslužiteljskom infrastrukturom.

Glavne komponente arhitekture sustava su:

- Klijent (mobilna aplikacija za Android)
- Poslužiteljska strana
- Baza podataka

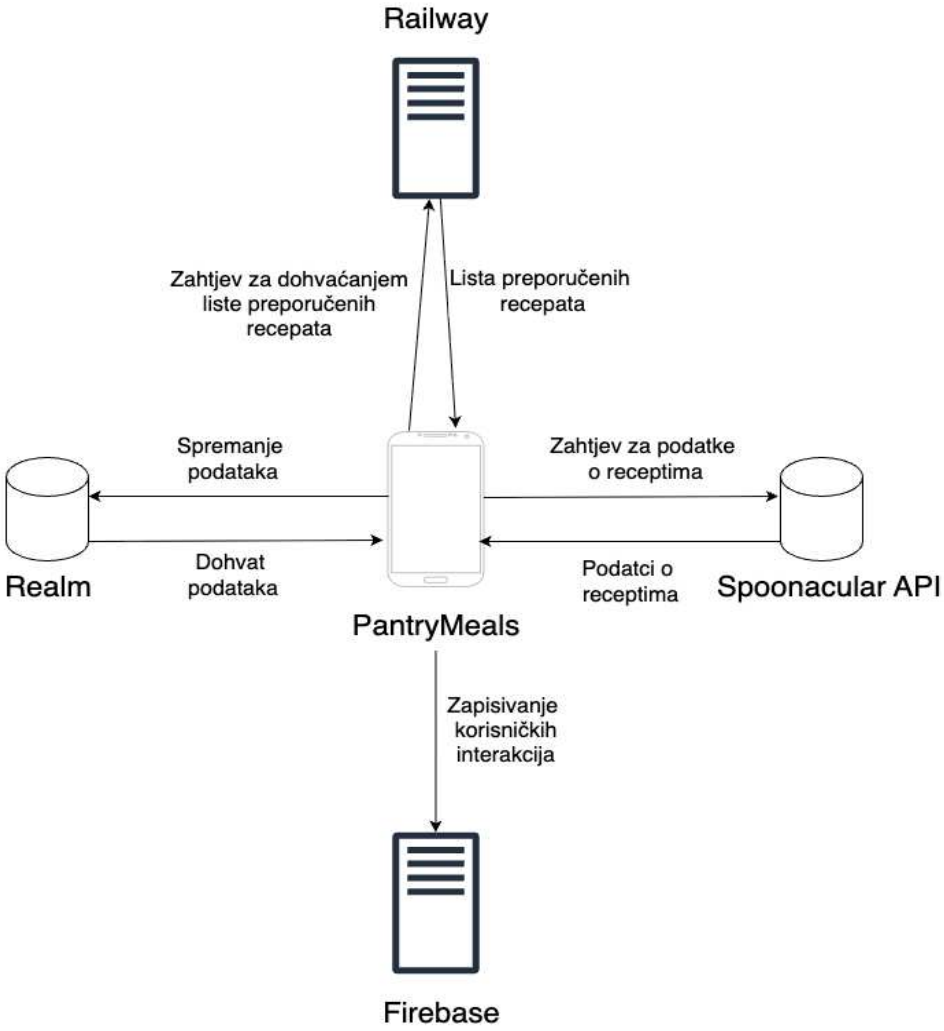
Aplikacija je razvijena koristeći alate i tehnologije prikazane na tablici 4.1.

4.1 Tablica korištenih alata i tehnologija

Komponenta	Tehnologija
Platforma za razvoj mobilne aplikacije	Android Studio
Programski jezik	Kotlin
Tehnologije poslužiteljske strane	REST API
Izvor podataka	Spoonacular Food API [13]
Razvojni alat za sustav preporuke	PyCharm
Lokalna baza podataka	Realm [14]
Zapis korisničkih interakcija	Firebase

4.1 Prikaz arhitekture sustava

Na slici 4.2 prikazana je cjelokupna arhitektura mobilne aplikacije, uključujući sve glavne komponente i njihove međusobne veze.



Slika 4.2 Skica arhitekture sustava

4.2 Klijentska strana – mobilna aplikacija

Klijentska aplikacija razvijena je u programskom jeziku *Kotlin* koristeći alat *Android Studio*.

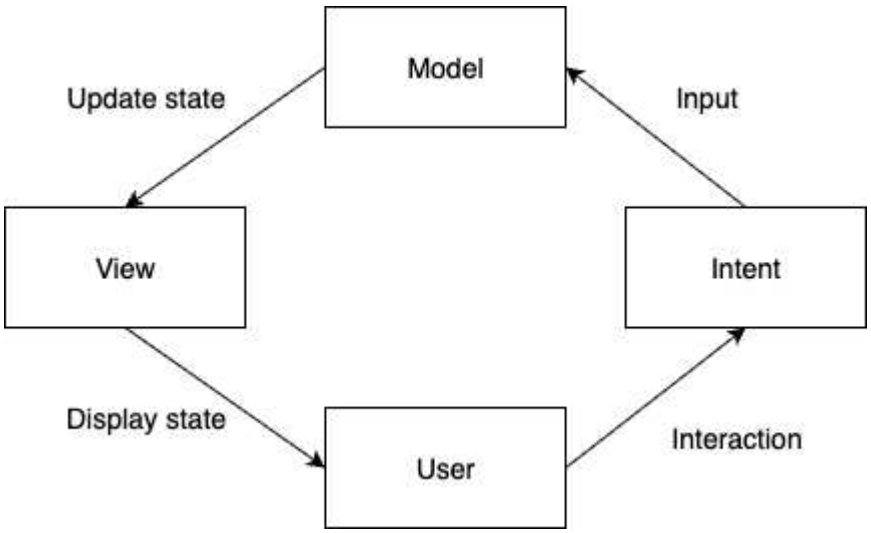
Glavne zadaće klijentske strane su:

- Spremanje i dohvaćanje lokalnih podataka s pomoću baze *Realm*
- Evidencija korisničkih interakcija sa sadržajem putem platforme *Firebase*
- Prikaz i upravljanje intuitivnim korisničkim sučeljem

- Komunikacija s uslugom *Spoonacular Food API* putem poziva *REST API* za dohvaćanje recepata i srodnih podataka
- Komunikacija s uslugom *Railway* za dohvat personaliziranih preporuka recepata temeljenih na korisničkim preferencijama

4.2.1 Pregled arhitekture klijentske strane

Mobilna aplikacija implementirana je korištenjem arhitekturnog obrasca MVI (engl. *Model-View-Intent)* [15] prikazanog na slici 4.3. Navedena arhitektura sadrži tri funkcionalne cjeline što omogućuje jasnu podjelu odgovornosti i jednostavan tok podataka unutar aplikacije. Protok podataka se odvija samo u jednom smjeru između cjelina.



Slika 4.3 Skica arhitekture MVI

Opis arhitekture:

- ***Model***
  - Sadrži sve podatke i poslovnu logiku aplikacije. Njegova zadaća je upravljati izmjenom stanja aplikacije te reagirati na promjene inicirane korisničkim interakcijama
- ***View***
  - Prikazuje trenutačno stanje aplikacije korisniku. Potpuno je odvojen od poslovne logike i ne mijenja podatke izravno već se ažurira kao odgovor na promjene stanja putem modela.
- ***Intent***

- Predstavlja poveznicu između korisničkih akcija i promjena stanja u modelu. Svaka korisnička interakcija generira *intent* koji se šalje modelu na obradu.

Tok podataka:

1. Korisnik izvršava radnju (npr. klik na gumb)
2. Radnja se šalje prema modelu s pomoću *intenta*
3. Model obrađuje zahtjev i sukladno ažurira stanja
4. *View* primjećuje promjenu stanja i ažurira korisničko sučelje
5. Ciklus se ponavlja pri svakoj novoj korisničkoj interakciji

4.3 Poslužiteljska strana

Poslužiteljsku stranu čine vanjski *Spoonacular API*, sustav Railway za *deployment* vlastitog sustava za preporuke te platforma *Firebase* za zapis korisničkih interakcija.
Poslužiteljska strana je odgovorna za:

- Obradu REST zahtjeva aplikacije
- Dohvat i obradu podataka o receptima (preko *Spoonacular* API-ja)
- Generiranje personaliziranih preporuka (sustav za preporuke koji se nalazi na platformi Railway)
- Praćenje korisničkih interakcija preko *Firebasea*

4.4 Baza podataka

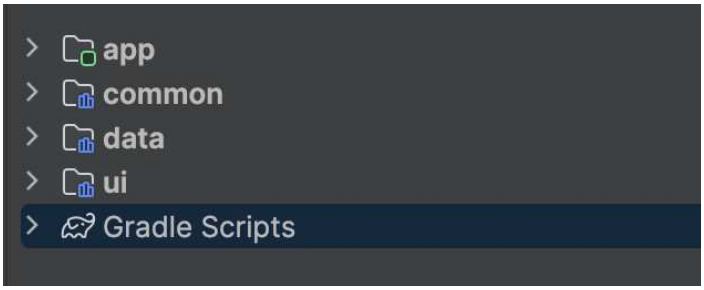
Sustav koristi višeslojni pristup pohrani podataka:

