Aplikacija za Android platformu za mjerenja masnoće u hrani

Markanović, Marko

Undergraduate thesis / Završni rad

2020

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

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:761387

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-26**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

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

Stručni studij

Aplikacija za Android platformu za mjerenja masnoće u hrani

Završni rad

Marko Markanović

Osijek, 2020.

Sadržaj

1. Uvod ... 1

1.1 VLCAD ... 2

1.2 Zadatak završnog rada ... 2

2. Opis korištenih tehnologija ... 3

2.1 Android operacijski sustav ... 3

2.2 Java programski jezik ... 4

2.3 XML jezik za označavanje podataka ... 5

2.4 Android Studio ... 6

2.5 Firebase ... 6

3. Model aplikacije za mjerenja masnoće u hrani ... 7

3.1 Početna stranica/stranica za prijavu ... 7

3.2 Stranica za registraciju ... 9

3.2 Stranica za prikaz unosa kalorija i masti ... 11

3.3 Stranica za prikaz svih namirnica u bazi podataka ... 13

3.4 Navigacijski izbornik ... 14

3.5 Stranica za dodavanje nove namirnice ... 15

3.6 Stranica za prikaz osobnih podataka ... 17

4. Baza podataka ... 19

5. Zaključak ... 21

6. Literatura ... 22

1. Uvod

Tema završnog rada je aplikacija za Android platformu za mjerenja masnoće u hrani. Završni je rad namijenjen za pomoć ljudima pri praćenju količine masti koje pojedu. Program je primarno namijenjen za ljude koji imaju specifičnu i relativno novu bolest koja im ne dopušta prekomjerno konzumiranje masti, a radi se o poremećaju beta oksidacije masnih kiselina vrlo dugih lanaca ili VLCAD (eng. Very long chain acyl-Co Adehydrogenase deficiency). U slučaju da je unos masti veći od dopuštene količine, osobe koje pate od ove bolesti mogu imati velikih zdravstvenih problema. Ova aplikacija pruža pomoć oboljelima od navedene bolesti u svakodnevnom životu na način da unosom vrste i količine hrane konzumirane svaki pojedini dan u aplikaciju, ista će im pokazati koliko masti ima odabrana hrana i koliko im je još dopušteno unijeti masti taj dan. Postojeće aplikacije nemaju opciju razdvajanja masti na podvrste (Omega 3 i Omega 6) čije je praćenje ključno za održavanje normalnog života osobama koje imaju VLCAD poremećaj.

Aplikacija je izrađena u Android operacijskom sustavu i razvijena u Android Studio razvojnom okuženju. Korišten je Java objektno orijentirani programski jezik. Za bazu podataka korištena je Firebase Googleova usluga koja je NOSQL baza podataka u stvarnom vremenu koja sprema podatke u JSON formatu.

1.1 VLCAD

Poremećaj beta oksidacije masnih kiselina vrlo dugih lanaca ili VLCAD (eng. Very long chain acyl-CoA dehydrogenase deficiency) je stanje tijela koje onemogućuje ljudskom organizmu pravilnu pretvorbu masti u energiju, posebice u vremenskim periodima bez hrane (tijekom posta). Znakovi i simptomi VLCAD-a tipično se javljaju tijekom ranog djetinjstva i mogu uključivati nizak nivo šećera u krvi (hipoglikemija) te nedostatak energije i slabosti u mišićima. Pogođene osobe su u opasnosti jer se mogu razviti komplikacije kao što su abnormalnosti u radu jetre i srčani problemi opasni za život. Simptomi u adolescenciji ili punoljetnosti obično uključuju bol u mišićima i raspadanje mišićnih vlakana (rabdomioliza). Uništenjem mišićnih vlakana ispušta se protein zvan mioglobin kojeg nakon toga obrađuju bubrezi (mioglobinurija). Mioglobin uzrokuje da urin bude crvene ili smeđe boje.

Kod djece i odraslih, problemi vezani za nedostatak VLCAD-a mogu se aktivirati tijekom perioda posta, bolesti ili vježbanja. Kod djece, ovaj poremećaj je ponekad pogrešno dijagnosticiran kao Reyev sindrom, poremećaj koji se može razviti kod djece dok se oporavljaju od viralnih infekcija kao što su gripa ili vodene kozice. Većina slučajeva Reyevog sindroma povezana je sa korištenje aspirina tijekom viralnih infekcija.

1.2 Zadatak završnog rada

Zadatak završnog rada je razvijati aplikaciju koja prati dnevni unos masti u Android operacijskom sustavu i pomoću Firebase baze podataka. Aplikacija omogućava lakše praćenje unosa masti i zbog „Oblak“ (eng. Cloud) baze podataka omogućava lagan nastavak korištenja aplikacije u slučaju promjene mobilnog uređaja.

2. Opis korištenih tehnologija

Tehnologije koje su korištene za razvoj aplikacije su Android operacijski sustav, Java objektno orijentirani programski jezik, XML jezik za označavanje podataka, Android Studio razvojno okruženje te Firebase NOSQL baza podataka u stvarnom vremenu.

2.1 Android operacijski sustav

Android je mobilni operacijski sustav temeljen na modificiranoj inačici Linux-ovog kernela i drugih „open source“ programa, koji je dizajniran primarno za mobilne uređaje osjetljive na dodir kao što su pametni telefoni i tableti. Android je razvio skup programera poznati kao „Open Handset Alliance“ koje je sponzorirao Google. Otkriven je u studenom 2007. godine, a prvi komercijalni Android uređaj je pušten u prodaju u rujnu 2008. godine.

Android je besplatan i softver otvorenog koda (engl. open source) koji se koristi za razvoj mnogih elektroničkih uređaja kao što su igrače konzole, digitalne kamere, prijenosne medijske uređaje, osobna računala i druge, a svaki sa posebnim sučeljem. Neke od poznatijih su Android TV za televizore i Wear OS za nosive uređaje koje je razvio Google.

2.2 Java programski jezik

Java je jedan od najpopularnijih programskih jezika. Prednost Jave je mogućnost izrade aplikacije koja se može pokrenuti na bilo kojem operacijskom sustavu za računala što je i ideja i slogan zbog koje je Java nastala. WORA (engl. Write Once, Run Anywhere) ili „napiši jednom, pokreni bilo gdje“. Realizaciju ove ideje omogućuje *Java Virtual Machine* (JMV), tj. sustav za pokretanje java aplikacija koji je besplatan i svi ga mogu koristiti. Sintaksa Java programskog jezika je slična C i C++ jezicima, ali ima manje nisko razinskih objekata od oba jezika. Java se koristi u brojnim računalnim platformama od ugradbenih uređaja i mobilnih uređaja do poduzetnih servera i superračunala.

