Android aplikacija za praćenje dnevnog treninga

Ivanović, Denis

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

Stručni studij

Android aplikacija za praćenje dnevnog treninga

Završni rad

Denis Ivanović

Osijek, 2019.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1S: Obrazac za imenovanje Povjerenstva za obranu završnog rada na preddiplomskom stručnom studiju

Osijek, 20.09.2019.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu završnog rada na preddiplomskom stručnom studiju

Ime i prezime studenta:	Denis Ivanović
Studij, smjer:	Preddiplomski stručni studij Elektrotehnika, smjer Informatika
Mat. br. studenta, godina upisa:	AI4548, 19.09.2018.
OIB studenta:	21223954683
Mentor:	Robert Šojo
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Doc.dr.sc. Josip Balen
Član Povjerenstva:	Krešimir Vdovjak
Naslov završnog rada:	Android aplikacija za praćenje dnevnog treninga
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak završnog rada	Kreirati android aplikaciju pomoću koje se može pratiti napredak dnevnog treninga. Aplikacija treba omogućiti unos informacije o korisniku aplikacije, kao što su godine, visina, težina, broja tjednog treninga, odabir vježbi po treningu, broj serija, broj ponavljanja, opterećenje, pauze između serija i vježbi, broj sati odmora, unos hrane s ispisom nutritivnih vrijednosti. Opisati tehnologije u procesu izrade aplikacije, opisati pojedine dijelove aplikacije, te kreirati potpuno funkcionalnu aplikaciju.
Prijedlog ocjene pismenog dijela ispita (završnog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	20.09.2019.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

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

IZJAVA O ORIGINALNOSTI RADA

Osijek, 26.09.2019.

Ime i prezime studenta:	Denis Ivanović
Studij:	Preddiplomski stručni studij Elektrotehnika, smjer Informatika
Mat. br. studenta, godina upisa:	AI4548, 19.09.2018.
Ephorus podudaranje [%]:	10

Ovom izjavom izjavljujem da je rad pod nazivom**: Android aplikacija za praćenje dnevnog treninga**

izrađen pod vodstvom mentora Robert Šojo

i sumentora

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

Potpis studenta:

Sadržaj

1. UVOD

Ovaj završni rad je namijenjen kao što mu i sam naziv kaže za praćenje dnevnog treninga. Trenutno postoji dosta stranih aplikacija na ovu temu, ali su one samo jednim malim djelom besplatne. Odnosno ukoliko želimo koristiti sve dijelove aplikacije moramo platiti određenu naknadu. Iz tog razloga bi razvoj ovakve aplikacije bio zanimljiv velikom broju ljudi. Aplikacija treba omogućiti unos informacija o korisniku, kao što su visina, masa te ostala fizička mjerenja koja mogu biti pokazatelj promjena na tijelu. Također aplikacija omogućuje praćenje unosa prehrane na dnevnoj bazi. Odnosno računa unos nutritivnih vrijednosti u tijelo na osnovu prethodno definiranih i unesenih namirnica te količine hrane koju smo unijeli. Omogućuje unos odrađenih vježbi za svaki dan, njihov broj ponavljanja, opterećenje koje je korišteno i vrijeme odmora između serije. Aplikacija je zamišljena da se koristi svakodnevno i da korisnik na dnevnoj bazi unosi podatke vezane uz njegovu aktivnost i prehranu te da na neki način potakne korisnika na zdravi život i više brige o kvaliteti i kvantiteti hrane koju unosi u svoj organizam. Pretilost je jedan od najvećih problema današnjice i sve više ljudi se suočava s njim. Velik broj ljudi koji imaju problema sa viškom kilograma krivca uvijek traže negdje drugdje, umjesto da krenu od sebe i da počnu voditi računa o svom zdravlju. Korištenje ove aplikacije može imati velikog utjecaja na životne navike ljudi, ova aplikacije im neće govoriti što da jedu ili kada da vježbaju jer nije takvog sadržaja, ali će se njenim redovnim korištenjem u podsvijesti stvoriti navika za vježbanjem i kontroliranim unosom hrane u organizam.

U drugom poglavlju se kroz teorijski dio objašnjavaju tehnologije vezane za proces izrade aplikacije. Opisuje se Android operacijski sustav, njegova povijest, te neke osnovne karakteristike. Dalje kroz ovo poglavlje opisuju se glavne komponente razvoja mobilne aplikacije, a to je Android Studio, te dva glavna programska i opisna jezika koja su korištena Java i XML (engl. *Extensible Markup Language*).

U trećem poglavlju prikazuje se čitav postupak izrade Android mobilne aplikacije. Od konceptualnog modela koji ugrubo prikazuje kako aplikacija radi, preko oglednog dizajna koji predstavlja zamišljeni izgled pojedinih zaslona aplikacije, do same realizacije čitavog projekta kroz njegovu implementaciju.

U četvrtom poglavlju prikazuje se način korištenja aplikacije i korisnički zahtjevi koje aplikacija mora ispuniti. Detaljno se objašnjava kako i na koji način se upravlja i služi pojedinim

djelom aplikacije. Također objašnjavaju se glavni mogući slučajevi korištenja aplikacije od strane korisnika.

1.1. Zadatak završnog rada

Izrada mobilne aplikacije koja omogućuje dnevno praćenje unosa prehrane kao i odrađenih vježbi. Aplikacija treba omogućiti unos informacija o korisniku aplikacije. Te imati mogućnost unosa fizičkih mjerenja kako bi korisnik mogao pratiti napredak tijekom treninga.