- *Spoonacular* API: putem HTTPS-a za dohvat recepata, sastojaka i ostalih podataka (vanjska baza)
- *Realm*: lokalno spremanje ključnih podataka (zalihe namirnica, omiljeni recepti)
- *Firebase*: pohrana podataka u oblaku za zapis korisničkih interakcija sa sadržajem aplikacije

5. Mobilna aplikacija

Aplikacija *PantryMeals* podijeljena je u četiri modula, kao što je prikazano na slici 5.1:

- *App*
- *Common*
- *Data*
- *Ui*



Slika 5.1 Moduli aplikacije

Moduli u aplikaciji za Android predstavljaju osnovne jedinice za organizaciju koda, resursa i konfiguracijskih datoteka unutar projekta. Svaki modul se može nezavisno graditi, ispitivati i uklanjati pogreške te sadrži specifične dijelove aplikacije poput poslovne logike, korisničkog sučelja i pristupa podacima. Razvoj aplikacije preko modula omogućava jasno razdvajanje odgovornosti, bolju skalabilnost i pojednostavljuje razvoj aplikacije. Svaki modul sadrži i vlastitu skriptu Gradle (*build.gradle*), koja definira specifične postavke za izgradnju poput ovisnosti, pluginova i verzija aplikacije specifičnih za pojedini modul.

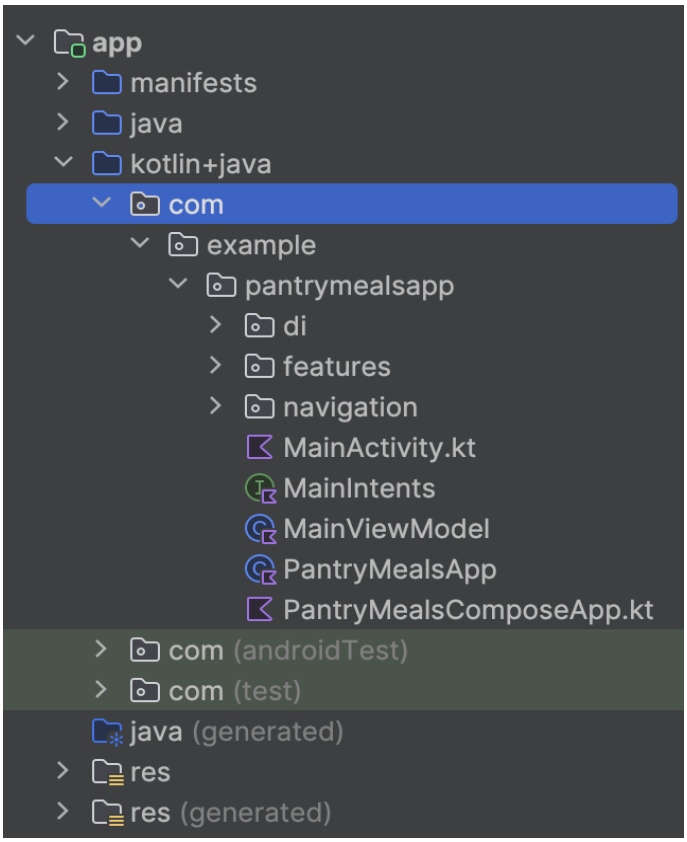
5.1 Aplikacijski modul

Aplikacijski modul predstavlja glavni sloj aplikacije *PantryMeals*. Obuhvaća osnovnu strukturu koda i ključnu logiku za pokretanje, upravljanje i navigaciju aplikacijom.

5.1.1 Pregled strukture

Na slici 5.2 prikazan je pregled strukture aplikacijskog modula u razvojnom okruženju Android Studio. Struktura sadrži sljedeće ključne datoteke i direktorije:

1. **manifests/**
    - Sadrži datoteku AndroidManifest.xml u kojoj su deklarirane osnovne informacije o aplikaciji poput aktivnosti, usluga i dozvola potrebnih za pokretanje i rad aplikacije
2. **res/**
    - U ovom direktoriju nalaze se svi resursi aplikacije, uključujući:
        - drawable/ - direktorij svih ikona i slika korištenih u aplikaciji
        - values/ - XML datoteke koje sadrže *stringove,* boje, stilove i sl.
3. **kotlin + java/**
    - Glavna hijerarhija aplikacijskog koda pisana u Kotlinu. U njoj se nalaze:
        - di/ - direktorij koji se koristi za *dependency injection (DI),* gdje se konfiguriraju ovisnosti koje se koriste unutar aplikacije
        - features/ - direktorij koji sadrži funkcionalne cjeline aplikacije kao što su ekrani i *ViewModeli*
        - navigation/ - komponente za upravljanje navigacijom kroz ekrane
        - Razredi MainActivity, MainIntents, MainViewModel, PantryMealsApp i PantryMealsComposeApp – osnovni ulazni i upravljački razredi za upravljanje životnim ciklusom i inicijalizaciju aplikacije

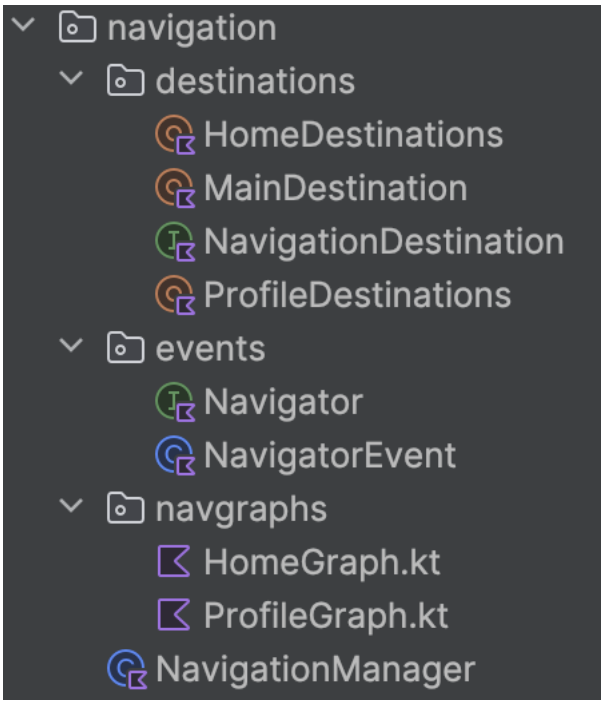


Slika 5.2 Pregled aplikacijskog modula

5.1.2 Navigacija

Navigacija unutar aplikacije je implementirana koristeći Android Jetpack Navigation. Struktura navigacijskog direktorija prikazana je na slici 5.3 i uključuje:

- Direktorije: destinations/, event/, navgraphs/ koji definiraju pojedinačne destinacije, navigacijske događaje i grafove
- Datoteku NavigationManager koja centralizira upravljanje navigacijom



Slika 5.3 Pregled navigacijskog direktorija

Razred *NavigationManager* implementira sučelje *Navigation* i koristi *Kotlin Coroutines* za emitiranje događaja navigacije, dok razred *NavigationEvents* definira tok navigacijskih događaja koji se primaju i šalju kroz sustav upravljanja navigacijom.

Metode za navigaciju

Ključne metode za navigaciju su prikazane na slici 5.4, *navigate* i goBack.

```
sealed class NavigatorEvent {  12 Usages  3 Inheritors
    object GoBack : NavigatorEvent()
    class Directions(val destination: String, val builder: NavOptionsBuilder.() -> Unit) :  2 Usages
        NavigatorEvent()

    class PopBackTo(  2 Usages
        val destination: String,
        val saveState: Boolean,
        val builder: NavOptionsBuilder.() -> Unit
    ) :
        NavigatorEvent()
}
```

Slika 5.4 Slika *NavigatorEvents* razreda

Metoda ***navigate*** koristi se za prelazak na određenu rutu (ekran) unutar aplikacije. Ruta može sadržavati argumente poput *recipeID* koji su potrebni za navigaciju na određene ekrane. Primjer korištenja je vidljiv na slici 5.5, gdje se poziva metoda *navigate*() s pomoću

komponente *NavigationManager* kako bi se otvorio ekran detalja recepta za recept s ID-jem navedenim u ruti.

```
navigationManager.navigate(
    route = HomeDestinations.recipeDetails( recipeId = intent.recipeId).route
)
```

Slika 5.5 Primjer navigacije unutar aplikacije

Metoda ***goBack*** vraća korisnika na prethodni ekran u navigacijskom stogu. Primjer poziva prikazan je na slici 5.6.

```
navigationManager.goBack()
```

Slika 5.6 Primjer korištenja funkcije *goBack*()

Navigacijske rute

Rute se unutar aplikacije definiraju na dva načina koja su vidljiva na slici 5.7:

- Kao jednostavne varijable (npr. ruta "home") koje predstavljaju statične ekrane
- Kao funkcije koje sadrže parametre (npr. ruta "recipeDetails") što omogućuje dinamičko prosljeđivanje argumenta potrebnog za navigaciju

```
object HomeDestinations {  16 Usages
    val home = NavigationDestination { "home" }

    fun recipeDetails(recipeId: Int? = null) =  4 Usages
        object : NavigationDestination {
            override fun destination(): String = "recipeDetails/{recipeId}"
            override val route: String = "recipeDetails/$recipeId"
        }
}
```

Slika 5.7 Sadržaj *HomeDestinations.kt* objekta

Navigacijski grafovi

Rute se mogu organizirati u navigacijske grafove koji grupiraju rute u jedinstvenu cjelinu s početnim ekranom i putanjama između ekrana. Primjer grafa je *profileGraph* vidljiv na slici 5.8 koji definira sve ekrane na kojima možemo navigirati s ekrana profila.

```
fun NavGraphBuilder.profileGraph() {  2 Usages
    navigation(
        route = "profile_graph",
        startDestination = ProfileDestinations.profile(),
    ) {
        composable( route = ProfileDestinations.profile()) { ProfileScreen() }
        composable( route = ProfileDestinations.bookmarks()) { BookmarksScreen() }
        composable( route = ProfileDestinations.recommendedRecipes()) { RecommendedRecipesScreen() }
        composable( route = ProfileDestinations.digitalPantry()) { DigitalPantryScreen() }
    }
}
```

Slika 5.8 Prikaz navigacijskog grafa *profileGraph*

5.2 Modul *common*

Modul *common* sadrži skup pomoćnih funkcija i razreda koje nisu vezane uz određeni dio aplikacije radi povećanja ponovne iskoristivosti i smanjenja dupliciranja logike.

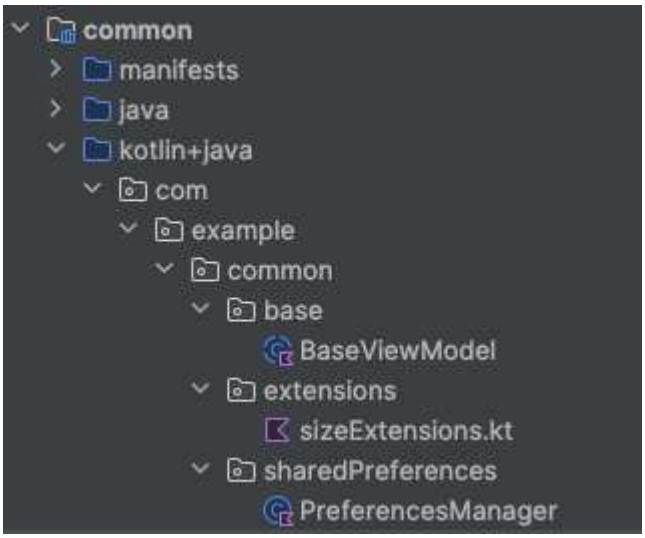
5.2.1 Pregled strukture

Na slici 5.9 prikazana je struktura modula zajedničkih funkcija. Modul sadrži sljedeće datoteke:

1. **BaseViewModel**
    - Apstraktni generički razred koji predstavlja temeljnu komponentu *ViewModel* potrebnu za implementaciju arhitekture MVI
    - Definira uobičajeni način upravljanja promjenama stanja aplikacije i korisničkim akcijama
    - Sadrži komponente:
        - *InitialState* - početno stanje varijabli
        - *InternalState* - spremište korišteno za praćenje i ažuriranje trenutačnog stanja varijabli

- *ViewState* - varijabla koja se koristi na ekranima za ažuriranje sučelja pri promjeni varijabli
- *OnIntent* - apstraktna metoda za obradu korisničkih akcija

2. **SizeExtensions.kt**
    - Datoteka sadrži ekstenzijsku funkciju *Int.dpFromPx(context: Context)* koja pretvara vrijednosti izraženu u pikselima (*px)* u *density-independent pixels (dp)* čime se postiže konzistentniji izgled aplikacije na uređajima različitih gustoća i veličina
3. **PreferencesManager**
    - Pomoćni razred za trajno spremanje i dohvaćanje podataka pomoću mehanizma *SharedPreferences*
    - Podatci spremljeni u razred ostaju trajnu zapamćeni i nakon zatvaranja aplikacije i ponovnog pokretanja
    - Razred je označena s oznakom *Singleton* kako bi se osiguralo da unutar aplikacije postoji samo jedan primjerak razreda
    - U aplikaciji *PantryMeals* koristi se za spremanje i dohvaćanje visine statusne trake



Slika 5.9 Pregled modula zajedničkih funkcija

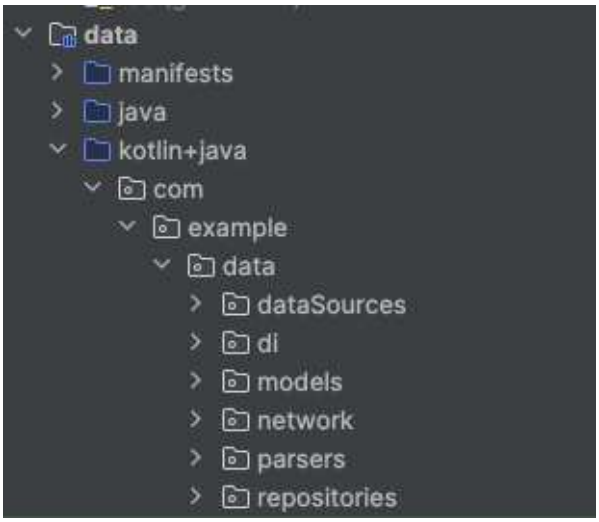
5.3 Modul podatkovnog sloja

Modul podatkovnog sloja (modul *data*) predstavlja ključnu komponentu aplikacijske arhitekture. Zadužen je za pohranu, dohvat i izmjenu podataka. Upravlja lokalnom bazom podataka te komunikacijom s vanjskim uslugama putem mrežnih poziva.

5.3.1 Pregled strukture

Na slici 5.10 prikazana je struktura modula podatkovnog sloja. Modul se sastoji od sljedećih direktorija i datoteka:

1. ***DataSources*/**
    - Sadrži sučelja i razrede za pristup podacima
    - Razdvojen je u dva poddirektorija za pristup podacima s lokalnih i udaljenih izvora
2. **Di/**
    - Sadrži definicije funkcija za injekciju ovisnosti koje omogućuju ostatku aplikacije pristup komponentama poput repozitorija, HTTP klijenta te lokalnih baza podataka
3. **Models/**
    - Definira strukture podataka unutar aplikacije
    - Razdvojen je u dva poddirektorija za definiciju modela za pohranu u lokalnoj bazi podataka i modela potrebnih za interakciju s vanjskim uslugama
4. **Network/**
    - Sadrži *interceptor* zadužen za dodavanje parametara pri slanju zahtjeva prema vanjskim uslugama
5. **Parsers/**
    - Sadrži funkcije potrebne za parsiranje podataka dohvaćenih iz udaljenih izvora u formate podataka koji su pogodni za spremanje u lokalnoj bazi
6. **Repositories/**
    - Predstavlja razrede koji skrivaju detalje implementacije dohvaćanja i manipulacije podataka od ostatka aplikacije



Slika 5.10 Pregled modula upravljanja podacima

5.3.2 Dohvaćanje podataka s vanjskih usluga

U direktoriju namijenjenom definiranju funkcija za injekciju ovisnosti implementiran je klijent HTTP koji koristi *interceptor* za prilagođavanje mrežnih zahtjeva prilikom dohvaćanja podataka s vanjskih usluga.

Interceptor ima ključnu ulogu u osiguravanja standardizacije svih mrežnih zahtjeva. Omogućuje dodavanje osnovnog URL-a prema kojem se šalju svih zahtjevi te automatsko dodavanje API ključa kao parametra.

Sama komunikacija s vanjskim uslugama definirana je u sučelju *BaseAPI* koji deklarira metode za dohvat podataka s pomoću poziva HTTP GET.

Primjer dohvaćanja podataka prikazan je na slici 5.11 gdje se implementira dohvat prijedloga recepata na temelju unosa teksta:

- Na osnovi URL dodaje se tekst *endpointa* "*recipes/autocomplete*"
- Definiraju se dva parametra:
    - Query – tekst unesen u traku za pretraživanje
    - Number – broj rezultata koji se želi dohvatiti
- Rezultat funkcije je lista mogućih recepata koja se korisniku može prikazati kao prijedlog

```
@GET( value = "recipes/autocomplete")  1 Usage
suspend fun getAutoCompleteRecipes(
    @Query( value = "query") query: String,
    @Query( value = "number") number: Int,
): List<AutoCompleteRecipeResponse>?
```

Slika 5.11 Primjer dohvaćanja podataka s vanjske usluge

Poziv funkcije odvija se u odgovarajućem repozitoriju, koji je putem funkcije *provideBaseRepository* dostupan ostatku aplikacije.