Java platforma je skup programa koji olakšavaju razvoj i rad programa napisanih u Java programskom jeziku. Java platforma uključuje izvršnu virtualnu mašinu (eng. Engine), prevoditelja (eng. Compiler) i set biblioteka, a također mogu biti dodane i serverske ili alternativne biblioteke, zavisno o zahtjevima.

2.3 XML jezik za označavanje podataka

XML (engl. Extensible Markup Language) je jezik koji je zamišljen kao jezik koji naglašava jednostavno i rašireno korištenje na Internetu. Napravljen je s idejom da zapis programa bude čitljiv i ljudima i računalnim programima. Koristi se za prikaz podataka i za povezivanje podataka u Androidu.

Osnovni dioXML dokumenta je element, definiran sa oznakom. Element ima početnu i završnu oznaku. Svi elementi u XML dokumentu su sadržani u elementu koji ih sve obuhvaća koji se zove korijenski element. XML podržava i ugniježđene elemente, tj. elemente u elementima. Ovo svojstvo omogućuje XML da podržava hijerarhijsku strukturu. Ime elementa opisuje sadržaj elementa, a struktura opisuje odnos između elemenata.

XML dokument se smatra dobro oblikovan (takav da ih XML interpreter može pročitati i razumiti) ako se format prevodi unutar XML specifikacija, ako je pravilno označen i ako su elementi pravilno ugnježđeni. XML podržava mogućnost definiranja atributa elemenata i opisivanja karakteristika elemenata na početku oznake elementa.

2.4 Android Studio

Android studio je službeno integrirano razvojno okruženje za Android operacijski sustav koji je napravljen na temelju *IntelliJ IDEA* software-a i dizajniran posebno za razvoj Android aplikacija. Dostupan je na svim računalnim operacijskim sustavima i predstavlja zamjenu za „Eclipse“ Android alat za razvijanje aplikacija koji je bio primarni alat za razvijanje Android aplikacija. U svibnju 2019. godine Kotlin je zamijenio Javu kao preferirani programski jezik za razvijanje Android aplikacija. Java je još uvijek podržana kao i C++ programski jezik.

2.5 Firebase

Firebase je tehnologija koja se temelji na „oblak“ (eng. Cloud) načinu spremanja podataka. Firebase sustav koji ima mnogobrojne mogućnosti kao što su autentifikacija korisnika, baza podataka, hosting, analitika stranice ili aplikacije i mnoge druge. U završnom radu koristi se autentifikacija korisnika i baza podataka. Baza podataka je NOSQL baza podataka te koristi JSON način zapisivanja podataka u bazu. Tvrtka Firebase je sada u vlasništvu Google-a te zato su u Android studiju implementirani koraci kako postaviti aplikaciju da radi i surađuje sa Firebase sustavom.

Firebase je razvojna platforma za mobilne i web aplikacije. Firebase je napravljen od komplementarnih značajki koji programeri mogu kombinirati prema svojim potrebama.

3. Model aplikacije za mjerenja masnoće u hrani

U aplikaciji se, nakon preuzimanja, od korisnika zahtijeva registracija ili prijava s emailom i lozinkom kako bi unio svoje podatke o količini grama masti koje smije dnevno unijeti, kao i količini Omega3 i Omega6 masti, kako bi praćenje njihova unosa bilo lakše.

3.1 Početna stranica/stranica za prijavu

Na početnoj stranici, kao što je prikazano na slici 3.1., moguća je registracija korisnika ako nema postojeći račun za aplikaciju i nakon verifikacije email adrese upisuje se email i šifra te se vrši prijava. Ako je korisnik zaboravio svoju lozinku može ju promijeniti pritiskom na tekst „Zaboravili ste lozinku?“ te nakon pritiska će mu se prikazati stranica sa slike 3.2. Korisnik ne mora svaki put pisati svoj email i lozinku jer ju aplikacija, nakon prve prijave, može zapamtiti. Prilikom pokretanja aplikacije korisniku će biti vidljiv glavni prikaz masti i hrane koju je konzumirao. Za unos informacija korišteni su XML komponente *editText*, *textView* i *button.* Pritiskom na *Prijavi se* gumb podatci se šalju u bazu podataka kako bi se mogli obraditi i koristiti. *Text View* se koristi u slučaju da korisnik nema račun te ga mora napraviti kako bi mogao koristiti aplikaciju.

Slika 3.1. Prikaz Početne stranice aplikacije

Slika 3.2. Prikaz stranice za obnovu lozinke

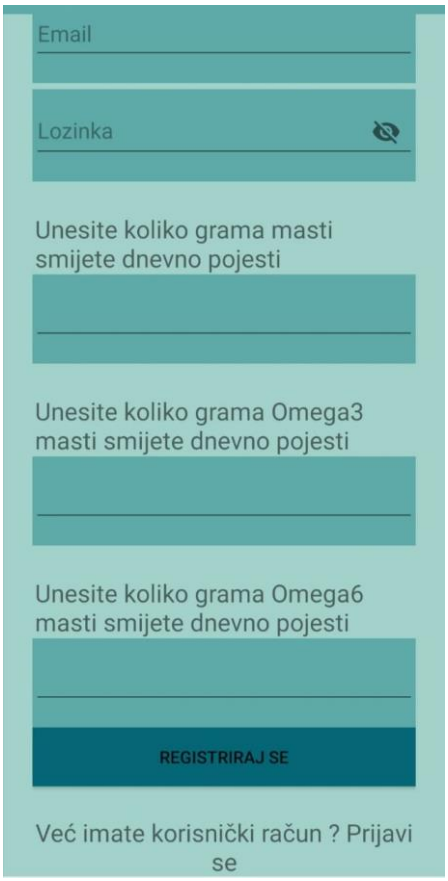
```
userLogin.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        if(!validateEmail() | !validate(textLoginPassword)){
            return;
        }

        firebaseAuth.signInWithEmailAndPassword(userEmail.getText().toString(), userPass.getText().toString())
                .addOnCompleteListener(new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if(task.isSuccessful()){
                    if (firebaseAuth.getCurrentUser().isEmailVerified()){
                        startActivity(new Intent( packageContext: LoginActivity.this, ProfileActivity.class));
                    }else{
                        Toast.makeText( context: LoginActivity.this,  text: "Molimo da verificirate svoju email adresu", Toast.LENGTH_LONG).show();
                    }
                }else{
                    Toast.makeText( context: LoginActivity.this, Objects.requireNonNull(task.getException()).getMessage(), Toast.LENGTH_LONG).show();
                }
            }
        });
    }
});

textView.setOnClickListener((view) -> {
        startActivity(new Intent( packageContext: LoginActivity.this, SignupActivity.class));
});

textViewForgotPassword.setOnClickListener((v) -> {
       startActivity(new Intent( packageContext: LoginActivity.this, ForgotPasswordActivity.class));
});
```