2. OPIS KORIŠTENIH TEHNOLOGIJA

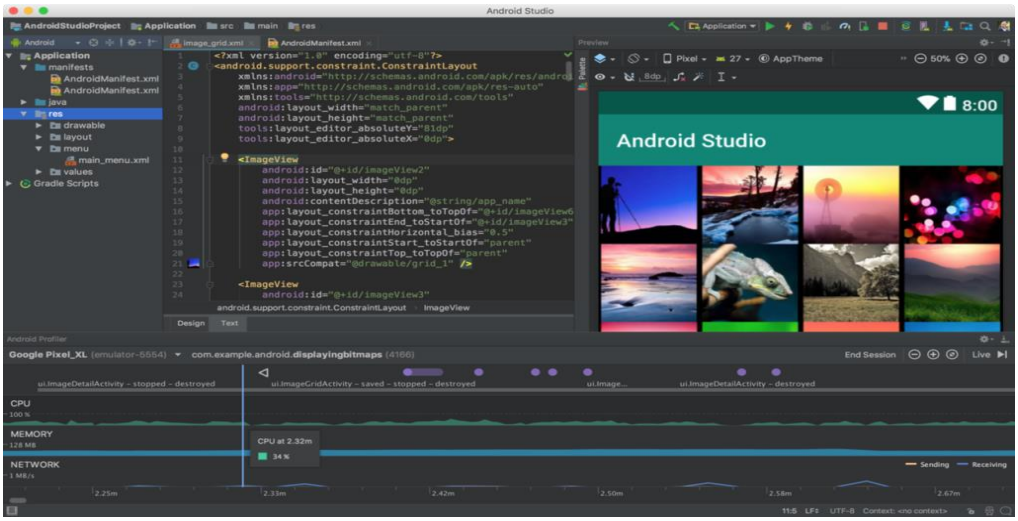
U ovom poglavlju su opisane tehnologije koje su korištene prilikom izrade aplikacije. Neki osnovni podatci vezani za Android operacijski sustav, te njegova povijest. Java i XML jezici njihova kratka povijest te za što koji služi.

2.1. Android

Android je operacijski sustav namijenjen mobilnim uređajima, kao što su mobilni telefoni i tableti. Android OS razvila je tvrtka Android Inc. koju je kupio Google. Andorid Inc. je osnovana u listopadu 2003. godine u mjestu Palo Atlo u Kalifoniji od strane Andy Rubin, Rich Miner, Nick Sears i Chris White. Iz početka je Android bio zamišljen da bude operacijski sustav za digitalne kamere, ali se onda sredinom 2004. godine zbog sve većeg interesiranja investitora prebacuje na mobilne uređaje. Android pripada Unix grupi operacijskih sustava te ima licencu otvorenog kôda (engl. *open-source license*) što znači da je javno dostupan svima na korištenje, dizajn i razvoj. Iako se Android primarno programira u Javi, ne postoji JDM (engl. *Java Development Machine*) u Android platformi. Umjesto da su dozvolili da se Java programski kôd pokreće preko JDM, Google je razvio Dalvik. Dalvik je virtualna mašina specijalizirana samo za Android. Dalvik pokreće izmijenjeni Java programski kôd i pokreće ga kao Dalvik *bytecode*. On je osmislit kako bi optimizirao rad telefona smanjio potrošnju baterije telefona, te kako bi radio bez većih poteškoća u okruženju gdje su ograničeni resursi. Najveća prednost Androida je ta što ima licencu otvorenog kôda te je dostupan svima, to povećava broj ljudi i okruženje u kojem se on razvija te se samim time drastično ubrzava njegov razvoj. Sam Android i aplikacije za Android se razvijaju u programu koji se zove Android Studio. Prilikom razvoja Android aplikacije u Android Studiju podržani su razni programski jezici od kojih su glavni: Java, Kotlin, C/C++, BASIC i C#. Ali ipak tu u samom vrhu najviše treba izdvojiti Javu i Kotlin koji se u najvećoj mjeri koriste. To je i usko povezano sa činjenicom da Android koristi Dalvik virtualni stroj koji je baziran na JVM (engl. *Java Virtual Machine*) Korištenje bilo kojeg Android uređaja je veoma jednostavno i svodi se na osnovne direktne manipulacije na ekranima kao što su stiskanja, povlačenja, pisanja itd. Sve aplikacije za Android se nalaze u Google trgovini (engl. *Google Play Store*). Danas Android zauzima 76% ukupnog tržišta pametnih telefona u cijelom svijetu. [2]

2.2. Android studio

Android studio je jedino službeno razvojno okruženje namijenjeno razvoju Android aplikacija. Sastoji se od mnogobrojnih alata koji su potrebni za razvoj same aplikacije, kao što su alati za dizajn korisničkog sučelja, alati za pisanje programskog kôda, alati za testiranje, alati za komunikaciju sa mobilnim uređajem, alati za simuliranje mobilnih uređaja na samom računalu itd. Izgled programskog okruženja Android Studija prikazan je slikom 2.1. Zasniva se na IntelliJ IDEA, razvojnom okruženju bazirano na Javi. Program Android Studio je pisan u Javi, Kotlinu i C++, te zauzima od 971 do 1036MB prostora na disku. Minimalni i preporučeni zahtjevi za instalaciju Android Studija su prikazani prema tablici Tab. 2.1. Android studio je prvi put objavljen 16.05. 2013. godine na Google I/O konferenciji, također je dostupan na Mac, Windows i Linux platformama.