Podatkovni modeli za komunikaciju s vanjskim uslugama su definirani kao serijalizabilni razredi podataka kao što je prikazano na slici 5.12, što omogućuje jednostavnu deserijalizaciju mrežnih odgovora u objekte.

```
@Serializable
data class AutoCompleteIngredientResponse(
    @SerialName( value = "name")
    val title: String?,
    @SerialName( value = "image")
    val image: String?
)
```

Slika 5.12 Primjer podatkovnih modela za komunikaciju s vanjskim uslugama

5.3.3 Upravljanje podacima u lokalnoj bazi podataka

Operacije upravljanja podataka u lokalnoj bazi podataka odvijaju se u razredu *BaseLocalSource.*

Za ažuriranje i spremanje podataka koristi se funkcija *realm.write*(), dok se za dohvat podataka koristi funkcija *realm.query*(). Primjeri navedenih funkcija nalaze se na slici 5.13 gdje se u funkciji *bookmarkRecipe*() sprema recept označen kao favorit u jednoj funkciji te se u funkciji *getRecipe*() dohvaća recept s pomoću njegovog ID-ja.

```
suspend fun bookmarkRecipe(recipe: RecipeCardDB) {  1 Usage
    realm.write {
        copyToRealm( instance = recipe, updatePolicy = UpdatePolicy.ALL)
    }
}

private fun getRecipe(recipeId: Int): RecipeCardDB? {  1 Usage
    return realm.query( clazz = RecipeCardDB::class,  query = "id == '$recipeId'").find().firstOrNull()
}
```

Slika 5.13 Primjer upravljanja podacima u lokalnoj bazi

Pozivi funkcija za upravljanje lokalnim podacima su implementirani u odgovarajućim repozitorijima na isti način kao i za dohvat podataka s vanjskih usluga.

Modeli podataka koji se spremaju u lokalnu bazu definirani su kao objekti baze Realm. Sadrže primarne ključeve, definicije varijabli te konstruktore potrebne za korištenje u parserima. Parseri su funkcije koje pretvaraju objekte vanjskih usluga u lokalne objekte baze Realm. Primjer modela je prikazan na slici 5.14.

```
class IngredientDB(): RealmObject {  13 Usages
    @PrimaryKey
    var id: Int? = null
    var name: String? = null
    var count: Int = 0  5 Usages

    constructor(
        id: Int?,
        name: String?,
        count: Int
    ) : this() {
        this.id = id
        this.name = name
        this.count = count
    }
}
```

Slika 5.14 Primjer strukture podataka za spremanje u lokalnoj bazi podataka

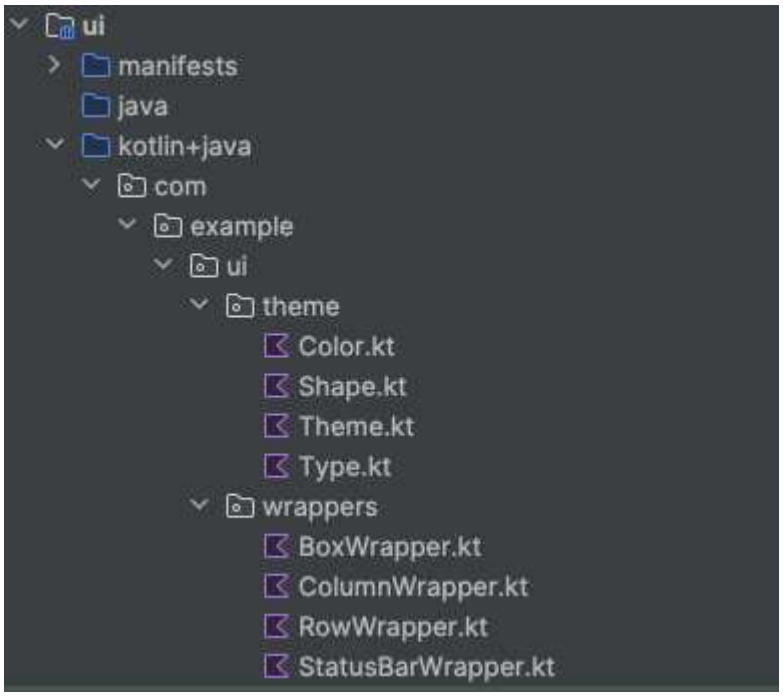
5.4 Modul korisničkog sučelja

Modul korisničkog sučelja (modul *UI*) predstavlja skup funkcija, komponenta i tematskih definicija koje se koriste za prikaz vizualnih elemenata aplikacije, a koje nisu izravno vezane uz pojedinačne funkcionalne cjeline. Cilj modula je osigurati konzistentan vizualni identitet i olakšanje ponovnog korištenja određenog elemenata korisničkog sučelja čime se pridonosi standardizaciji dizajna.

5.4.1 Pregled strukture

Na slici 5.15 prikazana je struktura modula korisničkog sučelja. Modul sadrži dva glavna direktorija:

7. ***Theme*/**
    - Direktorij *theme* sadrži temeljne definicije vizualnog identiteta aplikacije poput boja, oblika, tema i tipografije.
    - *Color.kt* - Definira boje kao varijable s pripadajućim heksadekadskim vrijednostima čime omogućava centralizirano upravljanje svim bojama koje se koriste u aplikaciji
    - *Shape.kt* - Sadrži definicije zakrivljenosti objekata
    - *Theme.kt* - Definira svijetlu i tamnu paletu boja
    - *Type.kt* - Sadrži definicije tipografije, poput fontova, njihovih veličina, težina i stilova
8. ***Wrappers*/**
    - Direktorij sadrži pomoćne komponente namijenjene bržem i konzistentnijem oblikovanju osnovnih elemenata korisničkog sučelja poput horizontalnog i vertikalnog poravnavanja, visina i sl.



Slika 5.15 Pregled modula korisničkog sučelja

5.5 Značajke

Sve značajke aplikacije definirane su unutar aplikacijskog modula zaduženog za prezentacijski sloj sustava.

Svaka značajka sadrži vlastite:

- Definicije ekrana
- Pripadajuće komponente korisničkih sučelja
- Razred *ViewModel* za upravljanje logikom i promjenama stanja
- Objekt *ViewState* za definiranje trenutačnih stanja
- Sučelje *Intent* koje predstavlja korisničke akcije i događaje

Ovaj pristup implementira arhitekturu MVI pri čemu se svaka funkcionalnost aplikacije razvija potpuno odvojeno.

5.5.1 Struktura značajki

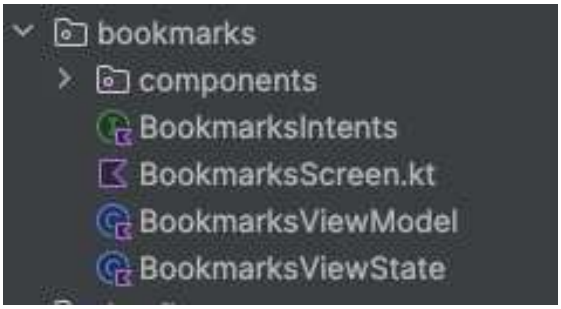
U direktoriju *features* nalaze se sljedeće značajke:

1. Splash – inicijalni zaslon aplikacije koji se prikazuje pri svakom pokretanju
2. Bookmarks – prikaz označenih omiljenih recepata
3. Details – prikaz detaljnih informacija odabranog recepta
4. DigitalPantry – zaslon za upravljanje digitalnom smočnicom korisnika
5. Home – početni zaslon aplikacije
6. Profile – zaslon s korisničkim profilom i opcijama navigacije na sukladne ekrane
7. RecommendedRecipes – prikaz personaliziranih preporuka recepata

Svaka značajka sadrži vlastite komponente korisničkog sučelja te logiku upravljanja stanjima.

5.5.2 Primjer - značajka *Bookmarks*

Kao primjer implementacije značajke opisan je modul *Bookmarks* čija je arhitektura prikazana na slici 5.16.