Slika 3.3. Prikaz koda početne stranice za prijavu korisnika

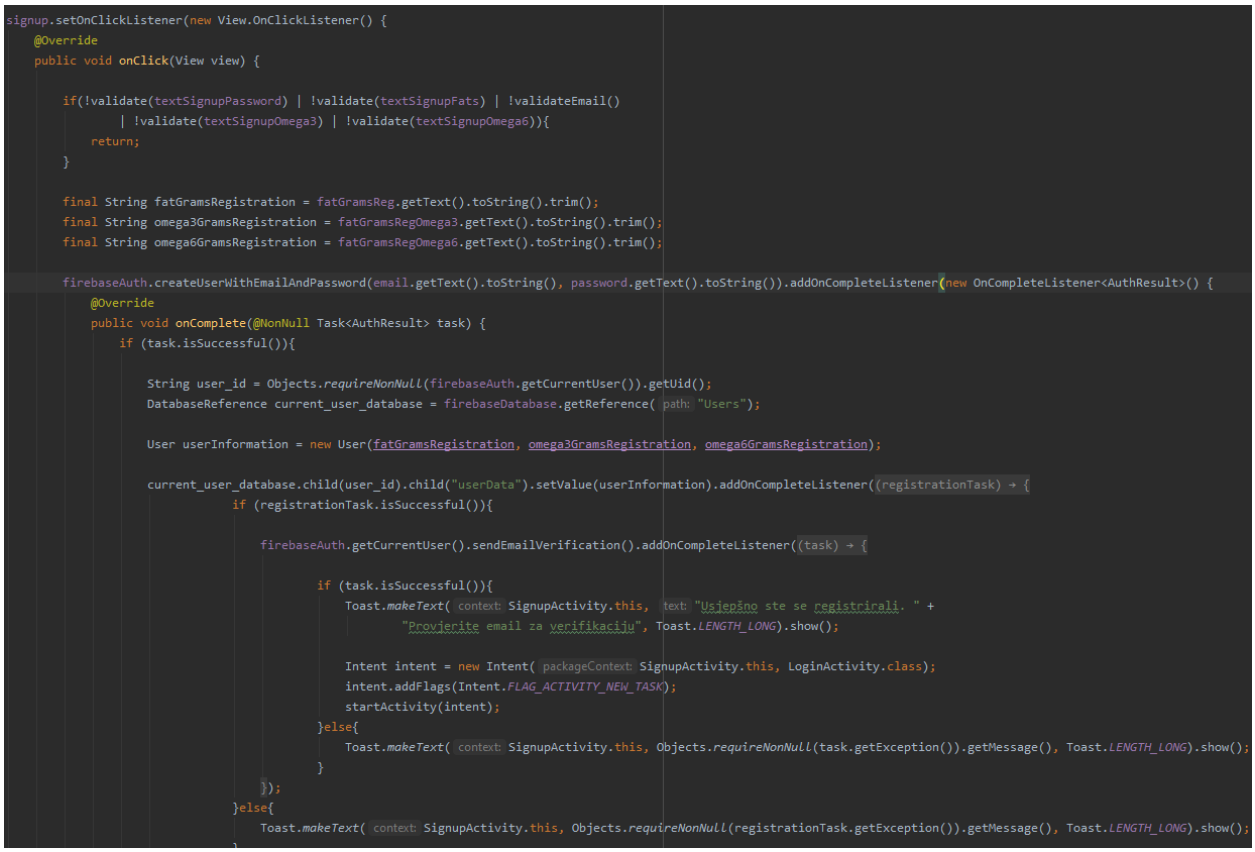
Na slici 3.3. prikazan je kod za prijavu korisnika. Slušatelj (engl. listener) se koristi kako bi korisnici znali kada pritisnuti gumb „Prijavi se“, a nakon što je isti pritisnut pokreće se kod koji se nalazi unutar slušateljske funkcije. Prvo se pozivaju funkcije za validaciju koje provjeravaju ispravnost unosa email adrese, a ako je polje za unos prazno one zaustavljaju tijek rada koda kako bi se unio ispravan email. Dodatno, korisnik se neće moći prijaviti u aplikaciju ako prije toga nije potvrdio svoju email adresu nakon registracije. Potom se koristi ugrađena Firebase funkcija koja prijavljuje korisnika i odvodi ga na glavnu stranicu aplikacije (slika 3.6).

3.2 Stranica za registraciju

Na stranici za registraciju, slika 3.4., korisnik unosi svoje podatke, odnosno email, lozinku i količinu masti, količinu omega 3 i omega 6 masti koje korisnik smije unijeti po danu. Nakon registracije, korisnik mora verificirati svoju email adresu jer u suprotnom neće imati pristup aplikaciji. Za unos informacija korišteni su XML komponente *textview* i *editText i Button.* U *editText* unose se vlastiti podatci potrebni za korištenje aplikacije, a to su email, lozinka, količina masti, količina Omega 3 i Omega 6 masti. Ovi će se podaci koristiti za kasniju prijavu te praćenje i računanje unosa hrane. Pritiskom na *Registriraj se* gumb podatci se šalju u bazu podataka kako bi se oni mogli obraditi i koristiti, a nakon toga će korisnik dobiti email pomoću kojeg mora potvrditi svoju email adresu kako bi mogao koristiti aplikaciju.



