

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 365

**MOBILNA APLIKACIJA ZA PLANIRANJE OSOBNE
PREHRANE**

Filip Marčec

Zagreb, lipanj 2022.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 365

**MOBILNA APLIKACIJA ZA PLANIRANJE OSOBNE
PREHRANE**

Filip Marčec

Zagreb, lipanj 2022.

Zagreb, 11. ožujka 2022.

ZAVRŠNI ZADATAK br. 365

Pristupnik: **Filip Marčec (0036525149)**
Studij: Elektrotehnika i informacijska tehnologija i Računarstvo
Modul: Računarstvo
Mentor: prof. dr. sc. Krešimir Fertalj

Zadatak: **Mobilna aplikacija za planiranje osobne prehrane**

Opis zadatka:

Proučiti reprezentativne mobilne aplikacije za evidenciju recepata, planiranje prehrane i nabavu namirnica te postaviti zahtjeve na vlastito programsko rješenje. Oblikovati vlastito rješenje nad bazom podataka koje će omogućiti evidenciju vlastitih recepata, planiranje prehrane za korisnički definirano razdoblje te generiranje popisa za kupovinu namirnica na temelju rasporeda prehrane. Ugraditi funkcionalnost generiranja popisa recepata na temelju evidencije namirnica raspoloživih u datom trenutku. Omogućiti generiranje plana prehrane u narednom razdoblju na temelju povijesnih podataka.

Rok za predaju rada: 10. lipnja 2022.

Sadržaj

UVOD.....	1
1. Analiza problemskog područja.....	1
1.2. Recipe Keeper.....	1
1.2. CookMate.....	4
1.3. Whisk: Recipes & Meal Planner.....	6
1.4. Zaključak analize	9
2. Specifikacija zahtjeva.....	9
2.1. Funkcionalni zahtjevi.....	9
2.2. Nefunkcionalni zahtjevi.....	10
3. Arhitektura aplikacije	10
3.1. Model baze podataka	10
3.1.1. Konceptualni model baze podataka.....	10
3.1.2. Fizički model baze podataka	11
3.2. Arhitektura sustava	18
3.2.1. Modeli.....	19
3.2.2. Pogledi.....	20
3.2.3. Kontroleri	26
3.3. Izvoz/ispis podataka u PDF formatu.....	29
3.4. Objavljivanje obavijesti	30
4. Korisničko sučelje aplikacije.....	30
4.1. Autentifikacija – prijava/registracija	30
4.1.1. Početna stranica	30
4.1.2. Registracija	31
4.1.2. Prijava.....	31
4.2. Moji recepti.....	32
4.2.1. Okvir za pretraživanje	33

4.2.2. Dodavanje novog recepta	33
4.2.3. Brisanje postojećeg recepta	33
4.2.4. Pregled pojedinačnog recepta	34
4.2.5. Uređivanje sastojaka	35
4.3. Raspored kuhanja	36
4.3.1. Plan za današnji dan	36
4.3.2. Pregled budućih i prošlih planova	37
4.3.3. Brisanje plana	37
4.3.4. Dodavanje novog plana	38
4.4. Smočnica	38
4.4.1. Okvir za pretraživanje	39
4.4.2. Dodavanje novog zapisa u smočnicu	39
4.4.3. Izmjena podataka za postojeću namirnicu u smočnici	40
4.4.4. Brisanje zapisa iz smočnice	40
4.5. Popis za trgovinu	41
4.5.1. Okvir za pretraživanje	42
4.5.2 Dodavanje novog zapisa na popis za trgovinu	42
4.5.3. Uređivanje zapisa s popisa za trgovinu	43
4.5.4 Označavanje zapisa kao kupljenog i brisanje zapisa s popisa za trgovinu	44
4.5.6. Izvođenje popisa za trgovinu u PDF formatu	45
5. Korištene tehnologije	46
5.1. React Native	46
5.2. EXPO	46
5.3 Firebase	46
5.3.1. Firebase Cloud Firestore	46
5.3.2 Firebase Auth	46
5.4. Visual Studio Code	47

5.5. Astah UML	47
5.6. ERDPlus.....	47
Zaključak	48
Literatura	49
Sažetak.....	50
Abstract.....	51

UVOD

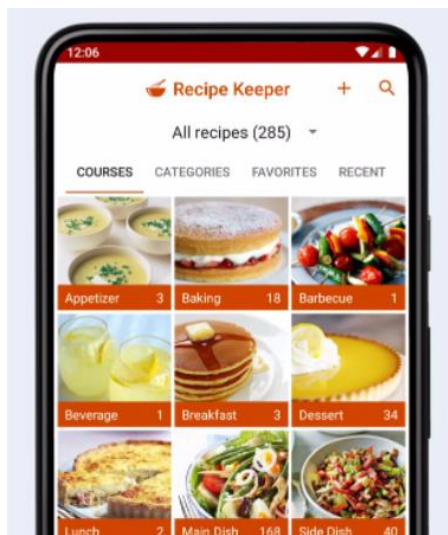
1. Analiza problemskog područja

Prije početka oblikovanja vlastitog programskog rješenja, potrebno je proučiti reprezentativne mobilne aplikacije za evidenciju recepata koje postoje na tržištu. Većina aplikacija besplatna je za korištenje, a neke implementiraju dodatne, napredne funkcionalnosti uz plaćanje mjesečne pretplate. Zanimljivo je da na tržištu dominiraju aplikacije koje nemaju opciju stvaranja vlastitih recepata, nego se baziraju na već postojećim bazama podataka koje sadrže recepte, a korisnik ima mogućnost ih pregledavati i spremati kako bi ih kasnije mogao koristiti.

Već prilikom odabira aplikacija za provedbu analize lako je uočljivo da sadrže slične funkcionalnosti poput dodavanja vlastitih recepata, pretraživanja i spremanja recepata s drugih web stranica te mogućnost stvaranja popisa za trgovinu. U nastavku je provedena analiza nad tri mobilne aplikacije: Recipe Keeper, Cookmate i Whisk: Recipes & Meal Planner. Za svaku od ovih aplikacija bit će navedeni općeniti podaci te pregled funkcionalnosti koje su implementirane.

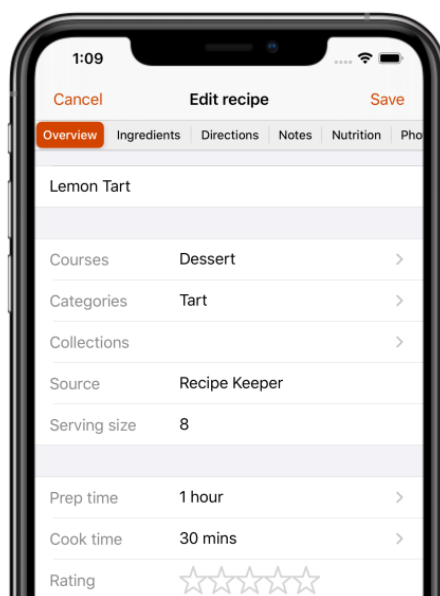
1.2. Recipe Keeper

Recipe Keeper [1] mobilna je aplikacija dostupna na Android, Apple i Windows uređajima, a trenutno na Google Play trgovini bilježi preko 100000 preuzimanja. Aplikacija omogućava pregled svih recepata grupiranih po korisnički definiranim kategorijama, nedavno korišteni recepti ili recepti koji su od strane korisnika označeni kao „omiljeni“.



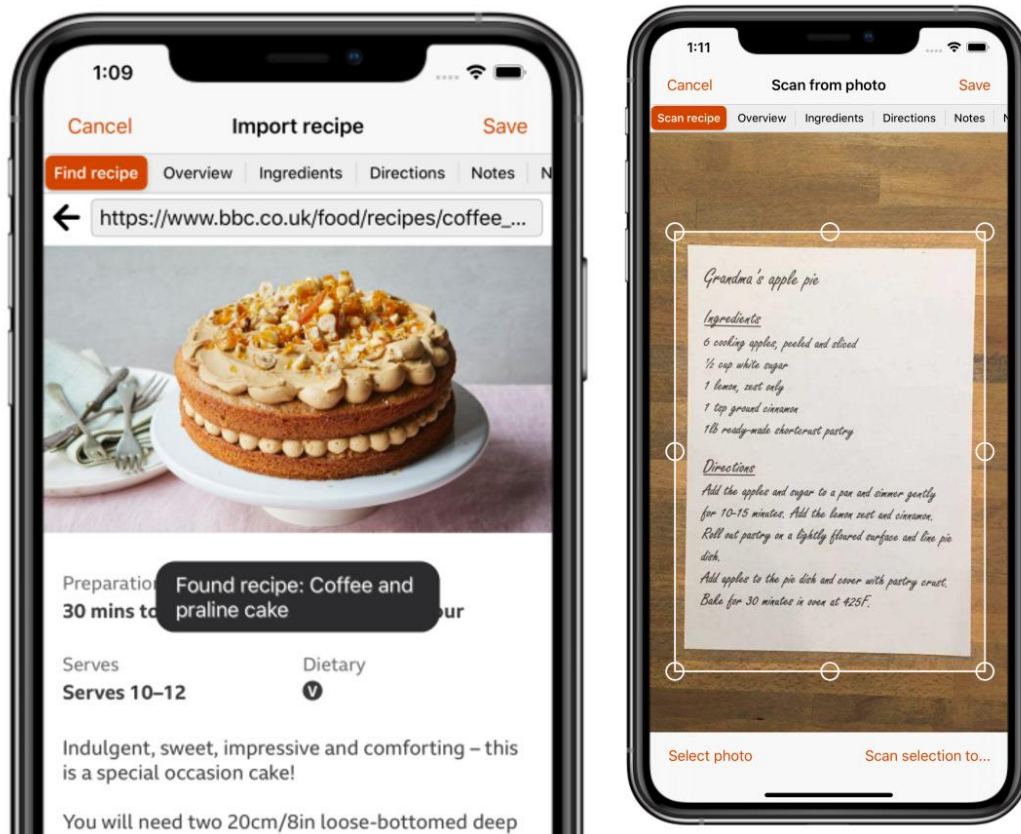
Slika 1. Recipe Keeper - pregled svih recepata

Aplikacija omogućava ručni unos i uređivanje recepata koje je vidljivo na slici 2. Za svaki recept moguće je definirati potrebne sastojke, upute za pripremu, dodatne bilješke, nutricionističke podatke te dodati sliku.



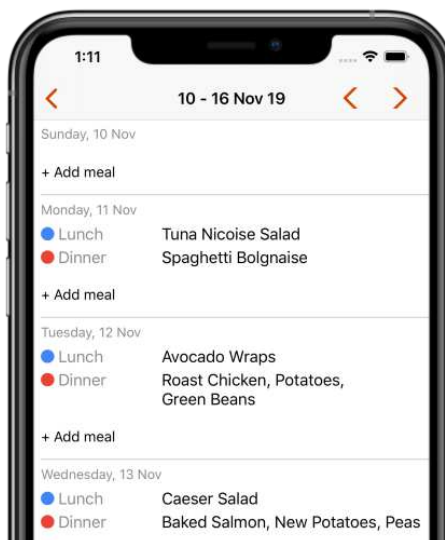
Slika 2. Recipe Keeper - uređivanje recepata

Osim ručnog unošenja recepata, aplikacija ima još dvije mogućnosti dodavanja recepata – skeniranje recepta te uvoz recepta s druge web stranice.



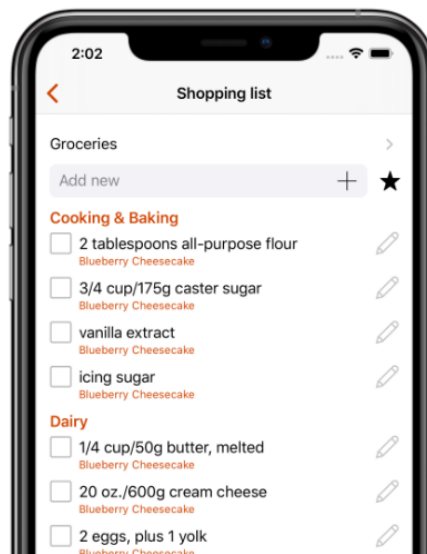
Slika 3. Recipe Keeper - dodatne mogućnosti unosa receptata

Za svaki recept prilikom stvaranje potrebno je unijeti za koji broj osoba je navedena količina dostatna, a prilikom upotrebe tog recepta, moguće je povećati ili smanjiti broj osoba, a aplikacija će automatski prilagoditi količinu potrebnih sastojaka. Nadalje, Recipe Keeper ima implementiranu i mogućnost stvaranja rasporeda obroka za korisnički odabrano razdoblje.