Slika 5.16 Pregled značajke za označavanje omiljenih recepata

1. Komponente korisničkog sučelja

Korisniku je dostupan zaslon na kojem se prikazuje lista omiljenih recepta. Unutar zaslona nalaze se elementi s kojima korisnik može komunicirat, poput:

- Klika strelice za povrat na prijašnji ekran
- Unosa teksta u traku za pretraživanje
- Dohvaćanja recepata s pomoću zadanog teksta

- Opcije označavanja recepta kao omiljenog
- Opcije micanja recepta iz liste omiljenih
- Odlazak na detalje recepta

2. Korisničke akcije

Svaka interakcija korisnika sa sučeljem pretvara se u *intent.* Svaki *intent* je definiran kao objekti ili razred koji sadrže argumente poput prikaza na slici 5.17.

```
sealed interface BookmarksIntents {  19 Usages  6 Implementations
    object GoBack: BookmarksIntents
    data class BookmarkRecipe(val recipe: RecipeCardDB): BookmarksIntents
    data class UnBookmarkRecipe(val recipeId: Int): BookmarksIntents
    data class GoToDetails(val recipeId: Int): BookmarksIntents  2 Usages
    data class OnSearchBarValueChange(val text: String): BookmarksIntents  2 Usages
    data class GetRecipesFromSearchBar(val text: String): BookmarksIntents  2 Usages
}
```

Slika 5.17 Slika sučelja *BookmarksIntents*

3. *ViewModel*

ViewModel razred definiraju funkciju *onIntent*() kao što je prikazano na slici 5.18. Pomoću navedene funkcije obrađuju se pristigli zahtjevi i povezuju s odgovarajućom poslovnom logikom. Ovisno o zahtjevu, može se dogoditi:

- Dohvat podatak iz repozitorija
- Izmjena stanja spremljenih podataka
- Navigacija na određeni ekran
- Itd.

```
override fun onIntent(intent: BookmarksIntents) {
    when (intent) {
        BookmarksIntents.GoBack -> {
            navigationManager.goBack()
        }

        is BookmarksIntents.BookmarkRecipe -> {
            bookmarkRecipe( recipe = intent.recipe)
        }
```

Slika 5.18 Primjer implementacije funkcije *onIntent*()

4. *ViewState*

Promjenama stanja u funkcijama u razredu *ViewModel*, ažuriraju se pripadajući podatci u razredu *ViewState* koju korisnički sloj promatra. Komponente zaslona se zatim ponovno iscrtavaju s ažuriranim podacima.

5.6 Implementacija sustava za preporuke

Pristup sustavu za preporuke recepata omogućen je preko vanjske usluge Railway, koja izlaže model putem REST API-ja. Zahtjev prema sustavu zahtijeva slanje dvije liste recepata, liste recepata koje je korisnik označio te liste kandidata za preporuke.

Lista kandidata za preporuke generira se pozivom metode GET na *endpoint "recipes/findByIngredients*" usluge Spoonacular API, kao što je prikazano na slici 5.19. U funkciji se definiraju sljedeći parametri:

- *Ingredients* -> trenutačno dostupni sastojci korisnika
- *Number* = 20 -> broj recepata koji će API vratiti kao prijedloge
- *Ranking* = 2 -> opcija kojom se minimizira broj nedostajućih sastojaka u predloženim receptima
- *IgnorePantry = true* -> opcija za ignoriranje osnovnih začina poput soli i papra

Slika 5.19 Implementacija funkcije *getRecipesByIngredients*

Nakon dohvata recepata kandidata, oni se s označenim receptima šalju metodom HTTP POST na *endpoint* "/*predict*" *web* usluge na platformi *Railway*.

U odgovoru se vraća lista preporučenih recepata, sortirana po vjerojatnosti da će se korisniku svidjeti, na temelju predikcija naučenog modela.

6. Korisničke upute

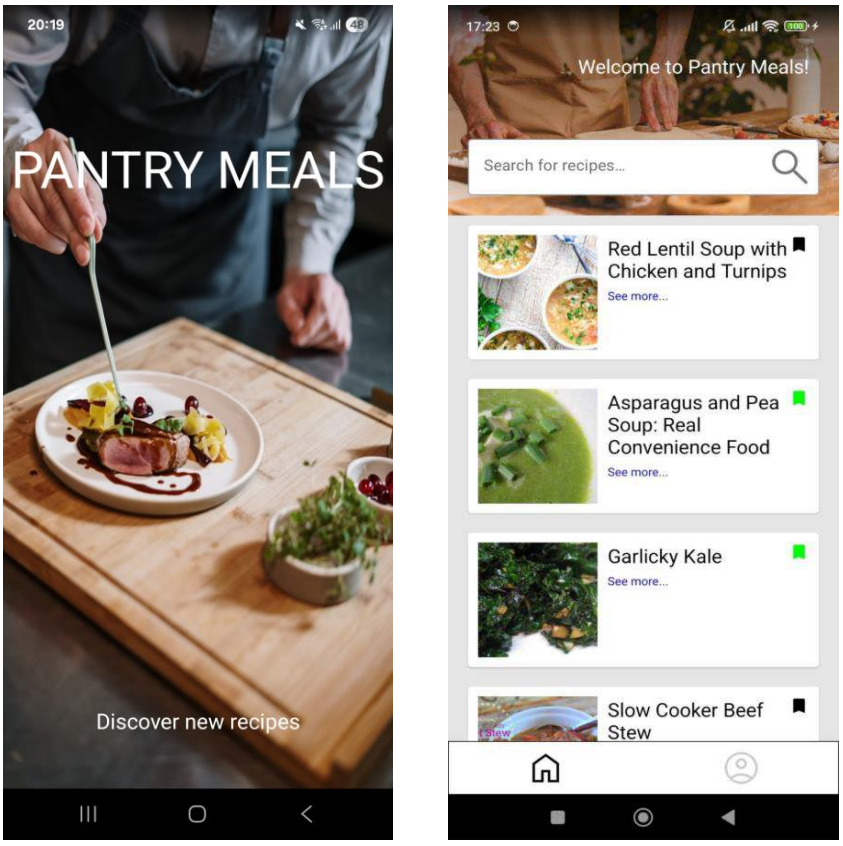
Ovo poglavlje sadrži pregled dostupnih ekrana u aplikaciji te upute korištenja mobilne aplikacije PantryMeals, uključujući opise svih dostupnih ekrana.

6.1 Pregled dostupnih ekrana

6.1.1 Početni ekran

Prilikom pokretanja aplikacije, korisnik najprije vidi ekran *Splash* (slika 6.1) koji se prikazuje dvije sekunde. Nakon toga aplikacija automatski navigira na početni ekran, prikazan na slici 6.2. Početni ekran sadrži sljedeće komponente:

- Traka za pretraživanje
- Lista recepata
- Navigacijska traka



Slike 6.1 (lijevo) - Prikaz ekrana *Splash*

Slika 6.2 (desno) - Prikaz početnog ekrana

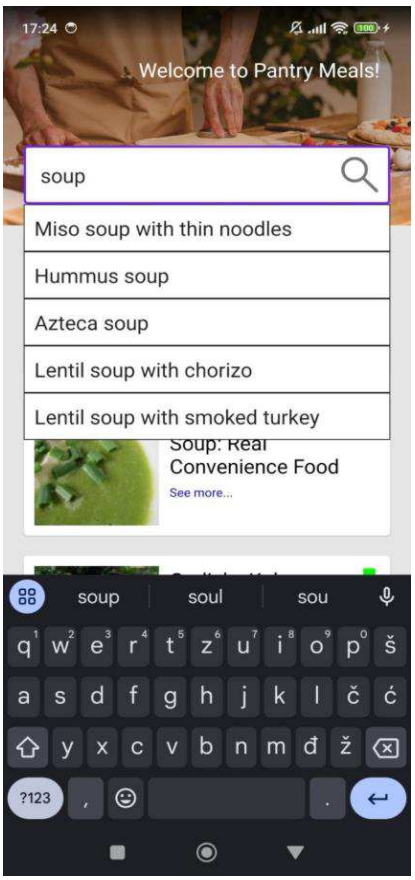
Navigacijska traka sadrži dva gumba:

- Gumb za odlazak na početni ekran
- Gumb za odlazak na ekran profila

Na listi je prikazano 20 slučajno odabranih recepata. Ostale recepte moguće je pretraživati putem trake za pretraživanje. Svaka kartica recepta sadrži naslov, sliku, opciju za pregled detalja i mogućnost označavanja recepta kao omiljenog.

Prilikom unosa teksta u traku za pretraživanje, pojavljuje se lista prijedloga recepta kao na slici 6.3. Prijedlozi se ažuriraju svakih pola sekunde nakon posljednjeg unosa. Klikom na jedan od prijedloga ažurira se prikazana lista recepata.

Klik na tekst "*See more*" na kartici otvara ekran s detaljima recepta.