Slika 3.4. Prikaz stranice za registraciju



```
signup.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        if(!validate(textSignupPassword) | !validate(textSignupFats) | !validateEmail()
                | !validate(textSignupOmega3) | !validate(textSignupOmega6)){
            return;
        }

        final String fatGramsRegistration = fatGramsReg.getText().toString().trim();
        final String omega3GramsRegistration = fatGramsRegOmega3.getText().toString().trim();
        final String omega6GramsRegistration = fatGramsRegOmega6.getText().toString().trim();

        firebaseAuth.createUserWithEmailAndPassword(email.getText().toString(), password.getText().toString()).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (task.isSuccessful()){

                    String user_id = Objects.requireNonNull(firebaseAuth.getCurrentUser()).getUid();
                    DatabaseReference current_user_database = firebaseDatabase.getReference( path: "Users");

                    User userInformation = new User(fatGramsRegistration, omega3GramsRegistration, omega6GramsRegistration);

                    current_user_database.child(user_id).child("userData").setValue(userInformation).addOnCompleteListener((registrationTask) -> {
                                if (registrationTask.isSuccessful()){

                                    firebaseAuth.getCurrentUser().sendEmailVerification().addOnCompleteListener((task) -> {

                                            if (task.isSuccessful()){
                                                Toast.makeText( context: SignupActivity.this,  text: "Uspješno ste se registrirali. " +
                                                        "Provjerite email za verifikaciju", Toast.LENGTH_LONG).show();

                                                Intent intent = new Intent( packageContext: SignupActivity.this, LoginActivity.class);
                                                intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                                                startActivity(intent);
                                            }else{
                                                Toast.makeText( context: SignupActivity.this, Objects.requireNonNull(task.getException()).getMessage(), Toast.LENGTH_LONG).show();
                                            }
                                        });
                                }else{
                                    Toast.makeText( context: SignupActivity.this, Objects.requireNonNull(registrationTask.getException()).getMessage(), Toast.LENGTH_LONG).show();
```

Slika 3.5. Prikaz koda za registraciju korisnika

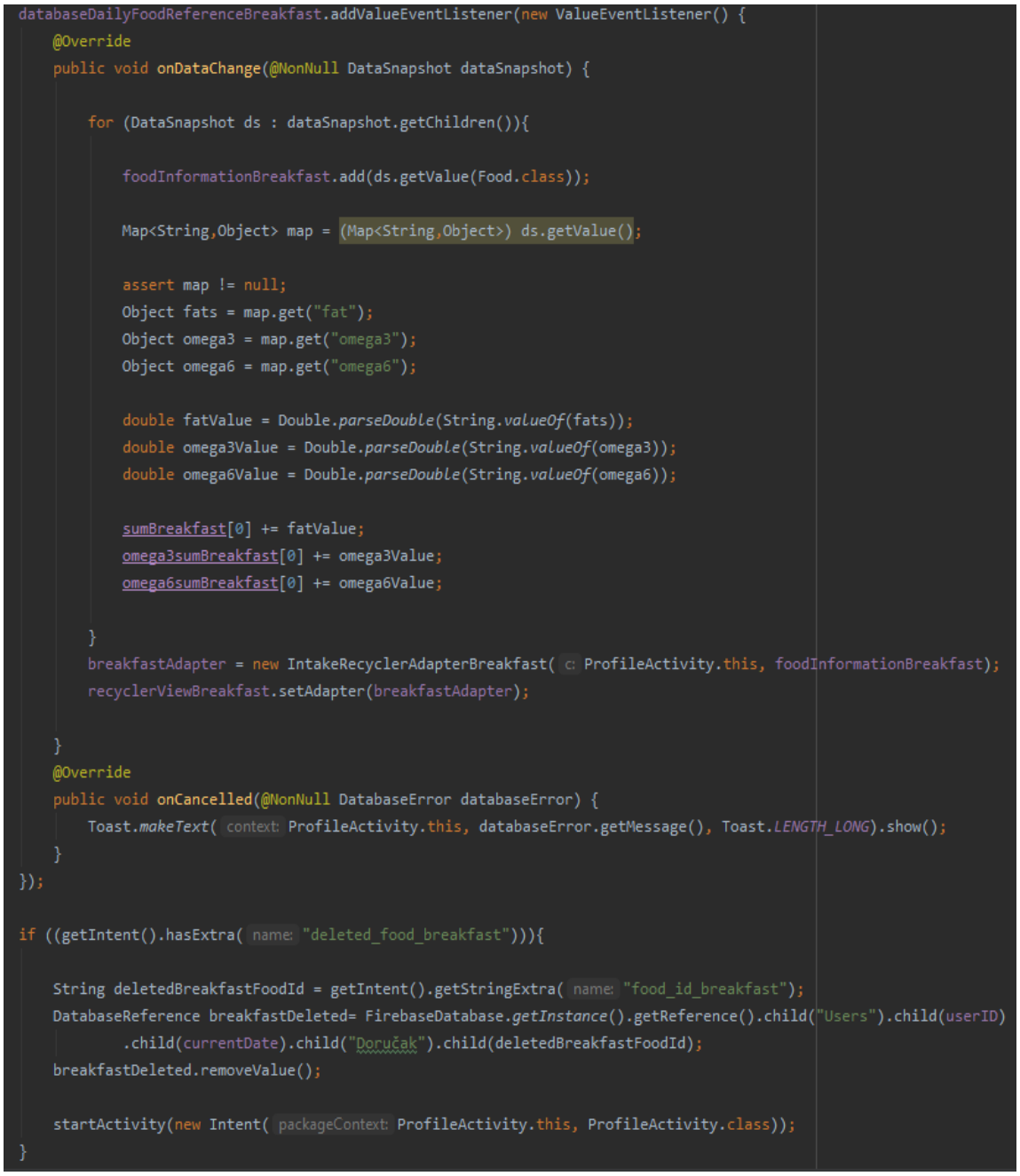
Na slici 3.5. prikazan je kod za registraciju korisnika. Kao i kod prijave korisnika, prvo se pozivaju funkcije koje provjeravaju ispravnost oblika unesenih podataka, kao što su email i lozinka te provjeravaju jesu li polja za unos prazna. Nakon toga će program unesene podatke pretvoriti u *String* varijable. Potom se koristi ugrađena Firebase funkcija za stvaranje novih korisnika uz pomoć email adrese i lozinke u kojoj se predaju uneseni podatci. Nakon toga se stvara novi objekt korisničke (engl. User) klase u koji se spremaju upisani podatci. Zatim se koriste ugrađene Firebase funkcije za spremanje podataka u bazu podataka te funkcija koja će poslati link za potvrdu korisničke email adrese.