Sl. 2.1. *Izgled Android studio programskog okruženja.*

Windows
Microsoft® Windows® 7/8/10 (32- ili 64-bit) Android Emulator je podržan samo na 64-bit Windowsima
4GB RAM minimalno, 8GB RAM preporučeno
2GB slobodnog prostora na disku minimalno, 4GB prostora na disku preporučeno
1280 x 800 minimalna rezolucija ekrana

Tab. 2.1. *Minimalni i preporučeni sistemski zahtjevi za instalaciju Android Studia na Windows operacijskim sistemima.*

2.3. Java programski jezik

Java programski jezik je objektno orijentiran, razvili su ga James Gosling, Patrick Naughton te neki drugi inženjeri firme Sun Microsystems. Razvoj Jave je započeo 1991. god. kao sastavni dio Green projekta, a objavljen je 1995. godine u studenom. Primjer java programskog kôda prikazan je prema programskom kodu 2.1.

```
public CrunchifyTimerTaskExample() {
    toolkit = Toolkit.getDefaultToolkit();
    timer = new Timer();
    timer.schedule(new CrunchifyReminder(), 0, // initial delay
            1 * 1000); // subsequent rate
}

class CrunchifyReminder extends TimerTask {
    int loop = 5;

    public void run() {
        if (loop > 0) {
            toolkit.beep();
            System.out.format("Knock Knock..!\n");
            loop--;
        } else {
            toolkit.beep();
            System.out.format("\nThat's it.. Done..!");
            timer.cancel();
        }
    }
```

Programski kôd 2.1. *Primjer Java programskog kôda.*

Samo značenje objektno orijentiranih programa se ogleda u metodama razvijanja programa gdje grupe objekata djeluju međusobno. Java programski jezik je dosta jednostavan i lakši za korištenje od npr. C ili C++. Java programski kôd se može izvoditi na svim operacijskim sustavima na kojima ima **JVM** (engl. *Java Virtual Mashine*), za razliku od nekih drugih programskih jezika kao što je npr. C. Java je programski jezik koji se danas najviše koristi, procjenjuje se da je trenutni broj korisnika Jave od 7 do 10 milijuna [6]. Jedna od naj bitnih značajki Java programskog jezika je višenitnost (engl. *threading*) odnosno podržava višenitno programiranje. Za razliku od C++ programskog jezika Java posjeduje automatsko oslobađanje memorije koje vrši skupljač smeća (engl. *Garbage Collector*).

2.4. XML

Proširivi opisni jezik (engl. *Extensible Markup Language*, XML) je jezik koji definira pravila i način kodiranja tako da to bude razumljivo i ljudima i računalima. On zapravo najčešće služi za kreiranje dizajna i izgleda naše aplikacije, kao što je i slučaj u Android Studiju. Zamisao bilo kojeg jezika za označavanje podataka je ta da se korisni sadržaj stavi između oznaka, a te bi oznake trebale biti jednostavne kako bi ih čovjek mogao lako razumjeti, ali opet ne prejednostavne kako bi ih računalni program mogao razlikovati. Jedan od najpoznatijih jezika za označavanje je HTML (engl. *HyperText Markup Language*). HTML je jezik koji se koristi prilikom izrade web stranice. Ako pogledamo primjer XML kôda prikazan programskim kôdom 2.2. može se primijetiti da na prvo gledanje taj sadržaj izgleda kompliciran, međutim ako ga krenemo čitati brzo nam postane razumljivo što se nalazi u kôdu i što se opisuje.

```
<?xml version="1.0" encoding="UTF-8" ?>
<memo>
    <contact type="person">
        <name>Ivo Ivić</name>
        <mobile>098 1234567</mobile>
        <email>ivo.ivic@fer.hr</email>
    </contact>
    <contact type="company">
        <name>Best Corporation</name>
        <mobile>091 123456</mobile>
        <email>best@best.com</email>
    </contact>
    <contact type="person">
        <name>Josip Horvat</name>
        <mobile>098 123456</mobile>
        <email>josip.horvat@fer.hr</email>
    </contact>
</memo>
```