Slike 6.3 - Prikaz korištenja trake za pretraživanje na početnom ekranu

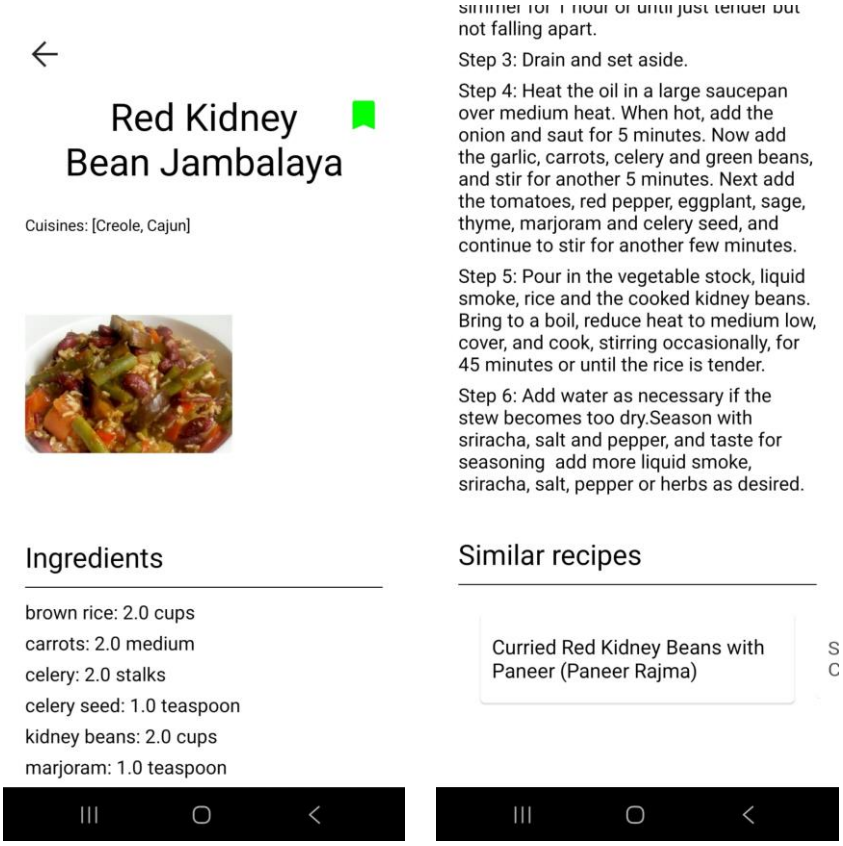
6.1.2 Ekran detalja recepta

Ekran detalja recepta (slike 6.4 i 6.5) prikazuje informacije o:

- Naslovu recepta
- Vremenu kuhanja
- Vrsti kuhinje
- Potrebnim sastojcima
- Koracima pripreme
- Slici jela
- Popisu sličnih recepata

Kao i na karticama recepata, dostupna je mogućnost oznake recepta (*bookmark*) za brži pristup kasnije. Popis sličnih recepata se nalazi na dnu ekrana.

Klik na strelicu za povratak vraća korisnika na prijašnji ekran.

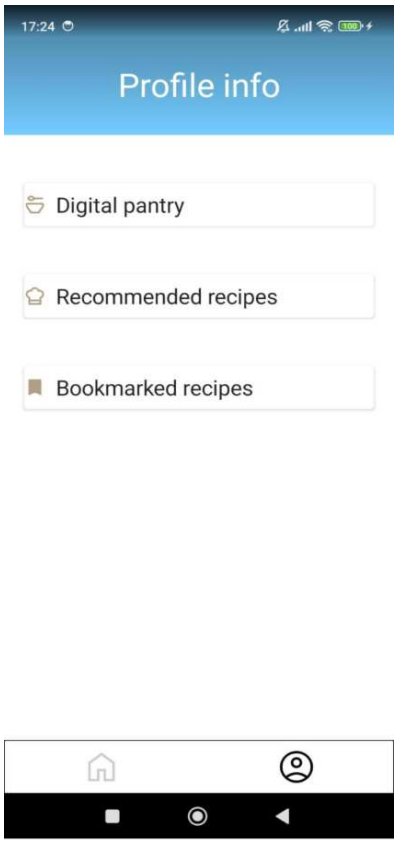


Slike 6.4 i 6.5 - Prikaz ekrana s detaljima recepta

6.1.3 Ekran profila

Ekran profila vidljiv je na slici 6.6 te sadrži navigacijsku traku i tri gumba koji vode na odgovarajuće ekrane:

- **Digitalna smočnica** – vodi na ekran s digitalnim popisom namirnica
- **Preporučeni recepti** – vodi na ekran s preporučenim receptima
- **Označeni recepti** – vodi na ekran s označenim receptima



Slike 6.6 - Prikaz ekrana profila

6.1.4 Digitalna smočnica

Na ekranu digitalne smočnice (slika 6.7) korisnik može vidjeti i upravljati svojom zalihom namirnica. Prikazana je traka za pretraživanje sastojaka i lista dostupnih namirnica s naznačenim količinama (samo one s količinom većom od nule).

Prijedlozi u traci za pretraživanje ažuriraju se svakih pola sekunde nakon prestanka unosa. Odabirom jednog od prijedloga, popunjava se tekst u traci za pretraživanje. Nakon klika na jedan od prijedloga, korisnik klikom na zeleni gumb za dodavanje otvara modal vidljiv na slici 6.8.

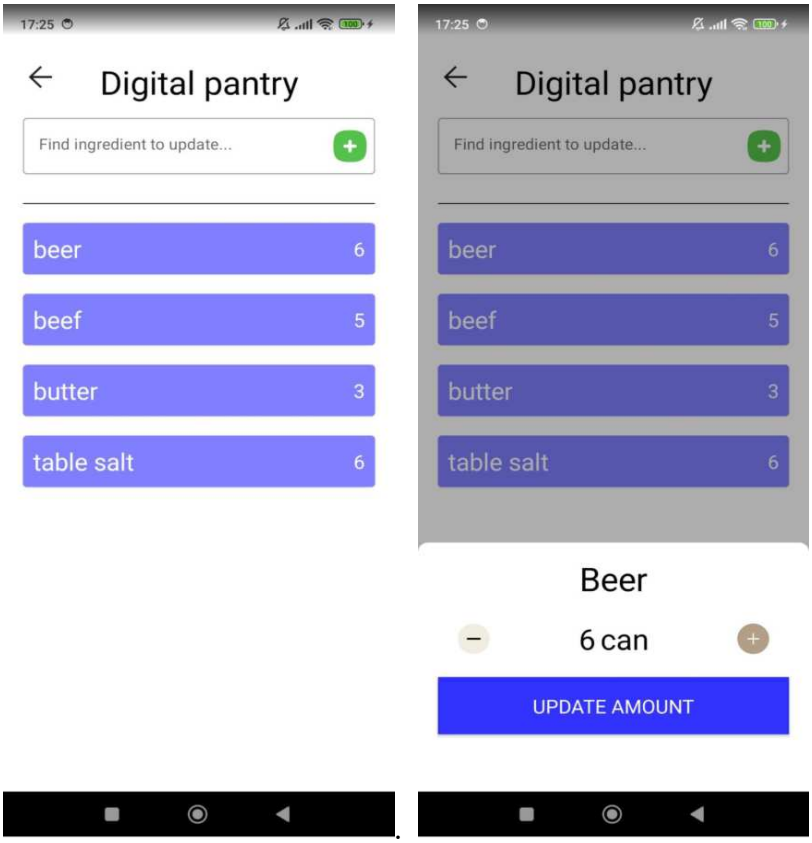
U modalu se prikazuju sljedeći podatci:

- Naziv namirnice
- Naziv jedinice mjere (npr.“limenke” za pive)
- Gumbi za ažuriranje količine

- Gumb za potvrdu unosa

Isti modal se otvara klikom na jedan od postojećih sastojaka u listi, čime korisnik može ažurirati količinu već unesene namirnice.

Klik na strelicu za povratak vraća korisnika na ekran profila.



Slike 6.7 (lijevo) - Ekran digitalne smočnice

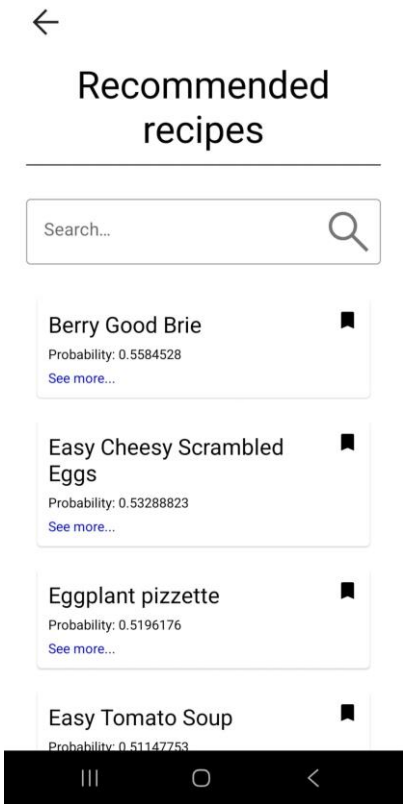
Slika 6.8 (desno) - Prikaz modala za dodavanje i ažuriranje sastojka

6.1.5 Preporučeni recepti

Ekran s preporučenim receptima (Slika 6.9) sadrži traku za pretraživanje i listu preporučenih recepata generiranih prethodno opisanim sustavom za preporuke. Svaka kartica recepta sadrži naslov, mogućnost pregleda detalja, ikonu za označavanje recepta kao omiljenog te vjerojatnost da će korisniku preporučeni recept biti zanimljiv, koju generira sustav preporuka.

Nakon unosa teksta u traku za pretraživanje korisnik treba kliknuti na ikonicu za pretraživanje te će se potom filtrirati i ažurirati lista recepta.

Klik na strelicu za povratak vraća korisnika na ekran profila.