3.3 Stranica za prikaz unosa kalorija i masti

Na stranici za prikaz unosa kalorija i masti, slika 3.6., vidljiv je ukupan izračun vrijednosti masti, omega 3 i omega 6 masti koje korisnik smije konzumirati za taj dan te količina koja je već do sada konzumirana i koliko je količinski masti preostalo za unijeti taj dan. Ispod toga vidljiv je gumb za dodavanje hrane koju korisnik planira konzumirati, a ispod se nalazi gumb za zaključivanje dana tj. sva se unesena hrana sprema, po mogućnosti na kraju dana. Ispod toga prikazana je vrsta i količina hrane koju je korisnik konzumirao taj dan, slika 3.7. Uz svaku namirnicu koju je korisnik konzumirao može se vidjeti i količina masti u toj porciji, kalorije koje sadrži ta namirnica s obzirom na veličinu porcije, količinu omega 3 i omega 6 masti u toj porciji te ikone pomoću kojih je moguće urediti unesenu namirnicu ili ju obrisati iz zapisa unosa hrane tog dana.

Slika 3.6. Prikaz glavne stranice aplikacije

Slika 3.7. Prikaz glavne stranice aplikacije, unesene hrane iz baze podataka



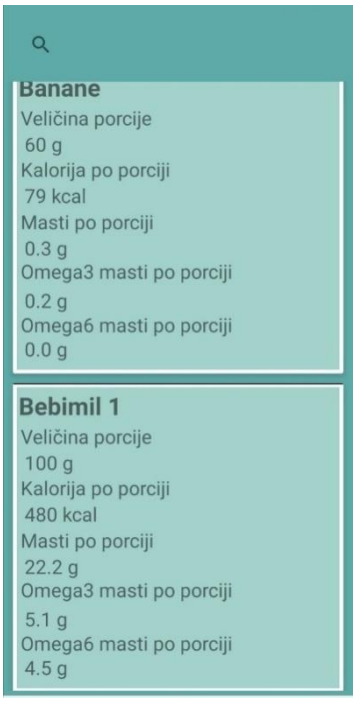
```
databaseDailyFoodReferenceBreakfast.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

        for (DataSnapshot ds : dataSnapshot.getChildren()){

            foodInformationBreakfast.add(ds.getValue(Food.class));

            Map<String,Object> map = (Map<String,Object>) ds.getValue();

            assert map != null;
            Object fats = map.get("fat");
            Object omega3 = map.get("omega3");
            Object omega6 = map.get("omega6");

            double fatValue = Double.parseDouble(String.valueOf(fats));
            double omega3Value = Double.parseDouble(String.valueOf(omega3));
            double omega6Value = Double.parseDouble(String.valueOf(omega6));

            sumBreakfast[0] += fatValue;
            omega3sumBreakfast[0] += omega3Value;
            omega6sumBreakfast[0] += omega6Value;

        }
        breakfastAdapter = new IntakeRecyclerAdapterBreakfast( c: ProfileActivity.this, foodInformationBreakfast);
        recyclerViewBreakfast.setAdapter(breakfastAdapter);

    }
    @Override
    public void onCancelled(@NonNull DatabaseError databaseError) {
        Toast.makeText( context: ProfileActivity.this, databaseError.getMessage(), Toast.LENGTH_LONG).show();
    }
});

if ((getIntent().hasExtra( name: "deleted_food_breakfast"))){

    String deletedBreakfastFoodId = getIntent().getStringExtra( name: "food_id_breakfast");
    DatabaseReference breakfastDeleted= FirebaseDatabase.getInstance().getReference().child("Users").child(userID)
            .child(currentDate).child("Doručak").child(deletedBreakfastFoodId);
    breakfastDeleted.removeValue();

    startActivity(new Intent( packageContext: ProfileActivity.this, ProfileActivity.class));
}
```

Slika 3.8. Prikaz koda za dohvaćanje podataka o unešenoj hrani iz baze podataka

Na slici 3.8. prikazan je dio koda koji se koristi za funkcioniranje glavne stranice aplikacije. Pomoću ugrađene Firebase funkcije „dodaj slušatelja za promjenu podataka“ (eng. Add Value Event Listener) koja šalje zahtjev za dohvaćanje podataka kada se neki od podataka promijeni ili kada se pokrene klasa u kojoj se nalazi funkcija. Nakon toga se podatci spremaju u mapu kako bi se mogli uzeti specifični podatci potrebni za prikazivanje zbroja masti, omega 3 i omega 6 masti, kroz dan. Zatim se postavlja adapter koji pomaže prikazati podatke kao što su prikazani na slici 3.8.

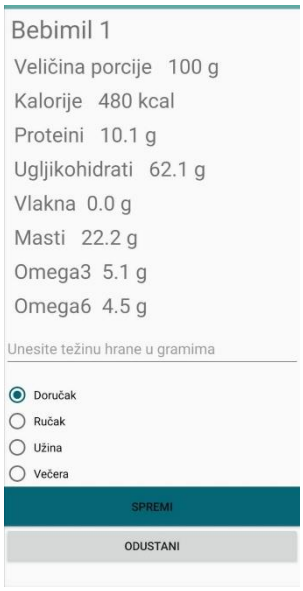
3.4 Stranica za prikaz svih namirnica u bazi podataka

Pritiskom na gumb „Dodaj hranu“, aplikacija dovodi korisnika na prikaz svih namirnica u bazi podataka (slika 3.9.). Baza se može pretraživati unutar aplikacije pomoću tražilice, koja je prikazana na slici. Nakon što se pronađe željena namirnica za dodavanje u dnevni unos, korisnik treba pritisnuti karticu s hranom koju želi dodati.



Slika 3.9. Prikaz stranice sa svim namjernicama u bazi

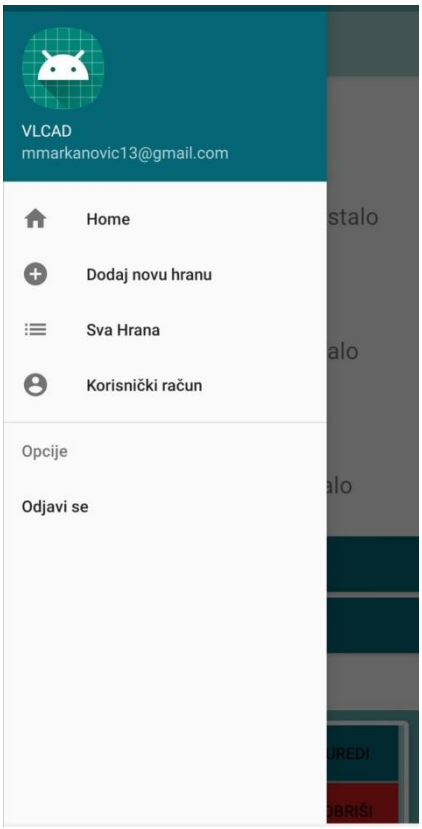
Pritiskom na karticu s hranom aplikacija će prikazati detalje hrane, slika 3.10., koju je korisnik odabrao te mora unijeti težinu hrane koju želi konzumirati u gramima. Ta će se namirnica prikazati na stranici za prikaz unosa masti, tj. na početnoj stranici. Korisnik ne mora preračunavati težinu hrane nego samo mora unijeti količinu hrane u gramima koju namjerava konzumirati jer je program napravljen tako da podijeli unesenu količinu hrane i veličinu porcije te s izračunatim faktorom množi sve ostale parametre hrane i sprema taj unos u bazu podataka. Npr. ako se uzme hrana odabrana na slici 3.7. i unese podatak da je konzumirano 200 grama te hrane, program će podijeliti 200 grama unesene hrane s veličinom porcije od 100 grama te će se dobiti faktor 2. Pomoću faktora će se potom pomnožiti svi ostali parametri u zapisu hrane i spremiti u bazu podataka.