Slika 4. Recipe Keeper - stvaranje rasporeda obroka

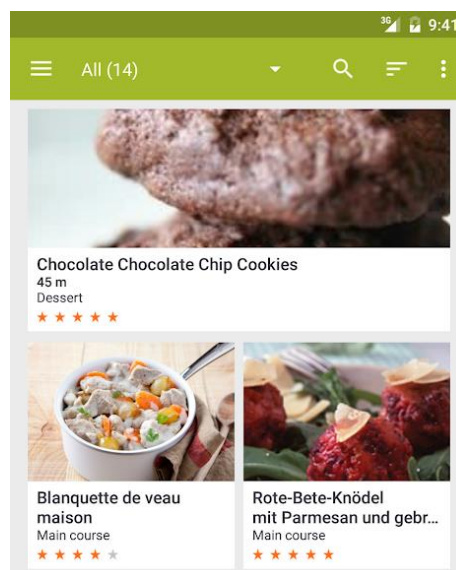
Osim rukovanja receptima, ova aplikacija ima i mogućnost stvaranja popisa za kupovinu na temelju sastojaka potrebnih za pripremu određenog jela.



Slika 5. Recipe Keeper - popis za kupovinu

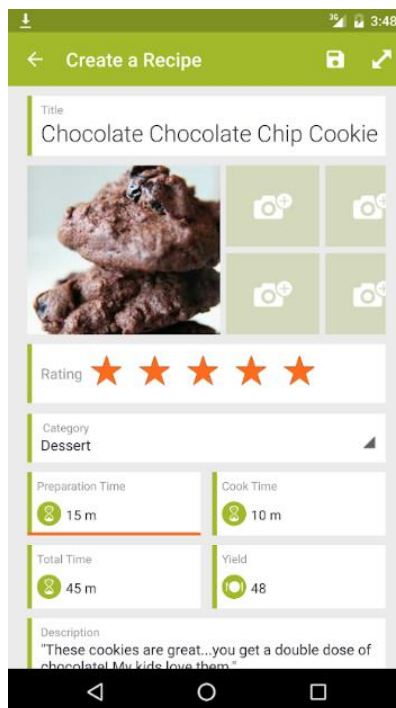
1.2. CookMate

CookMate [3] aplikacija je koja je dostupna za Android i Apple uređaje. Aplikacija dolazi u besplatnoj verziji koja ograničava broj pohranjenih recepata, popisa za trgovinu te mogućnost uvoza recepata, te *premium* verziju uz mjesečnu pretplatu. Aplikacija ima implementiranu funkcionalnost pregleda svih spremljenih recepata kako je vidljivo na slici 6.



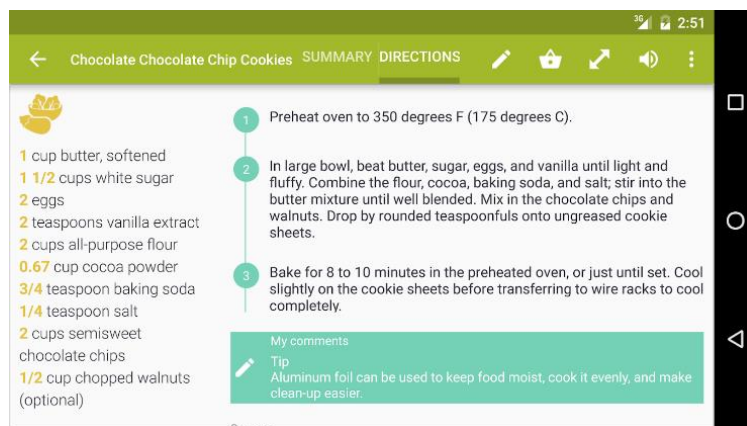
Slika 6. CookMate - prikaz svih recepata

Sučelje za unos novog recepta prikazano je na slici 7. Moguće je osim uobičajenih stavki poput sastojaka i uputa, definirati i vrijeme pripreme, vrijeme kuhanja te težinu pripreme.



Slika 7. CookMate - unos novog recepta

Na slici 8. vidljivo je sučelje na kojem se nalaze sastojci potrebni za pripremu određenog jela i upute za pripremu, kao i polje za dodatne bilješke.



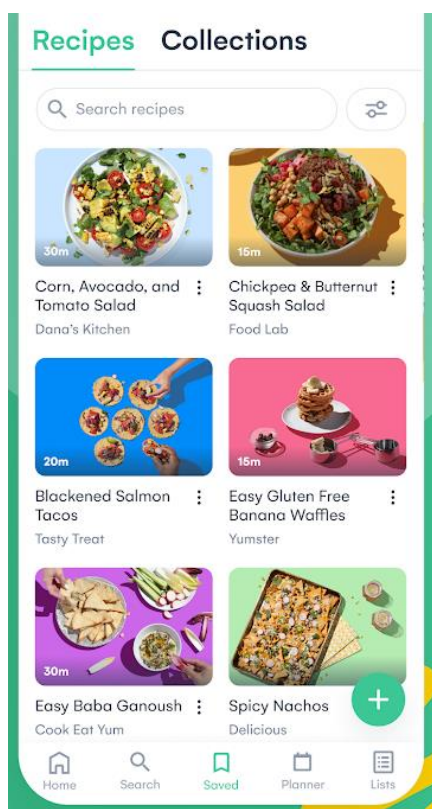
Slika 8. CookMate - sastojci i upute za pripremu

Uz do sad navedene funkcionalnosti, u aplikaciji CookMate implementirano je i stvaranje popisa za trgovinu, prilagodbu količine sastojaka prema broju ljudi za koje se priprema obrok te mogućnost korištenja usluge za „čitanje“ recepta. Aplikacija također veliki naglasak

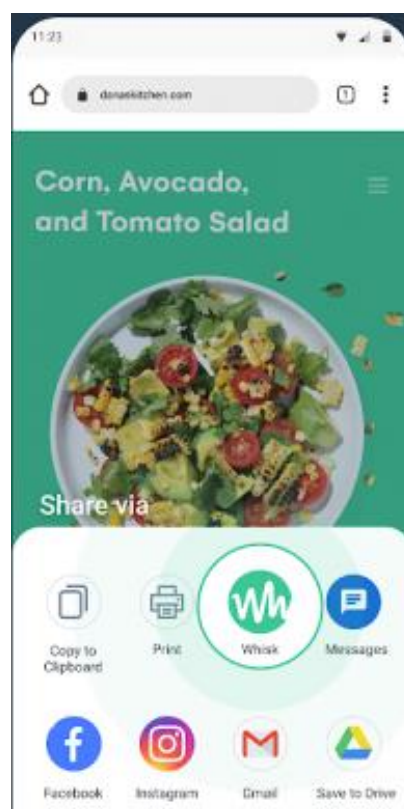
stavlja na mogućnost dijeljenja svoje „kuharice“ sa drugim osobama, kao i mogućnost sinkronizacije svojih recepata ako se aplikacija koristi na više uređaja.

1.3. Whisk: Recipes & Meal Planner

Whisk: Recipes & Meal Planner [2] besplatna je aplikacija koja je dostupna za korištenje na Android i Apple uređajima, a na Google Play trgovini bilježi preko 500 tisuća preuzimanja te je daleko najpopularnija aplikacija od tri navedene u ovoj analizi.



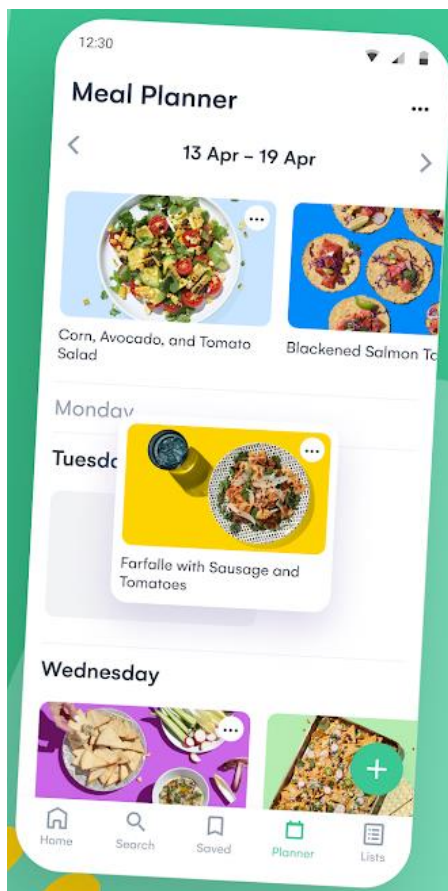
Slika 10. Whisk - prikaz svih recepata



Slika 9. Whisk - uvoz recepata.

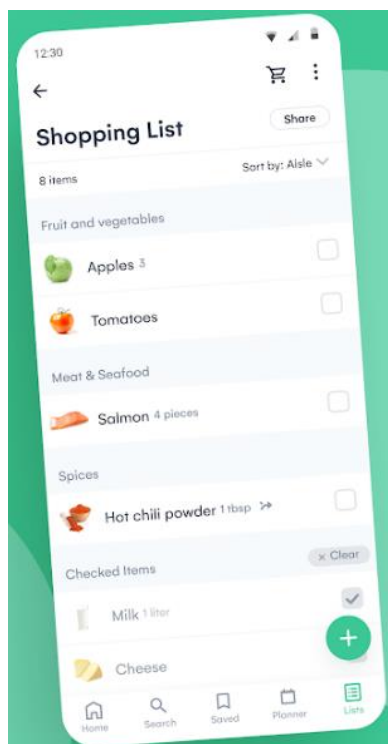
Recepti se mogu pretraživati pomoću polja za pretraživanje, a moguće ih je pregledavati i kroz „kolekcije recepata“ koje su korisnički definirane – slika 9. Osim ručnog unosa recepata, ova aplikacija također omogućava i uvoz recepata s gotovo svih web stranica što je vidljivo na slici 10.

Whisk također omogućava stvaranje tjednog plana prehrane. Sučelje koje to omogućuje prikazano je na slici 11. Za svaki dan moguće je odabrati jedan ili više obroka koji se planiraju pripremiti.



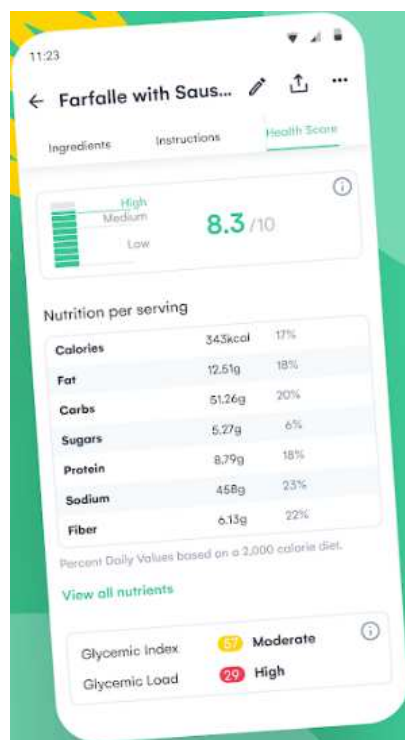
Slika 11. *Whisk* - tjedni jelovnik

Kao i prethodne aplikacije, *Whisk* ima opciju stvaranja popisa za kupovinu od namirnica potrebnih za pripremu odabranih recepata. Popis je grupiran po kategorijama proizvoda.



Slika 12. Whisk - stvaranje popisa za kupovinu

Ono što Whisk ističe od prethodno analiziranih aplikacija jest sučelje koje prikazuje detaljni pregled nutritivnih vrijednosti za pojedini recept što je vidljivo na slici 13.



Slika 13. Whisk - pregled nutritivnih vrijednosti

1.4. Zaključak analize

Nakon analize tri postojeća programska rješenja, jasno je da postoji nekoliko osnovnih funkcionalnosti koje su implementirane u sve tri aplikacije:

- dodavanje, uređivanje i brisanje vlastitih recepata
- pretraživanje recepata i spremanje istih
- planiranje i stvaranje tjednog rasporeda obroka
- stvaranje popisa za trgovinu.

Nadalje, mobilna aplikacija *Whisk* ima najbolje izvedeno rješenje za izgled sučelja i navigaciju, pritom koristeći navigacijsku traku koja se nalazi na dnu ekrana. Isti princip navigacije bit će implementiran u ovom radu. Recepti se mogu grupirati po raznim kategorijama, što će biti slučaj i u aplikaciji koja će biti dana kao programskog rješenje ovog rada. Korištene kategorije bit će kasnije definirane. Sučelje koje prikazuje nutritivne podatke neće biti implementirano jer bi se na taj način otežao i zakomplicirao unos recepata, a glavno načelo programskog rješenja jest preglednost i jednostavnost.

2. Specifikacija zahtjeva

Aplikacija ima jednu vrstu korisnika – registrirani korisnici. Dakle, potrebno je obaviti registraciju/prijavu u aplikaciju kako bi korisnik mogao pristupiti svim funkcionalnostima koje su implementirane.