Programski kôd 2.2. *Primjer XML kôda.*

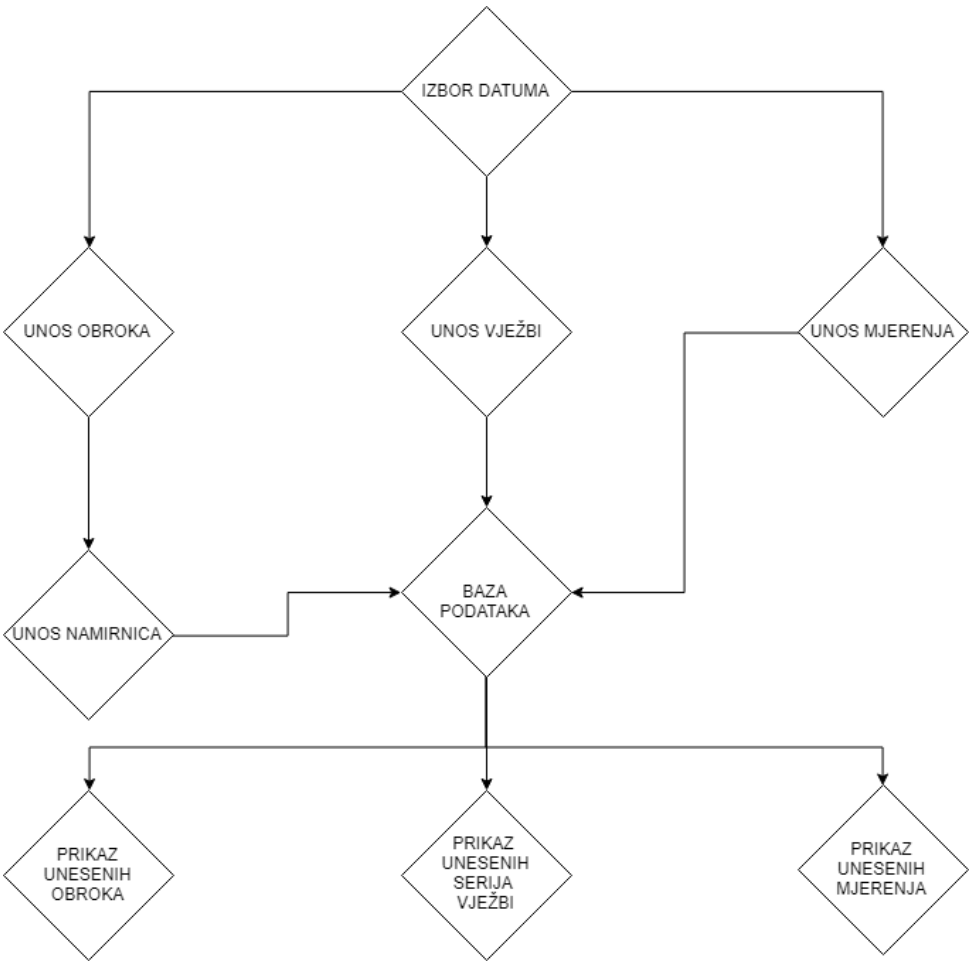
Postoje dvije verzije XML jezika za označavanje. Prva kreirana 1998. godine XML 1.0. do danas je imala par malih izmjena, općeprihvaćena verzija i danas se koristi i preporučuje za korištenje. Druga verzija pod nazivom XML 1.1 objavljena 2004. godine i ima određena svojstva da olakša rad programa na velikim i jakim računalima.

3. RAZVOJ MOBILNE APLIKACIJE

U ovom poglavlju je prikazan kompletan postupak izrade mobilne aplikacije, od dizajna preko konceptualnog modela do same implementacije projekta.

3.1. Konceptualni model projekta

Iz konceptualnog modela prikazan slikom 3.1. se vidi na koji bi način aplikacija trebala radit odnosno da nakon odabira datuma postoje tri važne komponente aplikacije, a to su unos obroka, unos vježbi i unos mjerenja. Iz unosa obroka se može otići na unos namirnica i dodati namirnice po želji te njihovu nutritivnu vrijednost. Uneseni podatci iz svih komponenti aplikacije se spremaju u bazu podataka. Odabirom određenog datuma može se pristupiti i vidjeti podatke o unesenim mjerenjima, obrocima i vježbama za taj odabrani datum.



Sl. 3.1. *Konceptualni model rada aplikacije*.