Slika 3.10. Prikaz stranice za unos pojedinačne namirnice

3.5 Navigacijski izbornik

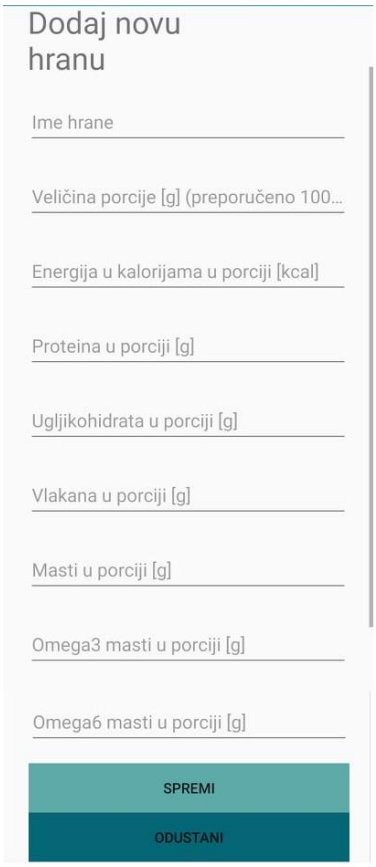
U navigacijskom izborniku, slika 3.11., koji se prikazuje pritiskom na ikonu za *hamburger menu*, može se vidjeti korisnikov email, kao i navigacijski gumbi pomoću kojih se može doći do stranice za prikaz unosa kalorija i masti (gumb „Home“), pregledati i pretražiti sve namirnice u bazi podataka (gumb „Sva hrana“), dodati novu namirnicu i unijeti njezino ime i sve podatke potrebne za korištenje aplikacije (gumb „Dodaj novu hranu“) te se mogu pregledati vlastiti podatci, a isti se mogu ažurirati pod gumbom „Korisnički račun“. Odjava iz aplikacije vrši se pritiskom na gumb „Odjavi se“.



Slika 3.11. Prikaz glavnog izbornika aplikacije

3.6 Stranica za dodavanje nove namirnice

Na stranici za dodavanje nove namirnice, slika 3.12, nova se namirnica može unijeti tako da se unese njeno ime i osnovne nutritivne vrijednosti. Pritiskom na gumb „Spremi“, nova se namirnica šalje bazi podataka kako bi ju svaki korisnik mogao koristiti.



Slika 3.12. Prikaz stranice za unos nove namirnice

```
DatabaseReference new_food_database = firebaseDatabase.getReference( path: "Food");



String foodIDs = foodName.getText().toString();
String foodNames = foodName.getText().toString();
String portionSizes = portionSize.getText().toString();
String energies = energy.getText().toString();
String proteins = protein.getText().toString();
String fats = fat.getText().toString();
String carbohydrates = carbohydrate.getText().toString();
String fibers = fiber.getText().toString();
String omega3s = omega3.getText().toString();
String omega6s = omega6.getText().toString();

Food foodInformation = new Food(foodNames ,portionSizes, energies,
        proteins, fats, carbohydrates, fibers, omega3s, omega6s);

new_food_database.child(foodIDs).setValue(foodInformation).addOnCompleteListener((task) -> {
        if(task.isSuccessful()){
            Toast.makeText( context: AddNewFoodActivity.this,  text: "New Food Created", Toast.LENGTH_LONG).show();

            startActivity(new Intent( packageContext: AddNewFoodActivity.this, ProfileActivity.class));
        }else{
            Toast.makeText( context: AddNewFoodActivity.this, Objects.requireNonNull(task.getException()).getMessage(), Toast.LENGTH_LONG).show();
        }
});
```

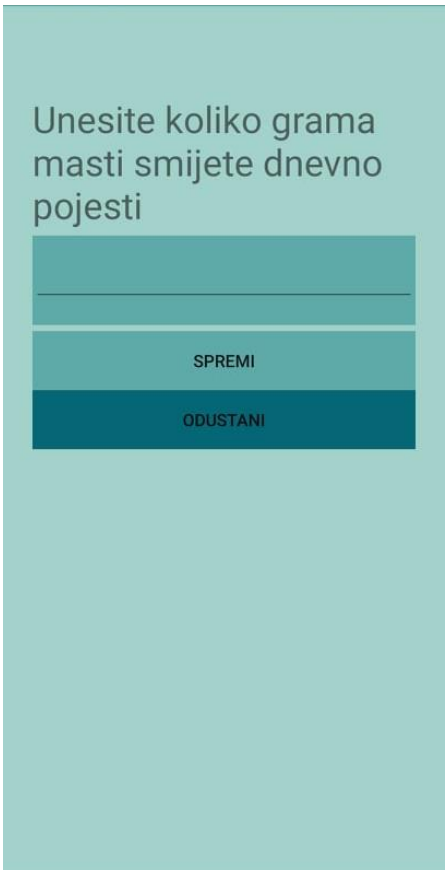
Slika 3.13. Prikaz koda za unos nove namirnice

Na slici 3.13. vidljiv je kod za unos nove namirnice. Program preuzima podatke iz svih *editText* elemenata te se oni spremaju u *String* varijable kako bi se mogli spremiti u bazu podataka. Nakon toga se stvara novi objekt klase „Hrana“ (engl. Food) te se u taj objekt predaju svi podatci koje je korisnik unio. Nakon toga se poziva funkcija za spremanje podataka u bazu podataka pomoću koje se stvara novi čvor u bazi podataka koji će imati oznaku imena hrane koju je korisnik unio.