2.1. Funkcionalni zahtjevi

- registracija/prijava u aplikaciju
- pregled, dodavanje, izmjena i brisanje recepata
- stvaranje plana prehrane za odabrano razdoblje
- generiranje popisa za trgovinu na temelju plana prehrane
- filtriranje recepata prema raspoloživim namirnicama
- pregledavanje, dodavanje, izmjena i brisanje sastojaka
- pregledavanje, dodavanje, izmjena i brisanje namirnica u smočnici
- pregledavanje, dodavanje, izmjena i brisanje stavki s popisa za trgovinu
- mogućnost izvoza popisa za trgovinu u .PDF obliku
- mogućnost izvoza plana prehrane u .PDF obliku.

Svaki korisnik može upravljati svojim Popisom recepata koji se sastoji od jednog ili više Recepta. Svaki Recept sastoji se od više Namirnica(sastojaka). Korisnik ima jednog Popis za trgovinu kojim upravlja te se on sastoji od jednog ili više Namirnica(stavki). Smočnica se također sastoji od jedne ili više Namirnica. Planovi prehrane dijele se na Prošle i Buduće planove, a svaki se sastoji od jednog ili više Plana. Svaki Plan sastoji se od jednog recepta.

3.1.2. Fizički model baze podataka

Pri izradi mobilne aplikacije korištena je baza podataka Firebase Firestore Cloud Database (u nastavku teksta Firestore). Firestore je NoSQL baza podataka. Za razliku od SQL baze podataka, Firestore nema tablice i redove. Umjesto takve strukture, podaci su organizirani u dokumente od kojih svaki pripada određenoj kolekciji. Na slici 15. nalazi se konfiguracijska datoteka u kojoj se nalaze podaci potrebni za spajanje mobilne aplikacije na bazu podataka.

```
js firebase.js > ...
1  import firebase from 'firebase/compat/app';
2  import 'firebase/compat/auth';
3  import 'firebase/compat/firestore';
4
5  const firebaseConfig = {
6    apiKey: "AIzaSyB6S1vh8GZesE1ZC018mgDo7DA1PQmBw4Is",
7    authDomain: "nutplan-5258e.firebaseio.com",
8    projectId: "nutplan-5258e",
9    storageBucket: "nutplan-5258e.appspot.com",
10   messagingSenderId: "673514617062",
11   appId: "1:673514617062:web:ad6860e4a5dcf17bd92327",
12   measurementId: "G-EENGSMLOMY"
13 };
14
15 if(!firebase.apps.length){
16   firebase.initializeApp(firebaseConfig)
17 }
18
19 export {firebase};
```

Slika 15. Konfiguracijska datoteka firebase.js

Svaki dokument sadrži parove podataka u obliku *ključ-vrijednost*, te je takva struktura odlično optimizirana za pohranu velikog broja jednostavnih dokumenata. Osim takvih parova podataka, dokument također može sadržavati i podkolekcije.

Podacima se pristupa putem referenci. Reference su objekti koji pokazuju na određenu poziciju u bazi podataka. U nastavku je dan primjer korištenja reference i čitanja podataka o svim namirnicama koje se trenutno nalaze u smočnici iz baze podataka.

```
const foodsRef = firebase.firestore().collection('pantry')

useEffect(async () => {
  foodsRef.orderBy('name').onSnapshot(
    QuerySnapshot => {
      const foods = []
      QuerySnapshot.forEach((doc) => {
        const {name, amount, unit, lastUpdated} = doc.data()
        foods.push({
          id: doc.id,
          name,
          amount,
          unit,
          lastUpdated,
        })
      })
      setFoods(foods)
      setFilteredData(foods)
    }
  )
}, [1])
```

Slika 16. Primjer uporabe reference i čitanja iz baze podataka

Ovim postupkom su u polje „foods“ pohranjeni podaci iz baze podataka u obliku JavaScript objekata. Na slici 17. prikazan je isječak koda u kojem se dodaje zapis u bazu podataka. Na slici je također vidljiva upotreba „timestampa“. *Timestamp* je tip podataka koji označava vrijeme, a u datom primjeru koristi se kako bi korisnik znao kada je zadnji put ažurirao podatke za određenu namirnicu.

```
const addToPantry = () => {
  setLoading(true)
  foodsRef.add({
    name: foodData.name,
    amount: parseInt(foodData.amount),
    unit: foodData.unit,
    lastUpdated: firebase.firestore.FieldValue.serverTimestamp()
  })
  .then(() => {
    console.log('Item added!');
    setLoading(false)
    Alert.alert(
      "Dodano na stanje"
    )
  });
}
```

Slika 17. Prikaz dodavanja podataka u bazu

Nadalje, na slici 18. prikazan je isječak koda u kojem je demonstrirano ažuriranje već postojećeg podatka u bazi podataka. U isječku je vidljivo korištenje parametra „merge“ koji signaliziramo bazi da želimo samo promijeniti određenu vrijednost, a ne u potpunosti prepisati podatak.

```
const updateFood = async () => {
  foodData.lastUpdated = firebase.firestore.FieldValue.serverTimestamp()
  try{
    const foodRef = await firebase.firestore().collection('pantry').doc(route.params.item.id)
    await foodRef.set(foodData, {merge: true})
    console.log("updejtno")
  }
  catch(error)
  {
    alert(error.message)
  }
}
```

Slika 18. Primjer ažuriranja dokumenta u bazi podataka

Brisanje dokumenata je vrlo jednostavno, osim ako dokument sadrži podkolekcije. Tada je potrebno posebno izbrisati pod kolekciju, a potom izbrisati dokument. Jednostavan primjer brisanja dan je na slici 19.

```
const deleteFood = (food) => {
  foodsRef
    .doc(food.id)
    .delete()
    .then(() => {
      Alert.alert("Obavijest:" , "Uspješno izbrisano sa stanja");
    })
    .catch(error => {
      alert(error);
    })
}
```

Slika 19. Primjer brisanja dokumenta

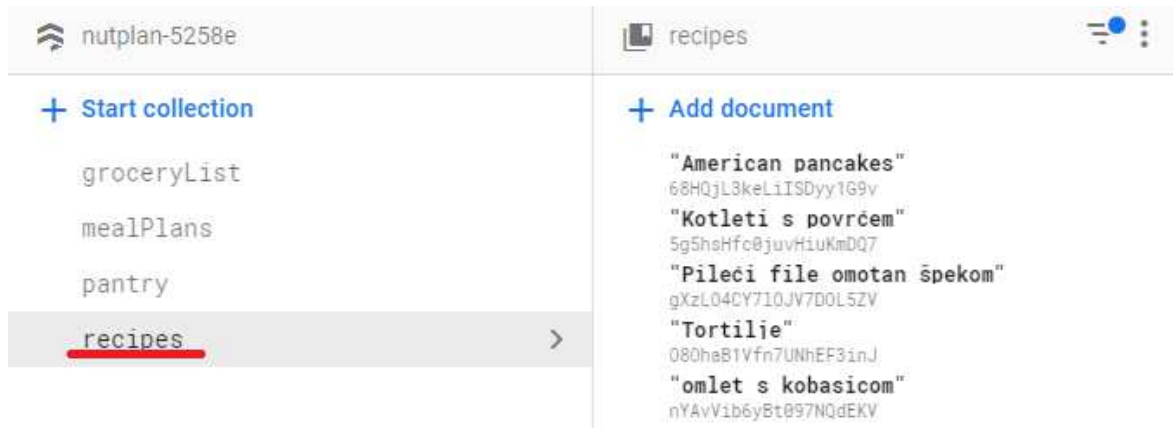
U nastavku teksta opisana je fizička struktura baze podataka uz priložene slike.

User (Korisnik)

Korisnici se u bazi podataka pohranjuju u unaprijed definiranu strukturu te su u Firestore konzoli vidljivi u odjeljku „Authentication“. Pri izradu ove mobilne aplikacije nije stavljan poseban naglasak na korisnika pa se tako za njega spremaju samo: ime, e-mail adresa i lozinka.

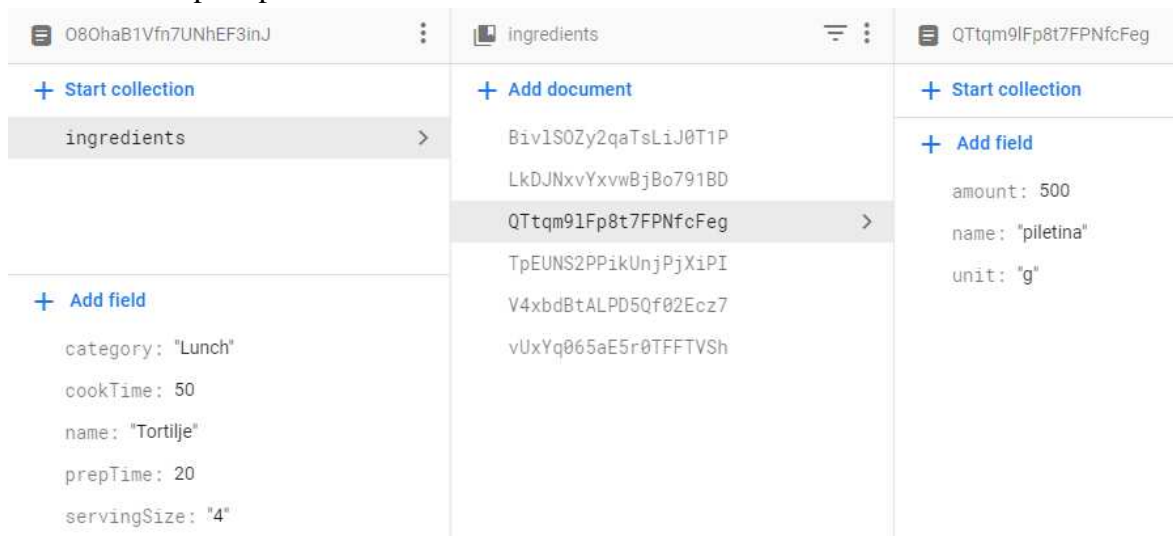
Recipes (Recepti)

Kolekcija „recipes“ sadrži dokumente od kojih svaki predstavlja jedan recept. Svaki dokument ima automatski generiran ID.



Slika 20. Prikaz kolekcije "recipes" u Firestore konzoli

U nastavku je dan primjer jednog dokumenta za recept na kojem će biti vidljiva struktura dokumenta i tipovi podataka.



Slika 21. Struktura dokumenta - recept

Recept opisuju sljedeći atributi:

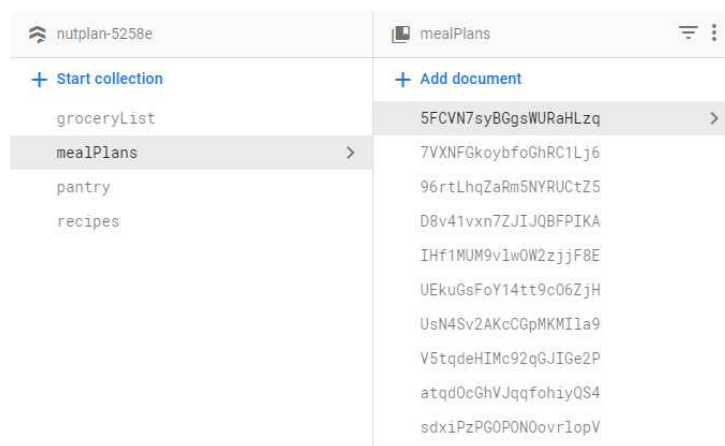
- *name*: string // naziv recepta,
- *category*: string, // kategorija kojoj pripada recept, npr. *Lunch* (ručak),
- *prepTime*: number, // vrijeme potrebno za pripremu sastojaka,
- *cookTime*: number, // vrijeme potrebna za obradu(kuhanje/pečenje) sastojaka,
- *servingSize*: number, // broj obroka/osoba za koje će biti dovoljan recept.

Podkolekcija **ingredients (sastojci)** također je vidljiva na slici 21. Svaki dokument u toj podkolekciji opisuju sljedeći atributi:

- *name*: string, // naziv sastojka,
- *amount*: number, // potrebna količina sastojka,
- *unit*: number, // mjerna jedinica.

MealPlans (Raspored prehrane)

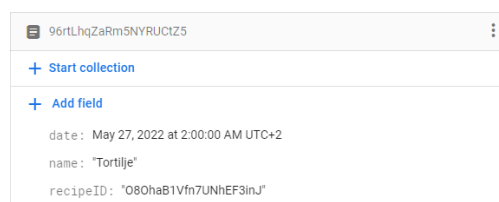
Kolekcija „MealPlans“ sadrži dokumente koji sadrže informacije o receptu i datum pripreme odabranog recepta. Kolekcija je vidljiva na slici 22. Svaki dokument ima jedinstveni ID koji je generiran od strane baze podataka.