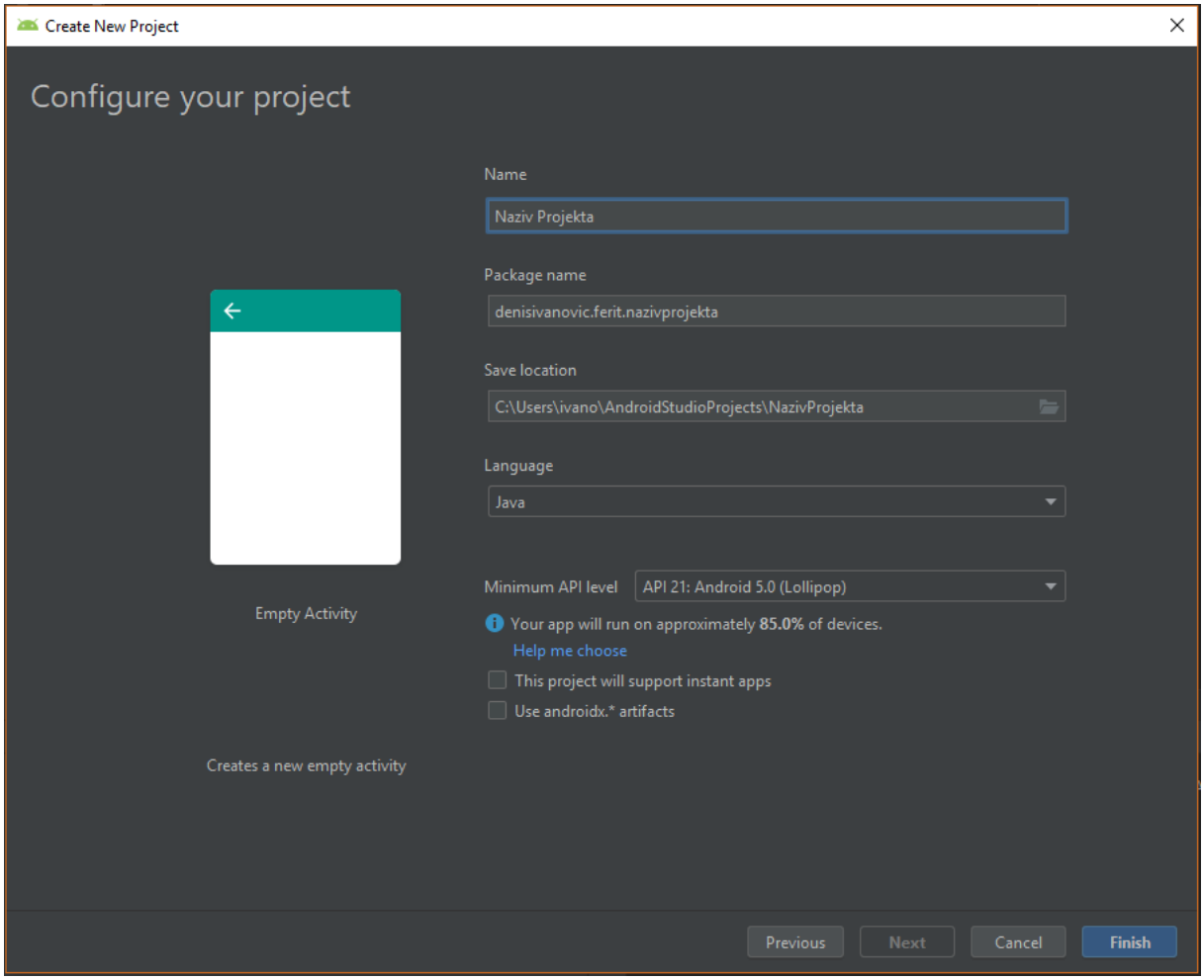
3.2. Dizajn i specifikacija projekta

Na slici 3.2 je prikazan početni zamišljeni dizajn aplikacije koji je napravljen na stranici FluidUI [12]. On služi kao glavna smjernica prilikom kreiranja dizajna same aplikacije. Prilikom izrade aplikacije može doći do odstupanja od zamišljenog dizajna, ali ukoliko je sam početni dizajn dobro kreiran ta odstupanja bi trebala biti minimalna.

Sl. 3.2. *Ogledni dizajn pojedinih zaslona aplikacije.*

3.3. Implementacija projekta

Nakon sastavljanja oglednog dizajna kreće sama implementacija i realizacija projekta. Obzirom da se radi u Android studiju počinje se sa kreiranjem projekta. Kreiranje projekta u Android Studiju se vrši na način da prilikom pokretanja aplikacija odaberemo „*Kreiraj novi projekt*“ (engl. *Create new project*), nakon toga se otvara prozor koji je prikazan slikom 3.3. u kojem se unose neke osnovne informacije kao što su naziv projekta, lokacija spremanja projekta, programski jezik u kojem pišemo aplikaciju, te Android verziju aplikacijskog programskog sučelja (engl. *application programming interface*, API).



Sl. 3.3. *Otvaranje novog projekta u Android studiju.*

Prilikom prvog pokretanja aplikacije od korisnika se traži unos osobnih podataka kao što su ime, prezime, godine starosti i spol. Na slici 3.4 je prikazan izgled zaslona na kojem se unose osobni podatci.

Sl. 3.4. *Unos osobnih podataka.*

Ovaj zaslon se otvara samo prilikom prvog pokretanja, odnosno nakon što se jednom unesu podatci i potvrdi unos prilikom svakog sljedećeg pokretanja aplikacije neće se otvarat ovaj zaslon. Ovo je ograničeno uz pomoć funkcije „*check_fullDatabase*“ prikazane prema programskom kôdu 3.2. koja provjerava tablicu koja sadrži podatke o korisniku te ukoliko je tablica prazna ovaj zaslon se pokreće, poziv funkcije se može vidjeti prema programskom kôdu 3.1.

```
if(myDb.check_fullDatabase()){
    Intent intent = new Intent(MainActivity.this, UserInfo.class);
    intent.putExtra(EXTRA_TEXT,curDate);
    startActivity(intent);
}
```

Programski kôd 3.1. *Provjera jesu li uneseni podatci o korisniku.*

```
public boolean check_fullDatabase() {
    open();
    Cursor res = db.rawQuery("select count(*) from "+TABLE_NAME,null);
    res.moveToFirst();
    int icount = res.getInt(0);
    if(icount>0){
        return false;
    }
    return true;
}
```

Programski kôd 3.2. *Funkcija check_fullDatabase koja provjerava ima li unosa u tablici.*

Na slici 3.5. je prikazan početni zaslon aplikacije na kojem se nalazi kalendar za odabir datum, tri tipke „*Unesite Prehranu*“, „*Unesite Vježbe*“ i „*Unesite Mjerenje*“ te pri dnu jedan dio za ispis postotka potkožnog masnog tkiva.

Sl. 3.5. *Prikaz početnog zaslona aplikacije.*

Postotak potkožnog masnog tkiva se računa i prikazuje u ovisnosti o zadnjem unosu mjerenja, ukoliko nije uneseno ni jedno mjerenje na mjestu gdje se ispisuje postotak potkožnog masnog tkiva ispisuje se tekst „*Morate unijeti barem jedno mjerenje*“ te se ne poziva metoda „*bodyFatCalculate*“ za izračun potkožnog masnog tkiva što je ograničeno „*if*“ izjavom koja je prikazana na programskom kôdu 3.3. Ukoliko ima prethodno uneseno mjerenje to se izračunava pomoću metode „*bodyFatCalculate*“ koja se može vidjeti na programskom kôdu 3.4. „*bodyFatCalculate*“ metoda dohvaća sva unesena mjerenja prija datuma odabranog na kalendaru i izračunava postotak potkožnog masnog tkiva za zadnje od tih mjerenja. Prilikom računanja postotka potkožnog masnog tkiva kod muškaraca uzimaju se u obzir sljedeći parametri: obujam vrata, obujam struka te visina, dok se kod žena gleda obujam vrata, obujam struka, obujam kukova i visina. Računanje se vrši prema formuli američke vojske za računanje postotka potkožnog masnog tkiva prikazanoj prema literaturi [11].

```
if(myDb_mjere.check_fullDatabase()){
    bodyFatCalculate();
}
else{
    TextView procent_show = findViewById(R.id.textView_prikaz_postotaka);
    procent_show.setTextSize(15);
    procent_show.setText("Morate unijeti barem jedno mjerenje!!");
}
```