3.7 Stranica za prikaz osobnih podataka

Na stranici za prikaz osobnih podataka, slika 3.14., prikazani su svi podatci koji su uneseni prilikom registracije korisnika. Na dnu stranice se, pod naslovom „Izvješće“, prikazuje dio u kojem se vrši odabir vremenskog razdoblja te se može preuzeti povijest unosa hrane u tom vremenskom razdoblju. Dokument povijesti unosa hrane se generira pritiskom na „Spremi“ gumb te se sprema na unutarnju memoriju uređaja. Pritiskom na „Ažuriraj broj masti“ podatak o dnevnom unosu masti, omega 3 ili omega 6 masti, slika 3.15., moguće je i ažurirati ovisno o napretku korisnika ili nekakvoj promjeni.

Slika 3.14. Prikaz stranice o podatcima korisnika



Slika 3.15. Prikaz stranice za ažuriranje korisničkih podataka

```
try {
    document.add(new Paragraph( string: "\n"));
    PdfPTable table = new PdfPTable( numColumns: 1);

    PdfPCell cellFoodId = new PdfPCell(Phrase.getInstance("Ime hrane : " + foodIdValue));
    table.addCell(cellFoodId);

    PdfPCell cellPortionSize = new PdfPCell(Phrase.getInstance("Velicina porcije : " + portionSizeValue + " g"));
    table.addCell(cellPortionSize);

    PdfPCell cellCalories = new PdfPCell(Phrase.getInstance("Kalorije : " + caloriesValue + " g"));
    table.addCell(cellCalories);

    PdfPCell cellProtein = new PdfPCell(Phrase.getInstance("Proteini : " + proteinValue + " g"));
    table.addCell(cellProtein);

    PdfPCell cellCarbohydrates = new PdfPCell(Phrase.getInstance("Ugljikohidrati : " + carbohydratesValue + " g"));
    table.addCell(cellCarbohydrates);

    PdfPCell cellFibers = new PdfPCell(Phrase.getInstance("Vlakna : " + fibersValue + " g"));
    table.addCell(cellFibers);

    PdfPCell cellFats = new PdfPCell(Phrase.getInstance("Masti : " + fatsValue + " g"));
    table.addCell(cellFats);

    PdfPCell cellOmega3 = new PdfPCell(Phrase.getInstance("Omega 3 masti : " + omega3Value + " g"));
    table.addCell(cellOmega3);

    PdfPCell cellOmega6 = new PdfPCell(Phrase.getInstance("Omega 6 masti : " + omega6Value + " g"));
    table.addCell(cellOmega6);

    document.add(table);
```

Slika 3.16. Prikaz koda za stvaranje tablice u PDF dokumentu

Na slici 3.16. vidljiv je kod za kreiranje tablice u PDF dokumentu koji se generira pritiskom na gumb „Spremi“. Za generiranje PDF dokumenta se koristi „iText“ biblioteka koja omogućava dinamično i lakše generiranje PDF dokumenata pomoću Android aplikacija. Kada program dođe do ovog dijela koda, prvi je korak stvaranje novog objekt tablice. Za svaki se podatak stvara nova ćelija u koju se unosi podatak i bilo koji dodatni tekst kako bi podatci bili lakše razumljivi. Na dnu slike istaknuta je funkcija za dodavanje tablice u dokument. Važno je istaknuti da je za postupak pisanja podataka u generirani PDF dokument potrebno pozvati funkciju za otvaranje dokumenta te, na kraju pisanja, zatvoriti dokument. Kod pozivanja funkcije za otvaranje dokumenta, otvorit će se početak dokumenta i podatci će se početi pisati od početka. Funkcija za zatvaranje dokumenta može se pozvati samo jednom te se ona poziva na kraju. U suprotnom će program izbaciti grešku prema kojoj korisnik pokušava pisati u dokumentu koji je zatvoren.

4. Baza podataka

Baza podataka je NOSQL, što znači da se ne koriste relacijski modeli za kreiranje baze podataka kao kod SQL baza podataka. Baza se kreira na temelju napisanog koda, slika 4.1., na način na koji je poželjno da se podatci spremaju.

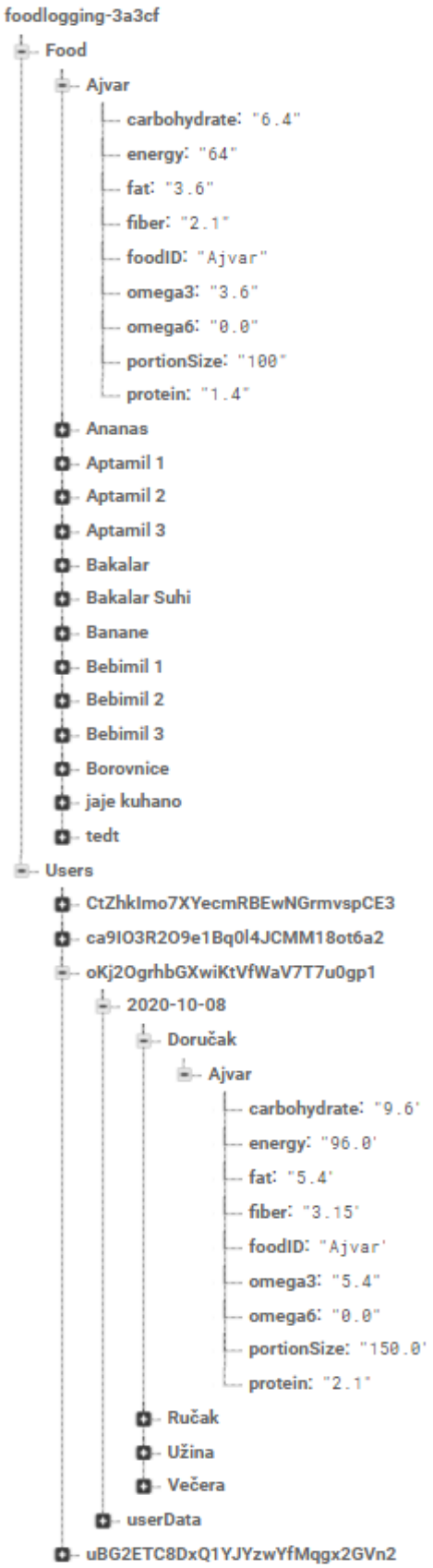
```
dailyFoodIntake.child(currentDate).child(intakeFoodId).setValue(foodInformationDaily).addOnCompleteListener(new OnCompleteListener<Void>() {
    @Override
    public void onComplete(@NonNull Task<Void> task) {
        if (task.isSuccessful()){
            Toast.makeText( context: AddFoodToIntakeActivity.this,  text: "New Food Created", Toast.LENGTH_LONG).show();

            startActivity(new Intent( packageContext: AddFoodToIntakeActivity.this, ProfileActivity.class));

        }else {

            Toast.makeText( context: AddFoodToIntakeActivity.this, task.getException().getMessage(), Toast.LENGTH_LONG).show();
        }
    }
});
```

Slika 4.1. Prikaz koda za unos podataka u bazu podataka

Kao što se vidi na slici 4.2., baza podataka je podijeljena na dva glavna dijela. Prvi je dio Hrana (eng. Food), u kojem se nalaze osnovni podatci o nekoj hrani kao što su njezino ime, veličina porcije, količina masti u toj porciji te kalorije, ugljikohidrati i proteini u toj porciji.