Slika 22. Prikaz kolekcije "MealPlans" u bazi podataka

Dokumenti u kolekciji „MealPlans“ imaju sljedeću strukturu:

- *name*: string, // ime recepta koji će se pripremati na odabrani datum,
- *date*: date, // datum na koji se određeni plan odnosi,
- *recipeID*: string, // jedinstveni ID recepta pomoću kojeg se dohvaćaju podaci o odabranom receptu.



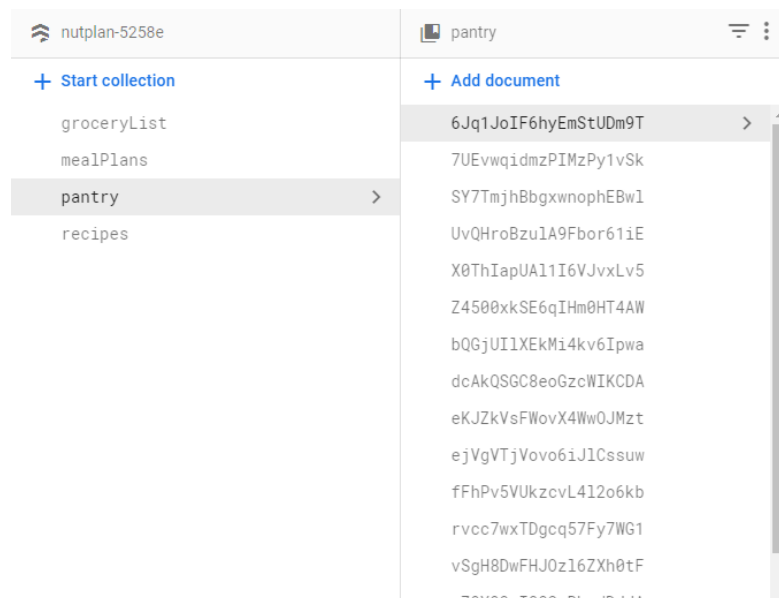
Slika 23. Prikaz jednog dokumenta kolekcije "MealPlans"

Čest problem prilikom razvoj aplikacija koje koriste Firestore bazu podataka jest dupliciranje podataka. Zbog toga umjesto da ovdje budu spremljene sve informacije o

receptu, spremljen je samo naziv i ID, što omogućuje da prilikom prikaza rasporeda prehrane imamo dovoljno informacija za prikaz, a ako je potrebno dohvatiti detalje o receptu, tada koristimo zapisani ID. Na taj način je smanjeno dupliciranje podataka. Ovdje je bitno napomenuti da su na slici 14., na konceptualnom dijagramu baze podataka *budući* i *prošli planovi* prikazani kao dva odvojena entiteta, međutim u fizičkom modelu izvedeni su unutar iste kolekcije, te se prilikom dohvaćanja podataka o planovima ti planovi razdvajaju na *buduće* i *prošle* na temelju atributa *date*.

Pantry (Smočnica)

Kolekcija „Pantry“ sastoji se od dokumenata koji pohranjuju informacije o pojedinim namirnicama koje korisnik trenutno ima kod kuće. Na slici 24. vidljiva je kolekcija i njeni dokumenti. Kao i prije, svaki dokument ima jedinstveni ID koji je generiran od strane baze podataka.



Slika 24. Prikaz kolekcije "Pantry"

Dokument u kolekciji „Pantry“ sadrži sljedeće atribute:

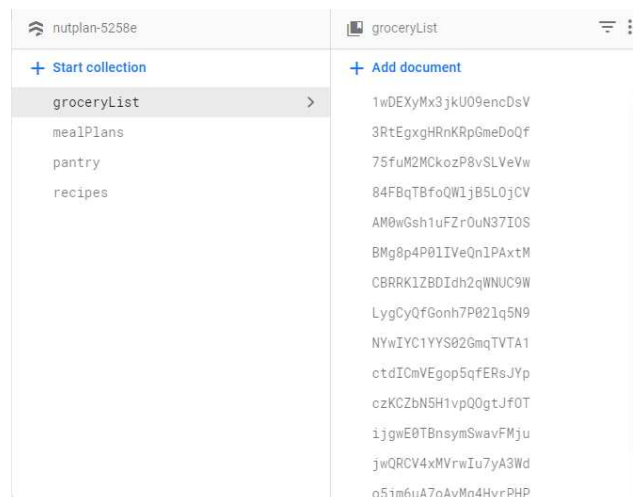
- name: string // naziv namirnice čije su informacije pohranjene u dokumentu,
- amount : number, // količina namirnice koju korisnik trenutno posjeduje
- unit: string, // mjerna jedinica koja zajedno s količinom opisuje stanje namirnice
- lastUpdated: timestamp, // objekt koji predstavlja vrijeme kada su zadnji put ažurirani podaci o toj namirnici.



Slika 25. Prikaz jednog dokumenta u kolekciji "Pantry"

GroceryList (Popis za trgovinu)

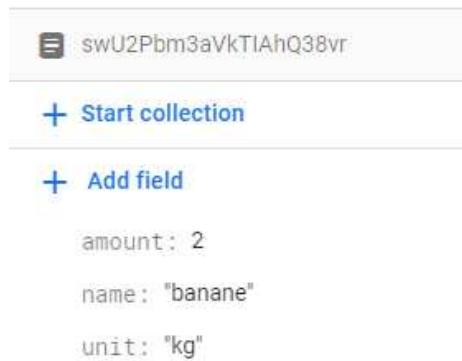
GroceryList kolekcija je koja sadrži dokumente koji modeliraju jednu od stavku na popisu za trgovinu. Svaki dokument ima jedinstven, automatski generiran ID, a kolekcija i njeni dokumenti vidljivi su na slici 26.



Slika 26. Prikaz kolekcije "GroceryList"

Svaki dokument u kolekciji „GroceryList“ sadrži sljedeće atribute:

- name: string, // naziv namirnice o kojoj dokument sadrži informacije
- amount: number, // količina namirnice koju je potrebno kupiti, odnosno minimalna količina
- unit: string, // mjerna jedinica koja zajedno s količinom daje potpunu informaciju o tome koliko je određene namirnice potrebno kupiti.



Slika 27. Primjer jednog dokumenta u kolekciji "GroceryList"

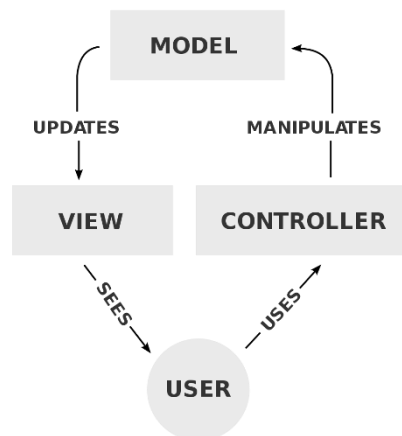
3.2. Arhitektura sustava

Obrazac model – pogled – upravljač (engl. MVC) zapravo dijeli aplikaciju na tri međusobno povezana dijela.

Model: odgovoran za ponašanje i promjenu podataka

View: korisniku prikazuje podatke iz modela

Controller: posrednik između modela i pogleda, odgovoran za obradu korisnikovih zahtjeva i akcija



Slika 28. Shema MCV obrasca [5]

3.2.1. Modeli

Pri razvoju ove mobilne aplikacije nisu eksplicitno definirane klase koje bi predstavljale modele. Prilikom čitanja, odnosno dohvata podataka iz baze, baza podataka vraća podatke u obliku JSON objekata te bi stvaranje dodatnih klasa i njihovih konstruktora bila redundantno.

```
const [foodData, setFoodData] = useState({
  id: "",
  name: "",
  amount: 0,
  unit: "",
});
```

Slika 29. Isječak koda - primjer inicijalizacije objekta prije dodavanja u bazu

Prilikom dodavanja novih podataka ili ažuriranja postojećih podataka u Firestore bazu podataka korišten je „*useState*“ [\[17\]](#) čiji je primjer korištenja dan na slici 29. gdje se definira „varijabla stanja“ koja ima atribute koji opisuju namirnicu koja se nalazi u smočnici. Atributi su inicijalizirani na početne vrijednosti koje u ovom slučaju odgovaraju tipovima podataka definiranim u bazi podataka. „*useState*“ vraća par: trenutno stanje - *foodData* i funkcija kojom se to stanje može mijenjati – *setFoodData*. Ukoliko želimo pristupiti samo jednom atributu, npr. trebamo ispisati naziv namirnice, to radimo na sljedeći način:

```
<Text>foodData.name</Text>
```

U slučaju kada putem obrasca u kojem korisnik unosi podatke koje je potrebno zapisati u definirani objekt, koristimo metodu koja je vidljiva na slici 30. Metodi se predaju dva parametra :

- name – naziv parametra kojem želimo promijeniti vrijednost
- value – nova vrijednost koju je korisnik unio putem obrasca.

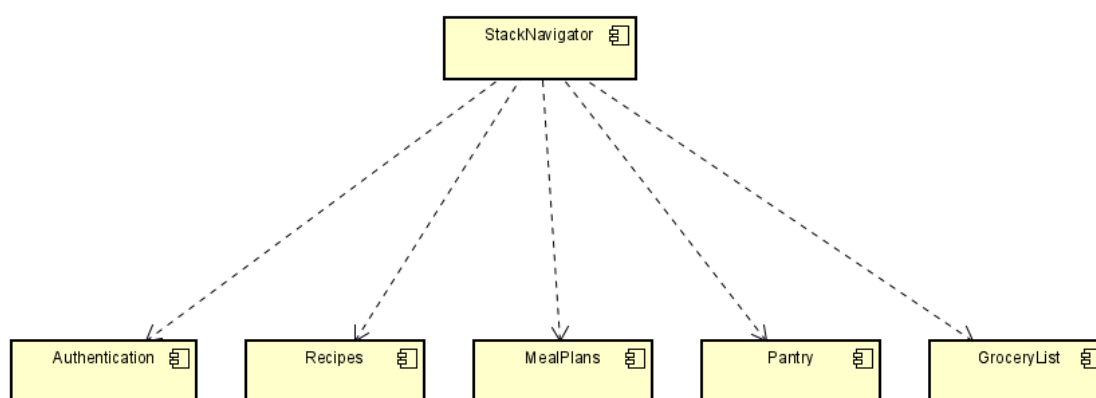
```
const handleChange = (name, value) => {
  setFoodData({
    ...foodData,
    [name]: value,
  });
};
```

Slika 30. Isječak koda - izmjena parametara korištenjem "useState"

3.2.2. Pogledi

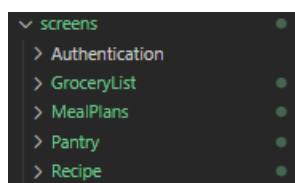
Pogled u ovoj mobilnoj aplikaciji možemo podijeliti na nekoliko glavnih grupa. Na dijagramu na slici 30. vidljive su grupe:

- *Authentication*
- *Recipes*
- *MealPlans*
- *Pantry*
- *GroceryList*



Slika 31. Dijagram glavne podjele pogleda

Svi pogledi opisani u nastavku unutar strukture projekta grupirani su unutar repozitorija *screens*.