Programski kôd 3.3. *Provjera ima li unosa u tablici mjerenja.*

```
public void bodyFatCalculate() {

    Mjerenje res=new Mjerenje();
    Cursor mjerenje = myDb_mjere.getAllData_byDate(res.date_calculate(godina,mjesec,dan));
    if (mjerenje.moveToLast()) {
        Cursor userInfo = myDb.getAllData();
        userInfo.moveToFirst();
        TextView procent_show;
        procent_show = findViewById(R.id.textView_prikaz_postotaka);
        double bodyFat;
        String spol = userInfo.getString(3);

        if (spol.equals("m")) {
            bodyFat=(495/(1.0324-0.19077*Math.log10(Integer.parseInt(mjerenje.getString(6)) -
            Integer.parseInt(mjerenje.getString(3)))+
            0.15456*Math.log10(Integer.parseInt(mjerenje.getString(2)))))-450;
        } else if (spol.equals("f")) {
            bodyFat=(495/(1.29579-0.35004*Math.log10(Integer.parseInt(mjerenje.getString(6))+
            Integer.parseInt(mjerenje.getString(7)) - Integer.parseInt(mjerenje.getString(3)))+
            0.22100*Math.log10(Integer.parseInt(mjerenje.getString(2)))))-450;
        }
    }
}
```

Programski kôd 3.4. *Kôd bodyFatCalculate metode.*

Do jednog od najvažnijih dijelova aplikacije se dolazi pritiskom na tipku „*Unesite prehranu*“, a to je dio za unos namirnica i obroka. Na slici 3.6. se nalazi prikaz zaslona koji služi za unos obroka i namirnica, te prikaz unesenih obroka i ukupne sume nutritivnih vrijednosti koje je korisnik unio za taj dan.

Prikaz obroka se vrši uz pomoć „*RecyclerView*“ kao što je prikazano na slici 3.6. „*RecyclerView*“ je spremnik za učitavanje većeg broja podataka i njegovo prikazivanje u jednoj listi koja se može pomicat i listat. „*RecyclerView*“ je puno bolji i učinkoviti od običnog „*ListView*“ zato što u radnoj memoriji učita puno više podataka te tako prilikom listanja ne dođe do zastajkivanja rada aplikacije.

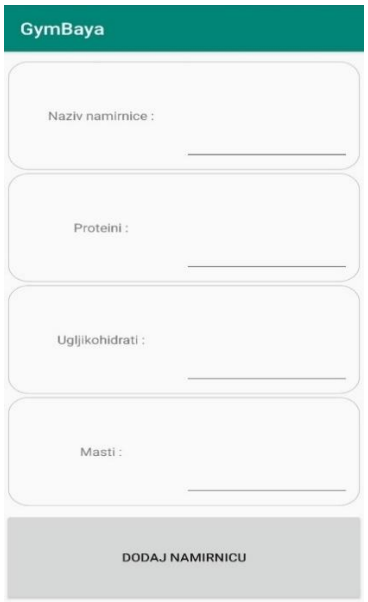
Sl. 3.6. *Prikaz zaslona za unos obroka i namirnica.*

Prema programskom kôdu 3.5. prikazana je metoda „*Refresh_RecyclerViewa*“ u kojoj se prvo iz baze uz pomoć funkcije „*getAllData_byDate*“ (koja kao parametar prima trenutni datum) dohvaćaju uneseni obroci za taj dan. Nakon toga se kreira objekt klase „*Popis_Obroka_Adapter*“ pod nazivom „*mAdapter*“ čiji konstruktor prima parametre kontekst trenutne klase i kursor sa dohvaćenim podatcima obroka, te na kraju „*Popis_Obroka_Adapter*“ ispisuje podatke u „*RecyclerView*“.

```
public void Refresh_RecyclerViewa() {
        final String DATUM = getIntent().getStringExtra(MainActivity.EXTRA_TEXT);
        RecyclerView recyclerView = findViewById(R.id.RecyclerView);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));
        Cursor kursor = myDb_add.getAllData_byDate(DATUM);
        kursor.moveToFirst();
        mAdapter = new Popis_Obroka_Adapter(this, kursor);
        recyclerView.setAdapter(mAdapter);
        total_proteins.setText(String.format("%.2f", myDb_add.getSumOfProteins_byDate(DATUM)));
        total_calories.setText(String.format("%.2f", myDb_add.getSumOfCalories_byDate(DATUM)));
        total_carbh.setText(String.format("%.2f", myDb_add.getSumOfCarbH_byDate(DATUM)));
        total_fat.setText(String.format("%.2f", myDb_add.getSumOfFat_byDate(DATUM)));
        new ItemTouchHelper(new ItemTouchHelper.SimpleCallback(0,
                ItemTouchHelper.LEFT | ItemTouchHelper.RIGHT) {
            @Override
            public boolean onMove(RecyclerView recyclerView, RecyclerView.ViewHolder viewHolder,
            RecyclerView.ViewHolder target) {
                return false;
            }
            @Override
            public void onSwiped(RecyclerView.ViewHolder viewHolder, int direction) {
                removeItem((int) viewHolder.itemView.getTag());
            }
        }).attachToRecyclerView(recyclerView);
    }
```