Korisnici (eng. Users) čine drugi dio baze, u koji se spremaju svi podatci koji su vezani za konkretnog korisnika. Pod datumom se spremaju svi unosi hrane koju je korisnik dodao za taj dan. Spremaju se pod 4 sekcije: doručak, ručak, užina i večera. Pod sekcijom userData se spremaju korisnički podatci kao što su dozvoljena dnevna granica unosa masti i sažetak jednog dana unosa hrane kako bi se podatci mogli poslati korisniku na kraju mjeseca.

Pristup podatcima u programu ostvaruje se na način da se programu preda putanja potrebnih podatka te se nakon toga pozivaju potrebne funkcije za spremanje, uređivanje ili brisanje podataka.



Slika 4.2. Prikaz strukture Firebase baze podatak

5. Zaključak

Brzim napredovanjem današnjih tehnologija poboljšava se i olakšava način života. Stvari koju su donedavno bile ili prenaporne ili čak nezamislive, postale su moguće uz razvitak tehnologije te su danas svakodnevnica.

Razvitkom ove aplikacije olakšan je život osobama s nedostatkom VLCAD-a, na način da mogu koristiti ovu automatiziranu aplikaciju za računanje dnevnog unosa hrane te tako mogu izbjeći prekoračenje dozvoljene dnevne granice. Time im je olakšan svakodnevni život te se manje opterećuju glede konzumacije i pripreme hrane. Važno je da korisnici unose hranu u aplikaciju prije obroka kako bi znali hoće li prekoračiti dozvoljenu granicu te sukladno tome korigirati unos i jelo.

Aplikacija je izrađena u Android programskom sučelju koja koristi Java i XML programskih jezika zbog jednostavnosti pristupa i dostupnosti onima kojima je potrebna. Uz dizajniranje izgleda i kodiranja funkcionalnosti aplikacije, važan dio aplikacije predstavlja i baza podataka koja je izrađena pomoći Google Firebase platforme u kojoj se nalaze svi korisnički podatci, unosi i informacije o svim namirnicama. Na ovaj način olakšan je pristup i spremanje podataka potrebnih za rad aplikacije. Za razvoj aplikacije korištena su znanja stečena na kolegijima Dizajn korisničkog sučelja, Baze podataka, Razvoj mobilnih aplikacija i Dizajn u objektno orijentiranom programiranju.

Iskustva stečena tijekom izrade aplikacije izuzeteno su korisna i mogu se koristiti za izradu složenijih projekata. Za daljnja poboljšanja autor predlaže implementaciju opcije korištenja kamere i skeniranja barkodova u aplikaciju, za lakši unos namirnica i poboljšanje korisničkog sučelja za još jednostavnije korištenje.

6. Literatura

[1] Android, Android Language Breakdown , 9.6.2017.

[2] Java, Oracle Java SE Support Roadmap. Oracle Corporation. 2018-09-25.

[3] Firebase, *Firebase - CrunchBase. CrunchBase*. 11.6.2014.

[4] Leslie, N.D. i sur. 2019. Very Long-Chain Acyl-Coenzyme A Dehydrogenase Deficiency, GeneReviews

[5] Android Studio, Android Studio release notes. Android Developers. 16.6.2020.

[6] Intent, https://developer.android.com/reference/android/content/Intent.html, 5.7.2017

[7],,Acute illness protocol fatty acid oxidation disorders very long chain acyl coa dehydrogenase (vlcadd) deficiency“, https://newenglandconsortium.org/protocols/acute_illness/fatty-acidoxidation-disorders/VLCADD.pdf, pristup ostvaren 28.6.2019

SAŽETAK

Naslov: Aplikacija za Android platformu za mjerenja masnoće u hrani

Cilj rada je izraditi Android aplikaciju koja će pomoći osobama sa VLCAD poremećajem tako da im olakša izračun dnevnog unosa, masti, kalorija, ugljikohidrata i proteina. Na početku rada je opisan VLCAD poremećaj, simptomi i posljedice koje taj poremećaj donosi oboljelim osobama. Zatim su ukratko opisane tehnologije koje su se koristile za realiziranje ove aplikacije. Nadalje, opisan je način kako se koristi aplikacija i njene mogućnosti. Potom je opisana baza podataka, kako se izrađuje i kako se stvara hijerarhija u bazi podataka.

Ključne riječi: Android, Java, Firebase, VLCAD, pohrana podataka.

ABSTRACT

Title : Android application for calculating and measuring intake of fats in food

The goal of this project was to developan Android application that would help people with VLCAD disorder, in the way, that would allow them to easily follow the daily intake of calories, fats, carbohydrates and protein. That being said, at the beginning of the paper, VLCAD disorder is described with all symptoms and consequences that this disorder brings to affected people. After that, the paper more thoroughly explains the technologies used in the development of the application. In addition, the exact example of how to use the application was defined, with all additional features. Lastly, the data base is presented, with the precise process of creation and hierarchy within itself.

Keywords : Android, Java, Firebase, VLCAD, data storage

ŽIVOTOPIS

Marko Markanović rođen je 13. listopada 1996. godine u Požegi. Osnovnoškolsko obrazovanje započinje 2003. godine u OŠ Dobriše Cesarić u Požegi. Nakon toga upisuje Gimnaziju u Požegi, prirodoslovno-matematički smjer. Tijekom školovanja sudjeluje u brojnim predmetnim aktivnostima te se aktivno bavi glazbom. Akademske godine 2015/2016 upisuje Stručni sveučilišni studiji elektrotehnike, smjer Informatika na Fakultetu elektronike, računarstva i informacijskih tehnologije u Osijeku. Tijekom studiranja radi kao student u tvrtci „Samurai Digital“ na poziciji „Service Delivry Manager“.