Slika 32. Repozitorij "screens"

Pogled *AllRecipes* koji se nalazi u repozitoriju *Recipe* primjer je pogleda koji ne nudi unos podataka nego samo njihov pregled i dugmad za brisanje i daljnju navigaciju. Za prikaz podataka koristi se sučelje „Flatlist“ [\[7\]](#) u kojem se u parametru *data* podaci koji se trebaju prikazati, u ovom slučaju to je lista objekata koji modeliraju recept, te se potom metodom „renderItem“ svaki od članova te liste zasebno obradi te se njegovi atributi prikazuju kako je definirano kodom na slici 33.

```

<SafeAreaView style={{backgroundColor: '#F3F2F2'}}>
  <TextInput style={styles.input} value={search} placeholder={"Search"} underlineColorAndroid='transparent' onChangeText={(text)=>searchFilter(text)}
  />
  <FlatList
    data={filteredData}
    renderItem={({ item }) => (
      <View style={styles.item}>
        <Pressable onPress={() => navigation.navigate('RecipeDetails', {navigation, item})} >
          <View style={styles.title}>
            <Text style={styles.title}>{item.name}</Text>
            <AntDesign name="delete" size={24} color="red" onPress={() => deleteRecipe(item)}>
          </View>
          <Text style={styles.details}>vrijeme pripreme: {item.prepTime} min | vrijeme kuhanja: {item.cookTime} min</Text>
          <Text style={styles.details}>broj obroka: {item.servingSize} | kategorija: {item.category}</Text>
        </Pressable>
      </View>
    )
  )
  keyExtractor={item => item.id}
  contentContainerStyle={{ paddingBottom: 170 }} />
</SafeAreaView>

```

Slika 33. Kod iz pogleda "AllRecipes"

Pogled *AddGeneralInfo* koji se također nalazi u repozitoriju *Recipe* reprezentativan je pogled koji omogućava unos podataka. Kako bi se omogućio unos koristi se komponenta „TextInput“ [7] koja je zapravo polje za unos podataka. Svaka od tih komponenti ima parametar „onChangeText“ kojim se nakon novog unosa poziva metoda koja obrađuje i sprema taj novi unos. Na kraju se nalazi i dugme koje omogućuje „slanje“ podatak i u ovom slučaju dodavanje novog recepta u bazu podataka. Isječak koda iz pogleda *AddGeneralInfo* na slici 34.

```

<SafeAreaView style={{}>
  <View style={styles.titleContainer}>
    <Text style={styles.title}>Dodavanje novog recepta:</Text>
  </View>
  <View>
    <TextInput
      label="Name"
      returnKeyType="next"
      defaultValue={recipeData.name}
      mode='outlined'
      style={styles.input}
      onChangeText={(text) => handleChange('name', text)}
    />

    <TextInput
      label="Category"
      returnKeyType="next"
      defaultValue={recipeData.category}
      mode='outlined'
      style={styles.input}
      onChangeText={(text) => handleChange('category', text)}
    />

    <TextInput
      label="Vrijeme pripreme"
      returnKeyType="next"
      defaultValue={recipeData.prepTime}
      mode='outlined'
      style={styles.input}
      onChangeText={(text) => handleChange('prepTime', text)}
    />

    <TextInput
      label="Vrijeme kuhanja"
      returnKeyType="next"
      defaultValue={recipeData.cookTime}
      mode='outlined'
      style={styles.input}
      onChangeText={(text) => handleChange('cookTime', text)}
    />

    <TextInput
      label="Broj obroka"
      returnKeyType="next"
      defaultValue={recipeData.servingSize}
      mode='outlined'
      style={styles.input}
      onChangeText={(text) => handleChange('servingSize', text)}
    />

    <View style={styles.submitButtonCon}>
      <Text style={styles.submitButton} onPress={() => addNewRecipe(navigation, recipeData)}> DODAJ </Text>
    </View>
  </View>
</SafeAreaView>

```

Slika 34. Kod iz pogleda "AddGeneralInfo"

Authentication (autentifikacija)

Ukupno 4 pogleda vezana su uz autentifikaciju.

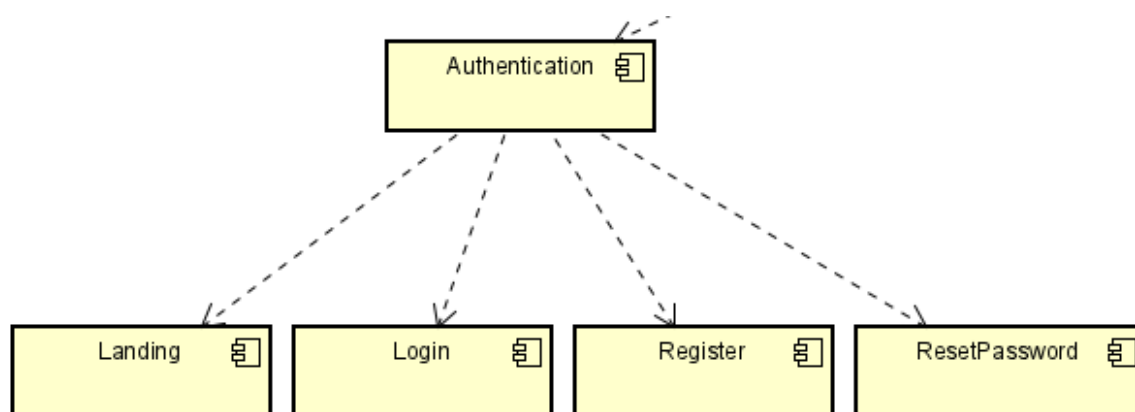
Landing.js – početni pogled na kojem se nalaze opcije za prijavu ili registraciju

Login.js – pogled u kojem se nalazi forma za unos korisničkih vjerodajnica

Register.js – pogled u kojem se nalazi forma za registraciju novog korisnika

ResetPassword.js – pogled u kojem korisnik ima mogućnost izmjene zaboravljene lozinke

Na slici 35. prikazan je dijagram pogleda vezanih uz autentifikaciju korisnika.



Slika 35. Dijagram pogleda autentifikacije

Recipes (recepti)

Na slici 36. vidljivi su svi pogledi koji su povezani s prikazom podataka o receptima, dodavanjem novog recepta i slično.

AllRecipes – prikazuje popis svih korisnikovih recepata, sadrži dugme za dodavanje novog recepta i okvir za pretraživanje

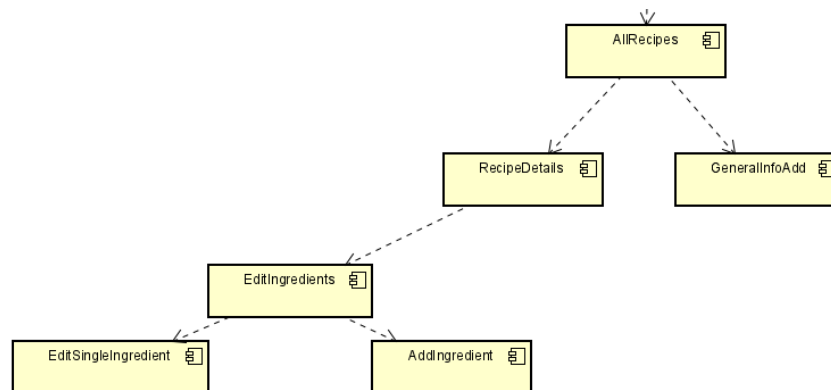
RecipeDetails – pogled koji sadrži informacije i popis sastojaka za pojedini recept, sadrži dugme za označavanje određenog sastojka potrošenim i dugme za stavljanje sastojka na popis za trgovinu te dugme za pristup pogledu za uređivanje podataka o sastojcima

GeneralInfoAdd – pogled koji sadrži obrazac za stvaranje novog recepta, nakon upisa općih informacija, korisnika se preusmjerava na pogled u kojem može dodavati sastojke

AddIngredient – pogled u kojem korisnik ispunjava obrazac za dodavanje novog sastojka za odabrani recept

EditIngredients – pogled koji sadrži popis svih sastojaka te su korisniku ponuđene opcije izmjene postojećih sastojaka, dodavanja novog sastojka ili brisanje sastojka

EditSingleIngredient – pogled za izmjenu podataka određenog sastojka



Slika 36. Dijagram pogleda vezanih uz recepte

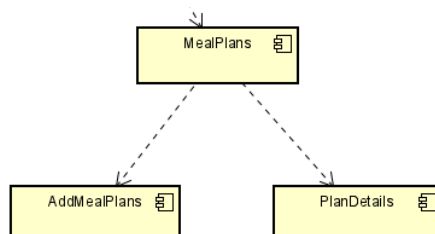
MealPlans (planovi prehrane)

Postoje ukupno 3 pogleda vezana za planove prehrane koji su opisani u nastavku teksta te prikazani na dijagramu na slici 37.

MealPlans – pogled koji prikazuje planirani obrok za današnji dan, dugmad za prikaz budućih i prošli planiranih obroka, dugme za dodavanje novog plana te dugme za izvoz plana prehrane u PDF format

PlanDetail – pogled koji prikazuje recept i raspoloživost sastojaka u datom trenutku

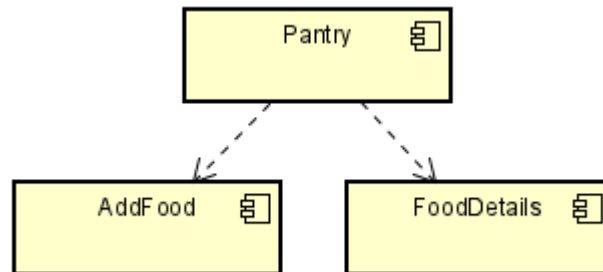
AddMealPlan – pogled koji korisniku omogućuje odabir datuma i recepta koji želi pripremati na odabrani datum, sadrži dugme za filtriranje recepta na način da prikaže samo one recepte za koje korisnik u datom trenutku ima sve sastojke u smočnici



Slika 37. Dijagram pogleda vezanih uz plan prehrane

Pantry (smočnica)

Postoje ukupno 3 pogleda vezana za prikaz podataka i funkcionalnosti vezanih uz „smočnicu“: *Pantry*, *FoodDetail* i *AddFood* koji su opisani u nastavku teksta te su vidljivi na dijagramu na slici 38.



Slika 38. Pogledi vezani uz "Pantry"

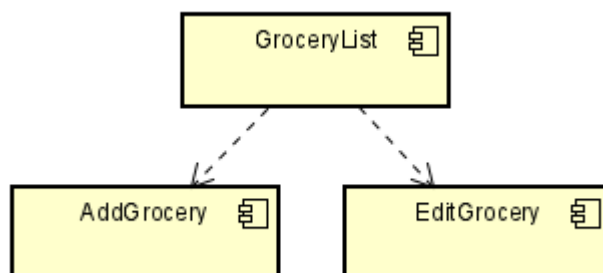
Pantry – prikaz svi namirnica koje se trenutno nalaze u smočnici, sadrži okvir za pretraživanje te dugme za dodavanje nove namirnice u bazu podataka

FoodDetails – prikaz obrasca s trenutnim podacima koje je moguće izmijeniti

AddFood – pogled koji sadrži obrazac pomoću kojeg je moguće dodati novi zapis u Firestore bazu podataka

GroceryList (popis za trgovinu)

Pogledi vezani uz „popis za trgovinu“ jesu: *GroceryList*, *AddGrocery* i *EditGrocery* koji su detaljnije opisani u nastavku te su vidljivi na dijagramu na slici 39.



Slika 39. Pogledi vezani uz popis za trgovinu

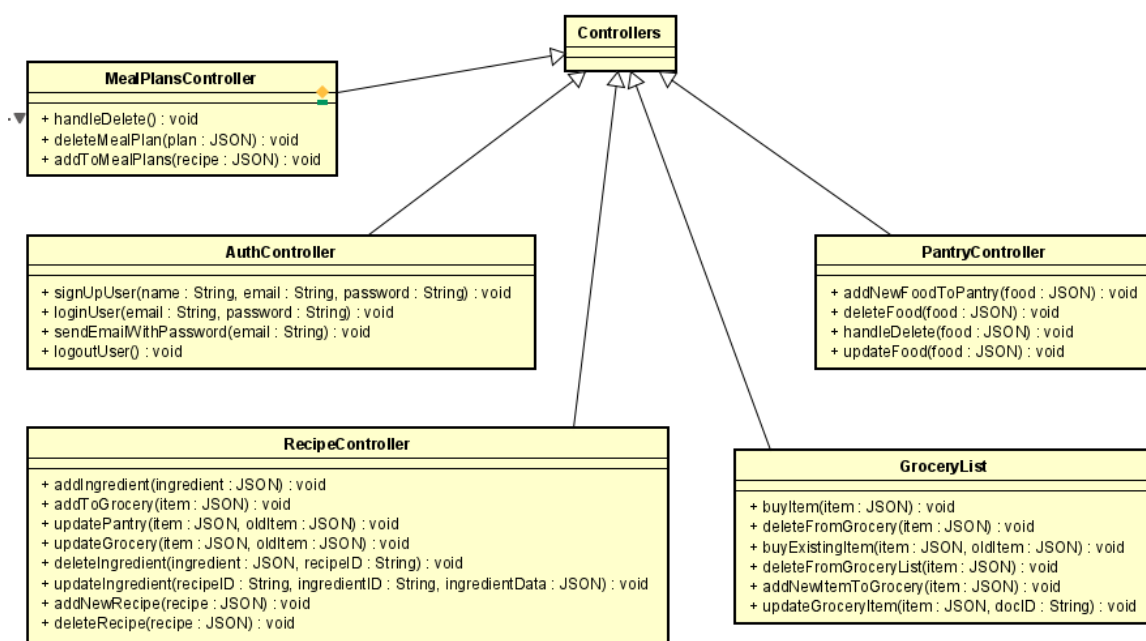
GroceryList – pogled koji sadrži popis svih namirnica koje je potrebno kupiti, okvir za pretraživanje, dugmad za dodavanje, brisanje i označavanje stavke kupljenom, dugme za izvoz popisa za trgovinu u .PDF format

AddGrocery – pogled koji ima obrazac za dodavanje nove namirnice na popis za trgovinu

EditGrocery – pogled koji ima obrazac za izmjenu podataka vezanih uz namirnicu koja se već nalazi na popisu za trgovinu

3.2.3. Kontroleri

Svi kontroleri nalaze se u mapi „**Controllers**“. Kontroleri sadrže metode kojima se upravlja sadržajem koji se prikazuje na ekranu te se ostvaraju pojedine akcije zadane od strane korisnika. U nastavku ovog odlomka biti će navedeni svi kontroleri te ukratko opisane metode koje se nalaze u njima, a na slici 40. vidljiv je dijagram razreda kontrolera.