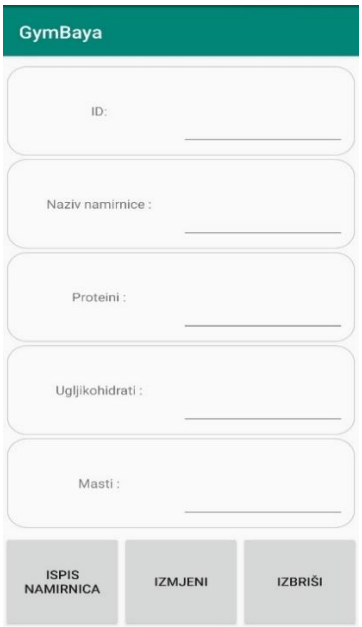
Programski kôd 3.5. *Kôd Refresh_RecyclerViewa metode.*

Prije dodavanja odnosno kreiranja obroka mora se dodati minimalno jedna namirnica. Dodavanje se vrši na zaslonu koji se otvara nakon pritiska na tipku „*Dodaj namirnicu*“ , koji je prikazan na slici 3.7.



Sl. 3.7. *Prikaz zaslona za dodavanje namirnica.*

Na slici 3.8. prikazan zaslon za brisanje i izmjenu dodanih namirnica, koji se otvara pritiskom na tipku „*Izmjeni namirnicu*“.



Sl. 3.8. *Prikaz zaslona za izmjenu i brisanje namirnica.*

Dodavanje namirnica vrši se tako da se prvo popune tražena polja (naziv namirnice, iznos proteina na 100 grama, iznos ugljikohidrata na 100 grama, iznos masti na 100 grama). Nakon toga pritiskom na tipku aktivira se „*onClickListener*“ koji onda kreira novi objekt klase „*Food*“ i u konstruktor daje sljedeće parametre: kontekst trenutne klase, ime namirnice, iznos unesenih proteina, iznos unesenih ugljikohidrata i iznos unesenih masti kao što je prikazano na programskom kôdu 3.6..

```
public  void AddData() {
    btnAddData.setOnClickListener(
            new View.OnClickListener() {
                @Override
                public void onClick(View v) {

                    Food food=new Food(
                            getApplicationContext(),
                            editName.getText().toString(),
                            editProteini.getText().toString(),
                            editUgljikohidrati.getText().toString(),
                            editMasti.getText().toString());
                    food.AddFood();



                }
            }
    );
}
```

Programski kôd 3.6. *Kôd AddData metode*.

Nakon kreiranja objekta „*food*“ klase „*Food*“ poziva se metoda klase „*Food*“ koja se zove „*AddFood*“ prikazana prema programskom kôdu 3.7. koja prvo provjerava dali je korisnik popunio sva polja odnosno dali je unio sve parametre namirnice, ako nije ispisuje „*Toast*“ poruku kao upozorenje i prekida daljnje izvođenje. „*Toast*“ je klasa implementirana u Android studio koja omogućava ispis poruke na zaslon u kratkom vremenskom periodu.

Ako je sve ispravno uneseno onda se putem funkcije „*insertData*“ u bazu podataka upisuju sljedeći podatci: naziv namirnice, kalorije, proteini, ugljikohidrati i masti. Ali prilikom davanja parametara za kalorije funkciji „*insertData*“ prema programskom kôdu 3.8. poziva se funkcija „*calculateKalorije*“ prema programskom kôdu 3.9. koja uz pomoć ostala tri parametra (proteini, ugljikohidrati i masti) računa iznos kalorija, prema formuli prikazanoj u literaturi [10].

```
public void AddFood (){

    int checkfood=myDb.checkFood(naziv_jela);

    if (naziv_jela.isEmpty()) {
        Toast.makeText(kontext , "Unesite ime jela !!", Toast.LENGTH_LONG).show();
    }
    else if (proteini.isEmpty()){
        Toast.makeText(kontext, "Unesite iznos proteina na 100 grama !!", Toast.LENGTH_LONG).show();
    }
    else if (ugljikohidrati.isEmpty()){
        Toast.makeText(kontext, "Unesite iznos ugljikohidrata na 100 grama !!", Toast.LENGTH_LONG).show();
    }
    else if (masti.isEmpty()){
        Toast.makeText(kontext, "Unesite iznos masti na 100 grama !!", Toast.LENGTH_LONG).show();
    }
    else if(checkfood == 0){
        Toast.makeText(kontext, "Već postoji jelo sa tim nazivom !!", Toast.LENGTH_LONG).show();
    }
    else {

        boolean isInserted = myDb.insertData(getNaziv_jela(),
                calculateKalorije(Double.parseDouble(getUgljikohidrati()),
                                  Double.parseDouble(getProteini()),
                                  Double.parseDouble(getMasti())),
                Double.parseDouble(getProteini()),
                Double.parseDouble(getUgljikohidrati()),
                Double.parseDouble(getMasti()));
        if (isInserted == true)
            Toast.makeText(kontext, kontext.getString(R.string.food_added), Toast.LENGTH_LONG).show();
        else {
            Toast.makeText(kontext, kontext.getString(R.string.error_toast), Toast.LENGTH_LONG).show();
        }
    }

}
```