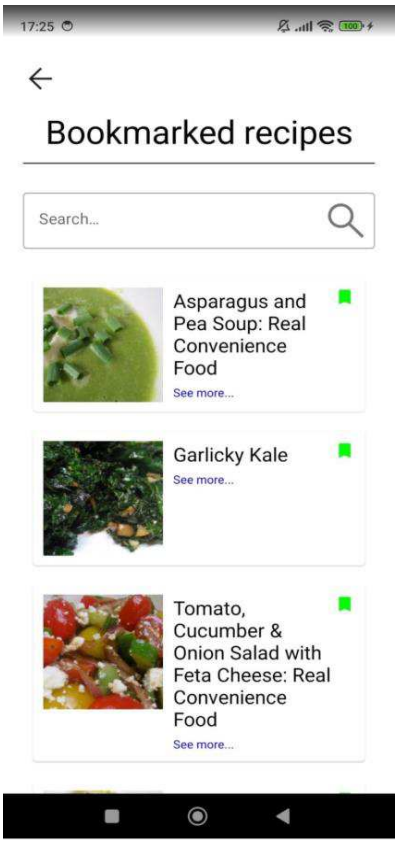
Slike 6.9 - Ekran preporučenih recepata

6.1.6 Označeni recepti

Ekran označenih recepata (Slika 6.10) sadrži traku za pretraživanje i listu označenih recepata. Svaka kartica recepta sadrži naslov, sliku, mogućnost pregleda detalja i ikonu za označavanje recepta kao omiljenog.

Nakon unosa teksta u traku za pretraživanje korisnik treba kliknuti na ikonicu za pretraživanje te će se potom filtrirati i ažurirati lista recepta.

Ponovnim klikom na ikonicu za označavanje recepta, recept se automatski uklanja iz liste.
Klik na strelicu za povratak vraća korisnika na ekran profila.



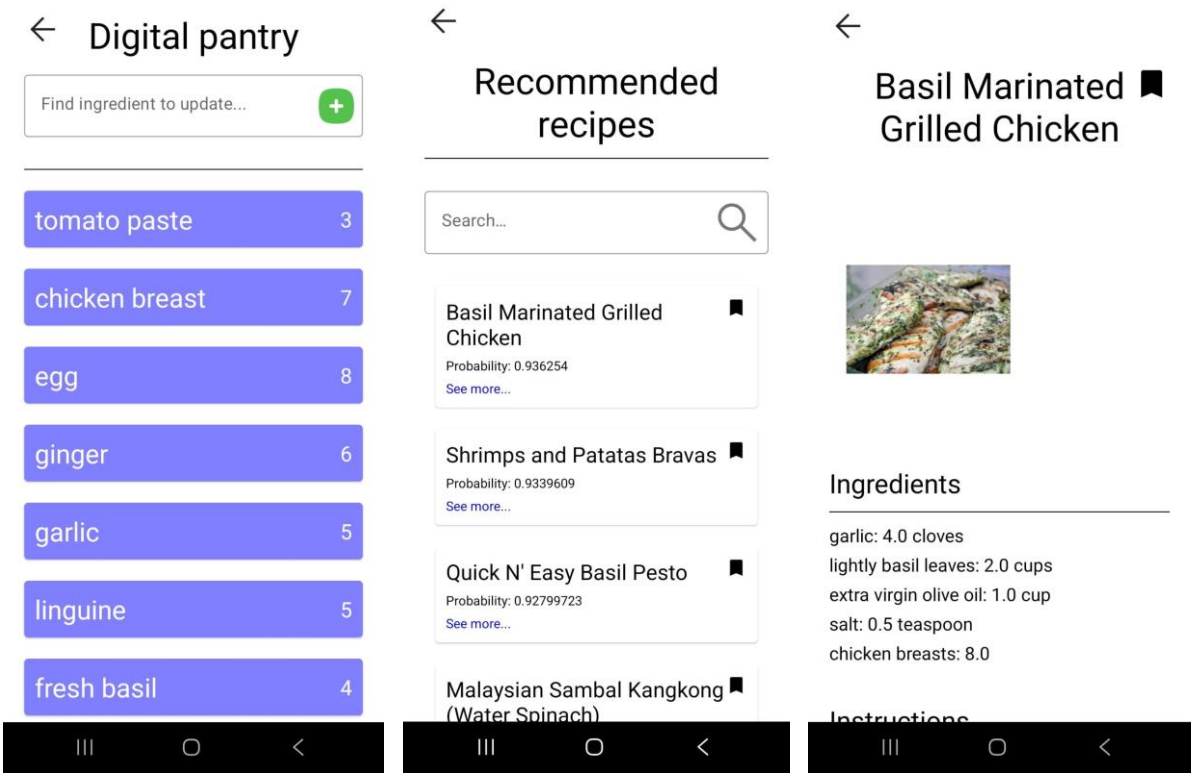
Slike 6.10 - Ekran označenih recepata

6.2 Uobičajeni scenarij korištenja aplikacije

Aplikacija je razvijena za jednostavnu svakodnevnu uporabu, olakšavajući korisnicima planiranje obroka uz optimalno iskorištavanje dostupnih namirnica. Tipičan scenarij korištenja aplikacije izgleda ovako:

- Korisnik nije siguran što bi pripremio za ručak i želi iskoristiti namirnice koje već ima kod kuće. Stoga prvo otvara ekran s digitalnom smočnicom, gdje ažurira popis sastojaka (slika 6.11). Nakon što popuni svoju digitalnu smočnicu, prelazi na ekran s preporučenim receptima gdje se prikazuje rangirana lista jela koja je moguće pripremiti na temelju dostupnih sastojaka s naznačenim vjerojatnostima kao što je prikazano na slici 6.12.
- Korisnik odabire recept s vrha liste, "*Basil Marinated Grilled Chicken",* koji aplikacija predviđa kao najprikladniji izbor.

- Pri otvaranju recepta, korisnik vidi detaljan prikaz s fotografijom, popisom potrebnih sastojaka te potrebnim količinama kao što je prikazano na slici 6.13. Na temelju tog prikaza, korisnik može planirati pripremu obroka bez dodatnih odlazaka u trgovinu.
- Ovakav postupak pojednostavljuje izbor jela, smanjuje bacanje hrane i štedi vrijeme korisniku, čime aplikacija postaje praktičan alat za svakodnevno kuhanje.



Slike 6.11 (lijevo) - Prikaz ekrana digitalne smočnice

Slike 6.12 (sredina) - Prikaz ekrana preporučenih recepata

Slike 6.13 (desno) - Prikaz detalja o izabranom receptu

7. Mogućnosti daljnjeg proširenja i razvoja

Iako je u ovom radu opisana implementirana aplikacija PantryMeals za upravljanje namirnicama s personaliziranim preporukama recepata, postoji niz smjerova za daljnji razvoj i proširenje mogućnosti. Ovo poglavlje navodi neke od mogućih sljedećih koraka za poboljšanje korisničkog iskustva i proširenje funkcionalnosti.

7.1 Multiplatformska podrška

Trenutačno je aplikacija razvijena isključivo za platformu Android. Razvoj podrške za uređaje za platformu iOS doveo bi do proširenja korisničke baze te omogućio pristup široj publici. Osim toga, izrada **web** aplikacije bi omogućila pristup neovisan o operacijskom sustavu te korištenje bez potrebe instalacije, što bi dodatno povećalo fleksibilnost.

7.2 Integracija s platformom Firebase

Aplikacija trenutačno šalje podatke o korisničkim interakcijama s opcijom označavanja recepata na platformu Firebase za analitičke svrhe. No, proširenjem analitičkih podataka, poput klikova na detalje recepata i drugih korisničkih interakcija, moguće je dodatno unaprijediti relevantnost sustava za preporuke.

Nadalje, dodatna integracija s platformom Firebase omogućila bi detaljnije praćenje korisničkog ponašanja unutar aplikacije, što može pomoći u boljem usmjeravanju razvoja i nadogradnji.

7.3 Proširenja funkcionalnosti aplikacije

1. Prepoznavanje namirnica kamerom

Uvođenjem automatskog prepoznavanja namirnica putem fotografija, ubrzao bi se i pojednostavio proces ručnog unošenja i ažuriranja podataka o količinama namirnica.

2. Integracija s cijenama namirnica u dućanima

Prema Zakonu o iznimnim mjerama kontrole cijena donesenom 21. veljače 2025. u Hrvatskoj, trgovci su dužni javno objavljivati cijene proizvoda i ažurirati ih u stvarnom vremenu [16]. Koristeći tu informaciju moguće je napraviti sustav koji će pratiti cijene u svim dućanima i izvještavati korisnike. Uz implementaciju sustava za praćenje cijena, moguće je omogućiti korisnicima i kreiranje lista za kupovinu s aktualnim cijenama te automatski prijenos navedenih artikala u digitalnu smočnicu nakon završetka kupovine.

3. Povećanje broja recepata

Ograničenost količine podataka iz Spoonacular API-ja moguće je zaobići integracijom s više izvora podataka ili razvojem vlastite baze podataka. Također, korisnicima bi trebalo omogućiti kreiranje vlastitih recepata. Pri tome je potrebno razviti mehanizme za rangiranje recepata sa sličnim ili istim nazivima i upravljanje dupliciranim sadržajem.