Slika 40. Dijagram razreda za kontrolere

AuthController

AuthController za autentifikaciju korisnika te pritom koriste metode iz sučelja **Firebase Auth** [16]. Metode koje se nalaze u kontroleru opisane su u nastavku.

signUpUser: metoda koja prima parametre ime, email i lozinka te provodi registraciju korisnika

loginUser: metoda koja prima korisnikove vjerodajnice i provodi prijavu korisnika u mobilnu aplikaciju

sendEmailWithPassword: metoda koja korisniku omogućava ponovno postavljanje lozinke

logoutUser: metoda koja odrađuje odjavu korisnika iz mobilne aplikacije

RecipesController

RecipesController sadrži metode koje se koriste za manipuliranje podacima i pogledima koji se nalaze u repozitoriju „*screens/Recipes*“. Neke od metoda koje se koriste su:

addIngredient: metoda koja dodaje novi sastojak u bazu podataka za postojeći recept

addToGrocery: metoda koja dodaje sastojak na popis za trgovinu, poziva se u slučaju kada na popisu za trgovinu nije postajala stavka koja odgovara odabranom sastojku

updatePantry: metoda koja u bazi podataka smanjuje količinu određene namirnice koju korisnik ima kod kuće, poziva se kada korisnik želi označiti da je iskoristio određenu količinu neke namirnice, čija je količina definirana u sastojcima recepta

deleteIngredient: metoda koja briše odabrani sastojak s popisa sastojaka recepta

updateIngredient: metoda koja izmjenjuje podatke već postojećeg sastojka u željenom receptu

addNewRecipe: metoda kojom korisnik stvara novi recept, u bazi podataka stvara novi dokument s predanim podacima te unutar tog dokumenta stvara praznu podkolekciju „*ingredients*“

deleteRecipe: metoda koja iz baze podataka briše odabrani recept i njegovu podkolekciju „*ingredients*“

GroceryListController

GroceryController sadrži metode koje se koriste za manipulaciju i izmjenu podataka vezanih uz popis za trgovinu. Metode koje se nalaze u *GroceryListControlleru* su:

buyItem: metoda koja se poziva kada korisnik želi u bazu podataka dodati novu namirnicu koju sada ima kod kuće jer je obavio kupnju

buyExistingItem: metoda koja se poziva kada korisnik želi u bazu podataka izmijeniti količinu koju posjeduje kod kuće za već postojeću namirnicu jer je kupio nove zalihe pa nije potrebno stvarati novi dokument nego samo ažurirati postojeći

deleteFromGrocery: metode *buyItem* i *buyExistingItem* brišu odabrani zapis s popisa za trgovinu ove metode

deleteFromGroceryList: poziva se kada korisnik želi jednostavno obrisati određeni zapis s popisa za trgovinu, ali ga ne označava kupljenim

addNewItemToGrocery: metoda koja u Firestore bazi podataka stvara novi dokument u kolekciji popisa za trgovinu

updateGroceryItem: metoda koja u Firestore bazi podataka izmjenjuje postojeći dokument za određenu namirnicu koja se nalazi na popisu za trgovinu, poziva se kada korisnik želi izmijeniti količinu koju je potrebno kupiti

MealPlansController

MealPlansController kontroler je koji sadrži metode koje se koriste za dodavanje i brisanje planova prehrane iz Firestore baze podataka, odnosno kolekcije „mealPlans“. Kratki opisi tih metoda navedeni su u nastavku.

handleDelete: metoda koja se poziva nakon što korisnik dodirne dugme za brisanje, stvara obavijest kojom se provjerava je li korisnik siguran da želi obrisati željeni plan prehrane

deleteMealPlan: metoda koja se poziva nakon što korisnik potvrdi da želi izvršiti brisanje, metoda briše zadani dokument iz Firestore baze podataka

addToMealPlans: metoda kojom se u Firestore bazu podataka dodaje novi dokument kada korisnik odluči napraviti novi plan prehrane za odabrani datum

PantryController

PantryController je kontroler u kojem se nalaze metode korištene za upravljanje podacima vezanih za smočnicu. Kratak pregled metoda nalazi se u tekstu ispod.

handleDeleteFood: metoda koja se poziva nakon što korisnik dodirne dugme za brisanje, stvara obavijest kojom se provjerava je li korisnik siguran da želi obrisati odabranu namirnicu iz baze podataka

deleteFood: metoda koja se poziva nakon što korisnik potvrdi da želi izvršiti brisanje, metoda briše zadani dokument iz Firestore baze podataka

addNewFoodToPantry: metoda koja stvara novi dokument u Firestore bazi podataka na temelju podataka koje je korisnik unio u obrascu za dodavanje nove namirnice

updateFood: metoda koja ažurira postojeće podatke u bazi podataka izmijenjenim podacima koje je korisnik unio putem obrasca za izmjenu podataka

3.3. Izvoz/ispis podataka u PDF formatu

Implementirana je funkcionalnost izvoza i ispisa rasporeda prehrane i popisa za trgovinu. Za ostvarivanje navedenog korišten je paket „*expo-print*“ te metoda iz tog paketa „**Print**“. Isječak koda u kojemu se koristi metoda „print“ nalazi se na slici 41.

```
const print = async () => {  
  await Print.printAsync({  
    html: createDynamicTable(),  
    printerUrl: selectedPrinter?.url,  
  });  
}
```

Slika 41. Isječak koda - korištenje metode "Print"

U metodi *createDynamicTable* su korištenjem HTML jezika formatirani podaci čiji se izvoz, odnosno ispis traži. Metoda je prikazana na slici 42. Podaci se metodi predaju lista JavaScript objekata od kojih svaki ima određene atribute koji se ispisuju. Na slici 42. se radi o popisu za trgovinu što znači da svaki objekt ima atribute : *name*, *amount* i *item*. Korištenjem petlje iterira se kroz listu podataka te se za svaki podatak dodaje novi redak u tablici.

```
const createDynamicTable = () => {  
  var table = '';  
  for (let i in grocery) {  
    const item = grocery[i];  
    table = table + `  
    <tr>  
      <td>${item.name}</td>  
      <td>${item.amount}</td>  
      <td>${item.unit}</td>  
    </tr>  
    `;  
  }  
}
```

Slika 42. Isječak koda - metoda "createDynamicTable"

3.4. Objavljivanje obavijesti

Dobra je praksa kod implementiranja brisanja podataka prije izvršavanja brisanja na neki način provjeriti je li korisnik stvarno želio izvršiti tu akciju. Pri razvoju ove mobilne akcije se taj proces odvija putem „skočnih prozora“ odnosno obavijesti ili upozorenja korisniku. Za objavljivanje obavijesti koristi se komponenta „Alert“ [15]. Na slici 43. nalazi se isječak koda u kojemu je dan primjer korištenja navedene komponente. Metoda `alert()` ima dva parametra tipa *string*, prvi se odnosi na naslov obavijesti – „Upozorenje“, a drugi na tekst obavijesti – „Potvrdi brisanje“. Unutar uglatih zagrada definirana je dugmad kojom se potvrđuje ili prekida izvršavanje operacije brisanja podataka.

```
// alert before deletion
export function handleFoodDelete(food, foodsRef)
{
  Alert.alert(
    "Upozorenje",
    "Potvrdi brisanje?",
    [
      {
        text: "Cancel",
        onPress: () => console.log("Cancel Pressed"),
        style: 'cancel'
      },
      { text: "OK",
        style: 'destructive',
        onPress: () => deleteFood(food, foodsRef) }
    ]
  );
}
```

Slika 43. Isječak koda - korištenje komponente "Alert"

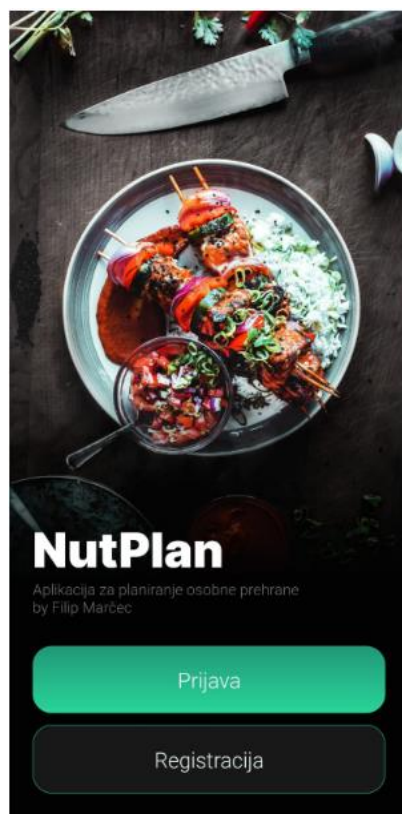
4. Korisničko sučelje aplikacije

Mobilna Aplikacija sastoji se od 4 glavna pogleda odnosno ekrana. Svaki od njih bit će opisan u ovom poglavlju te će uz njih biti opisani i načini pristupa određenim funkcionalnostima, a sve će biti potkrijepljeno i slikama ekrana iz aplikacije.

4.1. Autentifikacija – prijava/registracija

4.1.1. Početna stranica

Na početnoj stranici nalaze se dvije opcije: prijava ukoliko već posjeduje račun te registracija za nove korisnike. Na slici 38. vidljiva je opisana početna stranica.



Slika 44. Početna stranica

4.1.2. Registracija

Dodirom na dugme „Registracija“ koje se nalazi na početnoj stranici, korisnik otvara ekran na kojemu se nalazi obrazac za registraciju. Potrebno je unijeti ime, e-mail adresu i lozinku. Po završetku ispunjavanja obrasca potrebno je dodirnuti dugme „pošalji“.

4.1.2. Prijava

Dodirom na dugme „Prijava“ na početnoj stranici, korisniku se omogućava pristup ispunjavaju obrasca sa svojim vjerodajnicama za prijavu: e-mail i lozinka. Osim prijave, postoji i opcija oporavka zaporke. Opisani obrasci vidljivi su na slici 45.

The image shows three mobile app screens with a grid background and a chef icon at the top of each.

- Registriraj se:** Contains input fields for 'Ime', 'Email', and 'Lozinka', followed by a green 'Pošalji' button.
- Dobrodošli natrag:** Contains input fields for 'Email' and 'Lozinka', a green 'Login' button, and a link 'Zaboravljena lozinka?'.
- Oporavak lozinke:** Contains an 'E-mail addressa' input field, a note 'Dobit ćete mail s uputama za oporavak lozinke', and a green 'Pošalji upute' button.

Slika 45. Obrasci za registraciju, prijavu i oporavak lozinke

4.2. Moji recepti

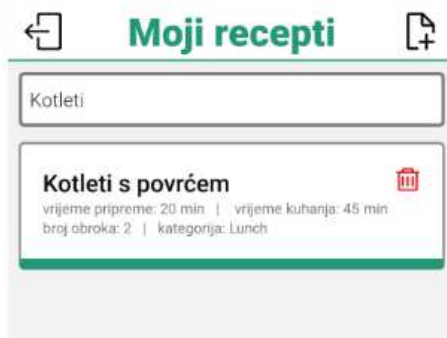
Prvi ekran koji korisnik vidi nakon uspješne prijave jest ovaj na kojem su prikazani svi recepti koje korisnik ima zapisane, odnosno koji su pohranjeni u bazi podataka. Korisnik može koristiti brojne funkcionalnosti koje će biti opisane u sljedećih nekoliko odlomaka. Na slici 46. vidljiv je prikaz „Moji recepti“.



Slika 46. Prikaz "Moji recepti"

4.2.1. Okvir za pretraživanje

Pri vrhu ovog prikaza nalazi se okvir za pretraživanje koji korisniku omogućuje pretraživanje vlastitih recepata prema njihovom nazivu. Na slici 47. je vidljiva opisana funkcionalnost



Slika 47. Prikaz funkcionalnosti pretraživanja recepata

4.2.2. Dodavanje novog recepta

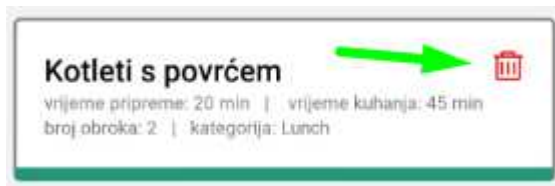
Novi recept korisnik može dodati dodirom na ikonu u gornjem lijevom kutu. Otvara se novi ekran na kojem se nalazi obrazac gdje korisnik unosi opće informacije o receptu: naziv, vrijeme pripreme i kuhanja, kategoriju i broj obroka. Po završetku ispunjavanja tih polja, korisnik treba dodirnut dugme „DODAJ“ čime će se u bazi stvoriti novi dokument te prazna kolekcije „ingredients“, a korisnika će se preusmjeriti na ekran za dodavanje sastojaka za novostvoreni recept. Opisani obrazac prikazan je na slici 48.

Slika 48. Obrazac za dodavanje novog recepta

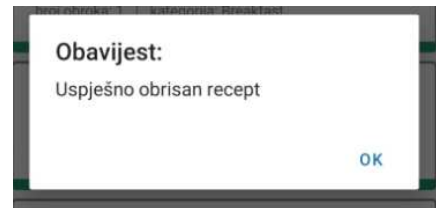
4.2.3. Brisanje postojećeg recepta

Svaka od „kartica“ recepata na sebi ima i crvenu ikonicu „kante za otpad“ čijim se dodirom korisniku postavlja upit je li siguran da želi obrisati recept, te ukoliko korisnik to potvrdi,

recept će se izbrisati iz baze podataka, a o uspješnosti tog procesa korisnik će biti obaviješten putem obavijesti.



Slika 50. Ikona za brisanje recepta



Slika 49. Obavijest o uspješnosti brisanja recepta

4.2.4. Pregled pojedinačnog recepta

Dodirom na „karticu“ recepta u prikazu svih recepata otvara se novi ekran s detaljnim prikazom podataka o receptu. Na samom vrhu nalazi se naziv recepta, a ispod naziva kratka obavijest u crvenoj boji ako korisnik nema sve sastojke kod kuće, odnosno zelenoj boji ako korisnik ima sve sastojke potrebne za pripremu tog recepta. Ekran je podijeljen na dva glavna dijela, u prvom su navedene općenite informacije o receptu, a u drugom se nalazi popis sastojaka te dugme za uređivanje sastojaka.

Na popisu sastojaka, svaki od recepata ima dvije, a neki sastojci i tri ikone. Njihova značenja objašnjena su ispod.



Slika 51. Primjeri prikaza detalja o receptu



- označava da korisnik u danom trenutku posjeduje zadanu količinu tog sastojka



- označava da korisnik u danom trenutku NE posjeduje zadanu količinu tog sastojka



- dodirrom na ovu ikonicu korisnik na popis za trgovinu dodaje ovaj sastojak u količini koja je zadana receptom



- dodirrom na ovu crvenu ikonicu „minus“ korisnik ažurira podatke za svoju smočnicu i umanjuje količinu kojom raspolaže jer je u pripremi upotrijebio količinu određenom receptom

Dodirrom dugmeta „Uredi sastojke“ korisnik pristupa ekranu za izmjenu sastojaka za odabrani recept.

4.2.5. Uređivanje sastojaka

Na ovom ekranu korisnik može izmijeniti sastojke za odabrani recept. Dodirrom na crvenu ikonicu koja se nalazi na desnom rubu „kartice“ pojedinog sastojka otvara se upozorenje kojim se provjerava s korisnikom želi li stvarno obrisati odabrani sastojak, te ako korisnik odgovori potvrdno, sastojak se briše iz baze podataka te ga se kroz obavijest izvještava o uspješnosti tog procesa.



Slika 52. Prikaz uređivanja sastojaka za recept "Kotleti s povrćem"

Postojeći sastojak može se izmijeniti dodirom na „karticu“ te se tada otvara obrazac gdje korisnik može izmijeniti naziv, količinu ili mjernu jedinicu. Novi sastojak dodaje se dodirom na dugme na dnu prikaza „Dodaj sastojak“ čime se otvara obrazac u koji se od korisnika traži da upiše naziv, količinu i mjernu jedinicu. Opisani obrasci vidljivi su na slici 53.



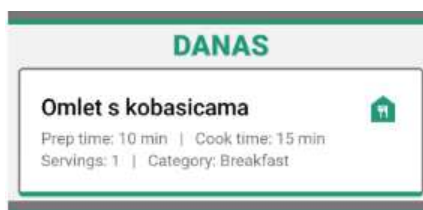
Slika 53. Obrasci za dodavanje i uređivanje sastojka

4.3. Raspored kuhanja

Na ovom ekranu korisnik može vidjeti koji recept je isplanirao da će danas kuhati, može pregledati što ima isplanirano za dane koji dolaze ili što je kuhao prethodnih dana kako bi mogao kontrolirati da jede raznovrsno. Ove navedene i ostale funkcionalnosti opisane su u nastavku.

4.3.1. Plan za današnji dan

Ispod teksta „danas“ nalazi se naziv i opće informacije o receptu kojeg je korisnik isplanirao da će pripremati na današnji dan, a ako nije ništa isplanirao umjesto informacija o receptu jednostavno će pisati „Ništa isplanirano“.



Slika 54. Plan za danas

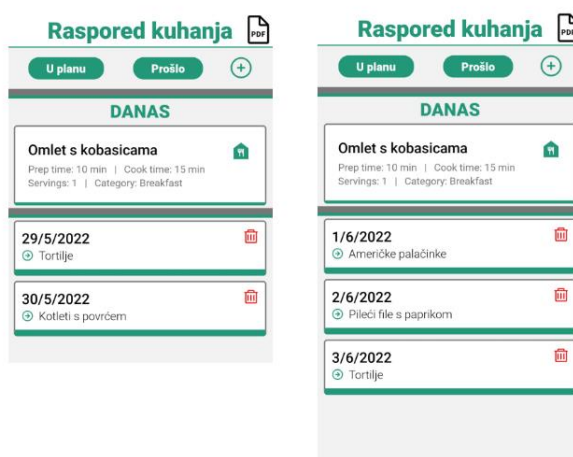
4.3.2. Pregled budućih i prošlih planova

Pritiskom na jedno od dugmadi sa slike 55. korisnik može pregledati planove koji tek slijede, odnosno pregledati što je pripremao prethodnih dana.



Slika 55. Filtriranje planova

Kako izgleda popis odabranih planova vidljivo je na slici 56.



Slika 56. Filtriranje planova kuhanja

4.3.3. Brisanje plana

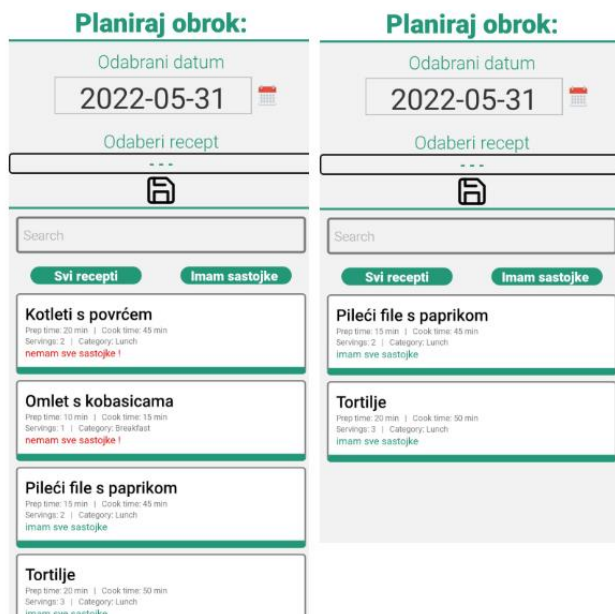
Svaka „kartica“ za pojedini na plan na desnom rubu ima crvenu ikonicu „kante za smeće“ što je zapravo dugme koji korisnik može obrisati odabrani plan. Prije nego ga u potpunosti obriše iz baze podataka, korisnika će se upitati je li siguran da želi izbrisati odabrani plan.



Slika 57. Dugme za brisanje plana

4.3.4. Dodavanje novog plana

Dodirom na zeleno dugme „plus“ pored dugmadi za filtraciju planova otvara se ekran gdje je korisniku omogućen odabir datuma i recepta koji želi pripremati na odabrani datum.



Slika 58. Primjeri ekrana za planiranje obroka

Odabir datuma vrši se dodirom na ikonu kalendara nakon čega se otvara skočni prozor u kojem korisnik na kalendaru bira datum. Pritiskom na dugme „Svi recepti“ korisnik pregledava sve recepte, a pritiskom na dugme „Imam sastojke“ korisnik može odabrati samo one recepte za koje trenutno ima sve sastojke prema onima koji su evidentirani u smočnici. Također je moguće i pretraživati recepte unosom naziva u polje za pretraživanje.

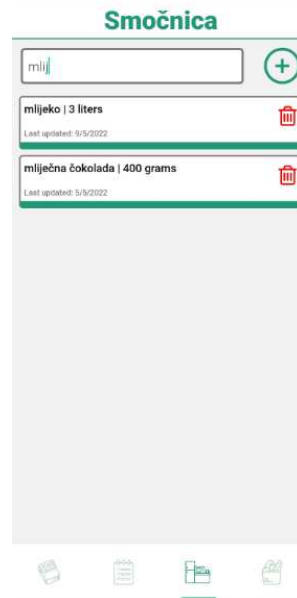
Kada korisnik odabere datum i recept, njegov trenutni odabir prikazuje se u crnom okviru, a da bi dodao taj plan u bazu podataka, potrebno je dodirnuti crnu ikonicu „diskete“.

4.4. Smočnica

Dodirom na treću ikonu na navigacijskoj traci na dnu ekrana korisnik može pristupiti ekranu „Smočnica“ gdje može voditi evidenciju namirnica koje trenutno ima kod kuće, odnosno njihovu količinu. Ostale funkcionalnosti opisane su u nastavku.

4.4.1. Okvir za pretraživanje

Implementirana je funkcionalnost pretraživanja namirnica koje se trenutno nalaze na popisu namirnica u smočnici. Unosom teksta, korisnik će dobiti filtrirani popis namirnica kao što je vidljivo na slici 59.



Slika 59. Pretraživanje smočnice

4.4.2. Dodavanje novog zapisa u smočnicu

Dodirom zelenog dugmeta „plus“, desno od okvira za pretraživanje korisniku se otvara ekran na kojem se nalazi obrazac za unos i evidenciju nove namirnice u smočnici. Kada završi s unosim podataka, korisnik treba dodirnuti dugme „DODAJ“. Prilikom stvaranja novog

The screenshot shows a mobile application interface titled 'Dodaj u smočnicu:'. It features a form with three input fields: 'Naziv' (Name), 'Količina' (Quantity), and a unit selector currently set to 'litra'. Below the form is a large green button with the text 'DODAJ' in white capital letters.

Slika 61. Obrazac za dodavanje novog zapisa u smočnicu

dokumenta u bazi, za atribut *lastUpdated* postaviti će se trenutno vrijeme i datum. Na slici 61. je vidljiv ekran za dodavanje novog zapisa.

4.4.3. Izmjena podataka za postojeću namirnicu u smočnici

Dodirom na željenu namirnicu, korisnik može pristupiti obrascu koji mu omogućava izmjenu podataka za već postojeću namirnicu. Može izmijeniti naziv ukoliko je pogriješio prilikom unosa, ili može izmijeniti količinu i mjernu jedincu kada potroši određenu količinu namirnice pa se njena raspoloživost smanji. Opisani proces završava dodirom na dugme „SPREMI“. Na slici 62. je dan primjer takvog obrasca.



Uredi podatke za:
jagode

jagode

400

g

grams

SPREMI

Slika 62. Obrazac za izmjenu podataka u smočnici

4.4.4. Brisanje zapisa iz smočnice

Svaki zapis na sebi ima crvenu ikonicu kante za smeće koja je zapravo dugme koje korisnik može dodirnuti kako bi obrisao određenu namirnicu. Prilikom dodira na dugme za brisanje korisnika će se upozoriti i upitati je li siguran da želi obrisati tu stavku, te ukoliko odabere opciju „Ok“ stavka će se obrisati iz Firestore baze podataka te će se o uspješnosti tog procesa korisnika obavijestiti kroz obavijest.

4.5. Popis za trgovinu

Dodirom posljednje ikone s desna na navigacijskoj traci na dnu ekrana korisnika pristupa ekranu „Popis za trgovinu“ koji je vidljiv na slici 63. Na ekranu je vidljiv popis namirnica koje je potrebno kupiti te nekoliko dugmadi čije funkcionalnosti su opisane u nastavku.