4. Prijava u korisnički račun

Uvođenjem sustava autentifikacije, korisnici bi dobili mogućnost očuvanja podataka u slučajevima poput mijenjanja mobitela ili brisanja aplikacije gdje bi izgubili podatke o označenim i preporučenim receptima.

5. Uvođenje opcije planiranja obroka

Dodavanje funkcionalnosti planiranja tjednog jelovnika i automatsko generiranje liste potrebnih sastojaka, koje korisnik trenutačno nema u svojoj digitalnoj smočnici, bi značajno povećalo praktičnost i dnevnu upotrebu aplikacije.

7.4 Unaprjeđenje *offline* funkcionalnosti

Omogućavanje korištenje ključnih funkcionalnosti čak i kad korisnik nema pristup internetu, poput pregleda označenih recepata, upravljanja zalihama i pretraživanja preporučenih recepata bi omogućilo neometani rad aplikacije. Sinkronizacija bi se dogodila u trenutku kada se korisnik ponovno spoji s internetom.

7.5 Personalizirana analitika

Dodavanje korisničkih statistika i vizualizacija, poput grafikona potrošnje određenih namirnica, najčešće korištenih jela ili povijesti pripremljenih recepata, donijelo bi korisnicima bolji uvid u vlastite prehrambene navike te potaknulo efikasnije upravljanje zalihama.

7.6 Socijalna funkcionalnost

Uvođenje mogućnosti dijeljenja recepata i lista namirnica između korisnika unutar i izvan aplikacije pomoglo bi u kreiranju zajednice aktivnih korisnika.

7.7 Podrška za lokalizaciju

Aplikacija je trenutačno dostupna samo na engleskom jeziku, a svi jezični sadržaji mogu se ažurirati jedino izdavanjem nove verzije aplikacije. Implementacija usluga za upravljanjem prijevoda poput Lokalisea [17], omogućila bi dinamičko ažuriranje prijevoda te bi takva podrška olakšala uvođenje dodatnih jezika čime bi se povećao doseg aplikacije na tržištu.

Zaključak

U ovom radu razvijena je mobilna aplikacija za platformu Android za upravljanje namirnicama i personaliziranu preporuku recepata na temelju dostupnih namirnica, što doprinosi smanjenju otpada hrane i boljoj organizaciji prehrane.

Opisane su tri osnovne vrste sustava preporuke, a kao najbolji odabran je model filtriranja sadržaja zbog svoje jednostavne implementacije i neovisnosti o velikoj količini korisničkih podataka. Za izgradnju sustava korištena je metoda vektorizacije TD-IDF te model logističke regresije, što je omogućilo precizne i relevantne preporuke prilagođene korisničkim preferencijama.

Aplikacija dohvaća recepte putem vanjske usluge Spoonacular API te korisnicima omogućuje intuitivno upravljanje zalihama namirnica, pregled i uređivanje recepata. Evaluacija modela pokazala je zadovoljavajuću točnost i dobar odziv na korisničke zahtjeve.

Iako su tijekom razvoja uočena određena ograničenja, poput ovisnosti o vanjskim uslugama te fokusiranosti na platformu Android, rad postavlja značajan temelj za daljnja proširenja, uključujući multiplatformsku podršku, proširenje baze podataka i unaprjeđenje analitike korisničkog iskustva.

Sveukupno, uspješna implementacija potvrđuje potencijal sustava preporuka za upravljanje zalihama namirnica i planiranje obroka, s jasnim smjerovima za daljnji razvoj i primjenu.

Literatura

[1] The state of food and agriculture, Food and Agriculture Organization of the United Nations (2019). Poveznica: https://food.ec.europa.eu/document/download/b35701da-c178-4a37-b420-899195e5ba16_en?filename=fw_lib_fao-2019_en.pdf; pristupljeno 30. svibnja 2025.

[2] The 6 Meal-Planning Apps That'll Take All the Brainwork Outta Making Dinner, Melanie Curry (2023). Poveznica: https://www.cosmopolitan.com/lifestyle/g44052513/best-meal-planning-apps/; pristupljeno 30. svibnja 2025.

[3] 5 Best Recipe Apps to Inspire Your Cooking in 2025, Sheharyar (2024). Poveznica: https://softwarehouse.au/blog/5-best-recipe-apps-to-inspire-your-cooking-in-2025/; pristupljeno 30. svibnja 2025.

[4] Paprika, recipe manager. Poveznica: https://www.paprikaapp.com/; pristupljeno 30. svibnja 2025.

[5] Meal planning made easy. Poveznica: https://www.mealime.com/; pristupljeno 30. svibnja 2025.

[6] Whisk your way to simplicity. Poveznica: https://www.whiskapp.net/; pristupljeno 30. svibnja 2025.

[7] Is Yummly the best recipe app ever?, Arielle Nicole Wallace (2021). Poveznica: https://ariellenicole.medium.com/is-yummly-the-best-recipe-app-ever-8f2a7c8ba337; pristupljeno 30. svibnja 2025.

[8] Yummly is Closing: Discover the Best Meal Planning Alternative, Plan to eat (2024). Poveznica: https://www.plantoeat.com/blog/2024/12/yummly-is-closing-discover-the-best-meal-planning-alternative/; pristupljeno 30. svibnja 2025.

[9] What are Recommender Systems?, GeeksForGeeks (2025). Poveznica: https://www.geeksforgeeks.org/machine-learning/what-are-recommender-systems/; pristupljeno 30. svibnja 2025.

[10] What is collaborative filtering?, Jacob Murel, Eda Kavlakoglu (2024). Poveznica: https://www.ibm.com/think/topics/collaborative-filtering; pristupljeno 30. svibnja 2025.

[11] What is content-based filtering?, Jacob Murel, Eda Kavlakoglu (2024). Poveznica: https://www.ibm.com/think/topics/content-based-filtering; pristupljeno 30. svibnja 2025.

[12] 7 Types of Hybrid Recommendation System, Jeffery chiang (2021). Poveznica: https://medium.com/analytics-vidhya/7-types-of-hybrid-recommendation-system-3e4f78266ad8; pristupljeno 30. svibnja 2025.

[13] Spoonaculari api. Poveznica: https://spoonacular.com/food-api; pristupljeno 30. svibnja 2025.

[14] Realm-kotlin. Poveznica: https://github.com/realm/realm-kotlin; pristupljeno 30. svibnja 2025.

[15] MVI Architecture with Android, Rim Gazzah (2020). Poveznica: https://medium.com/swlh/mvi-architecture-with-android-fcde123e3c4a; pristupljeno 30. svibnja 2025.

[16] Zakon o iznimnim mjerama kontrole cijena, Narodne novine (202, veljača). Poveznica: https://narodne-novine.nn.hr/clanci/sluzbeni/2025_03_40_540.html; pristupljeno 30. svibnja 2025

[17] How to build and manage multilingual products with Hygraph and Lokalise, Ilya (2024). Poveznica: https://hygraph.com/blog/manage-multilingual-products-with-hygraph-and-lokalise; pristupljeno 30. svibnja 2025.

Sažetak

Mobilna aplikacija za praćenje zaliha namirnica i planiranje obroka s uključenim sustavom preporučivanja recepata

U današnjem ubrzanom načinu života, upravljanje zalihama namirnica i planiranje obroka postaju sve veći izazov, što često rezultira bacanjem hrane i lošijom prehranom. U ovom radu razvijena je mobilna aplikacija za Android koja omogućuje učinkovito upravljanje zalihama namirnica, planiranje obroka te preporuke recepata na temelju dostupnih sastojaka. Za izradu sustava za preporuke korišten je model filtriranja sadržaja s metodom vektorizacije TD-IDF i logističkom regresijom kao vjerojatnosnim klasifikatorom. Recepti se dohvaćaju putem vanjskog servisa Spoonacular API-ja, a aplikacija omogućuje unos, praćenje i uređivanje namirnica, kao i detaljan pregled recepata. Na kraju rada analiziraju se mogućnosti proširenja funkcionalnosti, uključujući multiplatformsku podršku i proširenje baze podataka.

Ključne riječi: upravljanje namirnicama, preporuka recepata, filtriranje sadržaja, TD-IDF, logistička regresija, mobilna aplikacija, platforma Android

Summary

Mobile application for monitoring food supplies and meal planning with a recipe recommender system

In today's fast-paced lifestyle, managing food inventory and meal planning has become increasingly challenging, often resulting in food waste and poor nutrition. This thesis presents the development of an Android mobile application that enables efficient management of food inventory, meal planning, and personalized recipe recommendations based on available ingredients. The recommendation system employs a content-based filtering model using TD-IDF vectorization and logistic regression as a probabilistic classifier. Recipes are retrieved via the external Spoonacular API, and the application allows users to enter, track, and edit ingredients, as well as view detailed recipe information. The thesis concludes with an analysis of potential feature expansions, including multiplatform support and database growth.

Keywords: food inventory management, recipe recommendation, content-based filtering, TD-IDF, logistic regression, mobile application, Android platform