Slika 63. Prikaz ekrana "Popis za trgovinu"

4.5.1. Okvir za pretraživanje

Implementirana je funkcionalnost pretraživanja namirnica koje se nalaze na popisu za trgovinu. Unosom teksta, namirnice se filtriraju prema nazivu i u prikazu ostaju samo one koje odgovaraju pretraženom tekstu kao što je vidljivo na slici 64.



Slika 64. Funkcionalnost pretraživanja popisa za trgovinu

4.5.2 Dodavanje novog zapisa na popis za trgovinu

Dodirom zelene ikonice „plus“, lijevo od okna za pretraživanje, otvara se ekran za dodavanje nove namirnice na popis za trgovinu koje je na slici 65. označeno plavim kvadratom. Korisnik može unijeti ime namirnice, količinu te odabrati mjernu jedinicu kroz padajući izbornik. Konačno, proces dodavanja novog zapisa na popis završava dodirom dugmeta na ispod padajućeg izbornika.

Popis za trgovinu

+

goveđi temeljac 3 x		
keksi 800 g		
kobasica 400 g		
ledeni čaj 3 L		
maslac 250 g		
maslinovo ulje 120 mL		
meksička mješavina 1 x		
mlijeko 200 mL		
mrkva 900 g		
paprika 3 x		
tortilje 4 x		
češnjak 30 g		
čokolada 300 g		

Dodaj na popis za trgovinu:

Name

Amount

kilogram ▼ **kg**

DODAJ

Slika 65. Prikaz ekrana za dodavanje zapisa na popis za trgovinu

4.5.3. Uređivanje zapisa s popisa za trgovinu

Dodirom na onaj zapis za koji želi urediti podatke, korisnik otvara ekran koji mu omogućava izmjenu naziva ukoliko je prilikom dodavanja napravio pravopisnu pogrešku, izmjenu količine te izmjenu mjerne jedinice putem padajućeg izbornika. Proces promjene podataka za odabrani zapis završava dodirom dugmeta „SPREMI“ ispod padajućeg izbornika za odabir mjerne jedinice. Opisana forma vidljiva je na slici 59.

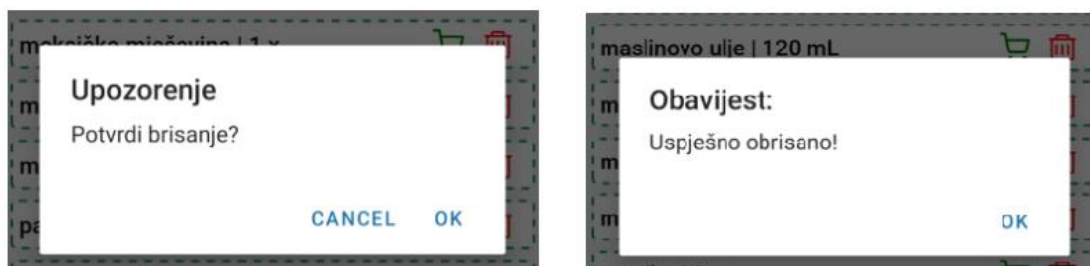


Slika 66. Prikaz ekrana za uređivanje zapisa na popisu za trgovinu

4.5.4 Označavanje zapisa kao kupljenog i brisanje zapisa s popisa za trgovinu

Svaki od zapisa osim informacija o nazivu i količini koju treba kupiti, na sebi ima još i dvije ikonice. Dodirom zelene ikonice „kolica“, korisnik označava odabrani proizvod kupljenim, te se količina koja mu je u tom trenutku pridijeljena dodaje na stanje prilikom čega se korisniku daje obavijest o uspješnosti tog procesa s tekstom: „Dodano na stanje“.

Dodirom crvene ikonice, korisnik započinje proces brisanja zapisa s popisa za trgovinu. Nakon dodira te ikonice, otvara se upozorenje koje s korisnikom provjerava želi li stvarno obrisati odabrani zapis s popisa za trgovinu, te ako korisnik odabere opciju „ok“, zapis će se izbrisati s popisa te će se uspješnost tog procesa korisniku javiti kroz obavijest.



Slika 67. Prikaz obavijesti prilikom brisanja zapisa s popisa za trgovinu

4.5.6. Izvođenje popisa za trgovinu u PDF formatu

Implementirana je funkcionalnost izvođenja trenutnog popisa za trgovinu u .PDF formatu. Dodirom ikonice na kojoj piše „PDF“, u desnom kutu ekrana, otvara se prozor u kojem se nalaze 2 opcije: spremanje popisa u PDF formatu ili njegov ispis na nekom od dostupnih pisača. Izgled PDF dokumenta vidljiv je na slici 68.



Slika 68. Prikaz izgleda popisa za trgovinu u PDF formatu

5. Korištene tehnologije

5.1. React Native

React Native [4] je radni okvir zasnovan na JavaScriptu koji se koristi za razvoj mobilnih aplikacija za platforme iOS i Android. Velika prednost *React Native* što omogućuje razvojnim programerima da koriste identičan bazni kod za razvoj aplikacija za naveden platforme. *React Native* izgrađen je na temeljima *Reacta* [6], JavaScript biblioteke namijenjene za izradu web stranica.

5.2. EXPO

Expo [9] je radni okvir čija je namjena izgradnja *React Native* aplikacija. Sastoji se od seta alata i servisa izrađenih posebno za *React Native* kako bi razvojnim programerima olakšao početak izgradnje *React Native* aplikacija te njihovo testiranje. Iako postoje određena ograničenja, *Expo* osigurava vrlo brzo *prvo* pokretanje aplikacije u danu, omogućuje pregled onoga što se kodira na vlastitom mobilnom uređaju putem *Expo* mobilne aplikacije, što znači da nema potrebe za korištenje dodatnih emulatora ili čak spajanjem mobilnog uređaja s računalom. Također, objavljivanje aplikacije obavlja se kroz nekoliko naredbi u *Expo* terminalu.

5.3 Firebase

Firebase [13] je *Backend-as-a-Service*. To je platforma koja razvojnim programerima pruža razne alate i usluge za izradu aplikacija izgrađena na Googleovoj infrastrukturi. Neke od najznačajnijih usluga koje *Firebase* pruža jesu : *Firebase Auth*, *Firebase Cloud Messaging*, *Firebase Cloud Firestore*, *Firebase Realtime Database*, *Firebase Storage*, *Firebase Hosting* i druge. Usluge korištene pri izradi ove aplikacije navedene su i kratko opisane u nastavku.

5.3.1. Firebase Cloud Firestore

Firebase Cloud Firestore [10] je „cloud-hosted“ NOSQL baza podataka namijenjena za pohranu i sinkroniziranje podataka. Dizajnirana je tako da podržava razvoj jednostavnih, ali i složenih aplikacija. Smanjuje količinu potrebnog kodiranja jer odrađuje poslužiteljski dio zadataka. U ponudi je nekoliko cjenovnih razreda, uključujući i onaj besplatni.

5.3.2 Firebase Auth

Firebase Auth [16] je set alata i usluga koji obavljaju poslužiteljske zadatke autentifikacije korisnika koji žele pristupiti aplikaciji. *Firebase Auth* podržava mnogo

različitih vrsta autentifikacije od kojih su neke: lozinka, broj mobitela, Google račun, Facebook račun, Twitter račun, GitHub račun i mnogi drugi.

5.4. Visual Studio Code

Visual Studio Code je uređivač koda koji se može koristiti na Windows, macOS i Linux operacijskim sustavima. To je alat s odličnom ugrađenom podrškom za kodiranje u JavaScriptu, TypeScriptu i NodeJS-u.

5.5. Astah UML

Astah UML [\[11\]](#) je desktop aplikacija namijenjena za izradu UML dijagrama kao što su dijagrami komponenti, dijagrami obrazaca uporabe, dijagrami stanja, dijagrami aktivnosti i mnogih drugih.

5.6. ERDPlus

ERDPlus [\[12\]](#) je web aplikacija namijenjena za izradu konceptualnih dijagrama baze podataka i generiranje SQL naredbi.

Zaključak

Razvojem moderne tehnologije postaje logično da stvari poput planiranja prehrane koje su se obično zapisivale na papir i ostavljale na hladnjaku kao podsjetnik također budu implementirane kao mobilna aplikacija.

Cilj ovog rada bio je proučiti postojeća rješenja na području aplikacija koje omogućavaju evidenciju recepata i planiranje prehrane te nakon toga izraditi vlastito rješenje. Većina analiziranih mobilnih aplikacija slijedi isti uzorak, no zanimljivo je primijetiti da dominiraju aplikacije koje je nemaju implementiranu evidenciju vlastitih recepata nego stvaranje popisa „omiljenih“ recepata koji se nalazi na nekom drugom izvoru.

Mobilna aplikacija NutPlan ima vrlo jednostavno i intuitivno sučelje. Dizajn korisničkog sučelja rađen je s namjerom da bude pristupačan i jednostavan za korištenje. Implementirane su brojne funkcionalnosti, no ipak ne u tolikom obimu kao neke od analiziranih mobilnih aplikacija.

U daljnjem razvoju ove mobilne aplikacije naglasak bi bio stavljen na povećanje skalabilnosti što bi se ostvarilo boljom reorganizacijom izvornog koda te uvođenje novih funkcionalnosti kao što su mogućnost skeniranja barkoda na kupljenim namirnicama te prikaz nutricionističkih vrijednosti za pojedine recepte.

Literatura

- [1] Recipe Keeper, <https://recipekeeperonline.com/>, 3.4.2022.
- [2] Whisk, <https://whisk.com/>, 3.4.2022.
- [3] Cookmate, <https://www.cookmate.online/hr/home/>, 3.4.2022.
- [4] React Native – Firebase, <https://rnfirebase.io/>, 29.5.2022.
- [5] MVC, <https://en.wikipedia.org/wiki/Model-view-controller>, 1.6.2022.
- [6] ReactJS, <https://reactjs.org/>, 29.5.2022.
- [7] React Native, <https://reactnative.dev/>, 29.5.2022.
- [8] Firestore, <https://firebase.google.com/docs/firestore>, 1.6.2022.
- [9] EXPO, <https://expo.dev/>, 28.5.2022.
- [10] AstahUML, <https://astah.net/>, 1.6.2022.
- [12] ERDPlus, <https://erdplus.com/>, 1.6.2022.
- [13] Firebase, <https://firebase.google.com/docs>, 29.5.2022.
- [14] EXPO - print, <https://docs.expo.dev/versions/latest/sdk/print>, 4.6.2022.
- [15] React Native – Alert, <https://reactnative.dev/docs/alert>, 4.6.2022.
- [16] Firebase Auth, <https://firebase.google.com/docs/auth>, 4.6.2022.
- [17] State Hook, <https://reactjs.org/docs/hooks-state.html>, 4.6.2022.

Sažetak

Naslov: Mobilna aplikacija za planiranje osobne prehrane

Sažetak:

Ovaj rad se bavi opisom postojećih programskih rješenja za planiranje osobne prehrane te opisom izvedbe i funkcionalnosti implementiranih u izrađenoj mobilnoj aplikaciji. U prvom dijelu rada opisane su tri mobilne aplikacije koje se bave navedenom problematikom, nakon čega su navedeni funkcionalni i nefunkcionalni zahtjevi. U nastavku je opisana baza podataka i arhitektura korištena pri izradi programskog rješenja. Nakon toga slijedi opis korisničkog sučelja te su na kraju rada navedene korištene tehnologije, zaključak te korištena literatura.

Razvijena mobilna aplikacija ima mogućnost evidencije vlastitih recepata, planiranje prehrane, vođenje evidencije o raspoloživim namirnicama te stvaranje popisa za trgovinu. Mobilna aplikacija razvijena je korištenje JavaScript radnog okvira React Native te je namijenjena za korištenje na uređajima s Android operacijskim sustavom. Za bazu podataka korišteno je Googleovo rješenje – NOSQL baza podataka Cloud Firestore Database.

Ključne riječi: Android aplikacija, ReactNative, Firebase, Firestore, prehrana, recept, sastojci, popis za trgovinu, planiranje, smočnica

Abstract

Title: Mobile application for personal nutrition planning

Summary:

This paper describes already existing software solutions for personal nutrition planning and gives description of developing process and functionality implemented in the developed mobile application. The first part of the paper describes three mobile applications that deal with this issue, followed by functional and non-functional requirements. Afterwards, the database and architecture in the development of the software solution are described. This is followed by a description of the user interface and at the end of the paper the technologies used, the conclusion and the used literature are listed.

The developed mobile application offers the possibility of managing your own recipes, planning your diet, keeping records of available groceries and creating a list for the store. The mobile application was developed using the React Native JavaScript framework and is intended for use on devices with the Android operating system. Google's NOSQL Cloud Firestore Database solution was used for the database.

Keywords: Android application, React Native, Firebase, Firestore, nutrition, recipe, ingredients, grocery list, planning, pantry