Programski kôd 3.7. *Kôd AddFood metode.*

```
public boolean insertData(String datum, String ime_jela,Double kalorije,
Double proteini, Double ugljikohidrati, Double masti, int gramaza)
{
    open();
    ContentValues contentValues = new ContentValues();
    contentValues.put(COL_1,datum);
    contentValues.put(COL_2,ime_jela);
    contentValues.put(COL_3,kalorije);
    contentValues.put(COL_4,proteini);
    contentValues.put(COL_5,ugljikohidrati);
    contentValues.put(COL_6,masti);
    contentValues.put(COL_7,gramaza);
    long result = db_add.insert(TABLE_NAME,null ,contentValues);
    if(result == -1)
        return false;
    else
        return true;
}
```

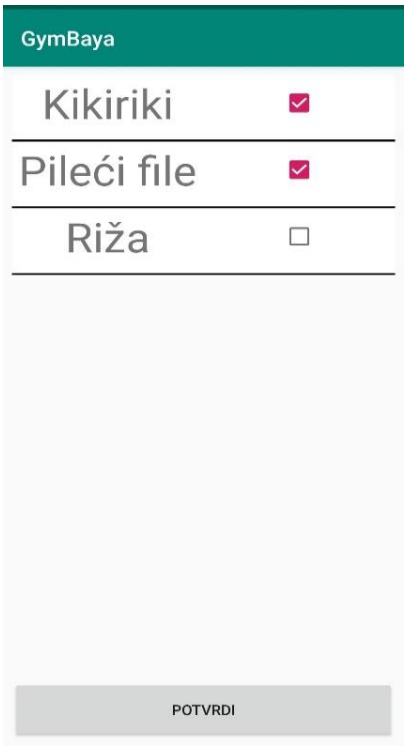
Programski kôd 3.8. *Kôd insertData funkcije.*

```
public Double calculateKalorije(Double UH, Double Proteini, Double Masti) {

    Double kalorije= (Proteini * 4 + UH * 4 + Masti * 9);

    return kalorije;
}
```

Programski kôd 3.9. *Kôd calculateKalorije funkcije.*

Kada je barem jedna namirnica dodana u bazu podataka, može se pristupiti kreiranju obroka. Prema slici 3.6. vidimo tipku „*Kreiraj obrok*“, nakon što se pritisne ta tipka otvara se zaslon za odabir namirnica koje korisnik želi u obroku, prikazano prema slici 3.9., prikaz namirnica se vrši preko „*RecyclerView*“ i adaptera za popunjavanje tog „*RecyclerView*“.

Dohvaćanje i predaja podataka adapteru je slična kao kod prikaza unesenih obroka prikazano prema programskom kôdu 3.5., dok se sam „*RecyclerView*“ popunjava u klasi „*Popis_CreateMeal_Adapter*“. Tu treba istaknuti „*onBindViewHolder*“ metodu prikazanu na programskom kôdu 3.10. koja popunjava pojedine „*ViewHolder*“ objekte liste „*RecyclerView*“, i sadrži „*onCheckedChangeListener*“ koji se aktivira kada se pojedini potvrdni okvir označi ili odznači, i ovisno dali je neki objekt liste označen ili odznačen dodaje njegov „*ID*“ ili ga uklanja iz polja brojeva „*namirnice*“.



Sl. 3.9. *Prikaz zaslona za dodavanje namirnica.*

```
public void onBindViewHolder(@NonNull Popis_CreateMeal_ViewHolder holder, int position) {
if (!mCursor.moveToPosition(position)) {
    return;
}
String imeJela = mCursor.getString(mCursor.getColumnIndex("IME_NAMIRNICE"));
final int id = mCursor.getInt(mCursor.getColumnIndex("ID"));
holder.imeJela.setText(imeJela);
holder.itemView.setTag(id);
holder.selectionState.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        if(isChecked){
            if(track==-1){
                namirnice[count]=id;
                count+=1;
                broj_zakacenih+=1;
            }
            else{
                namirnice[track]=id;
                track=-1;
                broj_zakacenih+=1;
                for(int i = 0;i<c;i++){
                    if(namirnice[i]==-1){
                        track=i;
                    }}}}
        else if (!isChecked){
            for(int i = 0;i<c;i++){
                if(namirnice[i]==id){
                    namirnice[i]=-1;
                    track=i;
                    broj_zakacenih-=1;
                }}}});
}
```

Programski kôd 3.10. *Kôd onBindViewHolder metode.*

Nakon završenog odabira namirnica i pritisnute tipke „*Potvrdi*“ prema slici 3.9. u „*onClickListener*“ prema programskom kôdu 3.11. se najprije provjerava da li je odabrana barem jedna namirnica te ukoliko nije ispisuje se „*Toast*“ poruka korisniku. Ukoliko je odabrana barem jedna namirnica dohvaća se polje „*namirnice*“ koje sadrži ID-ove odabranih namirnica te se to polje preko „*putExtra*“ metode iz klase „*Intent*“ prenosi na sljedeći zaslon prema slici 3.10., isto kao i podatak koliko iznosi broj označenih namirnica. Te se na kraju metodom „*startActivity*“ pokreće sljedeći zaslon na kojem se unosi masa za pojedinu odabranu namirnicu, prikazano prema slici 3.10.

Popis odabranih namirnica sa poljima za unos mase se ispisuje na identičan način kao i kod prethodnog zaslona preko „*RecyclerView*“ i klase odnosno adaptera „*Popis_PopulateMeal_Adapter*“. S tim da se sada prilikom kreiranja objekta klase „*Popis_PopulateMeal_Adapter*“ u konstruktor pored konteksta i kursora sa podatcima odabranih namirnica daje i parametar polja brojeva „*namirnice*“ koje sadrži ID-ove odabranih namirnica.

Sl. 3.10. *Prikaz zaslona za izmjenu i brisanje namirnica*.

Taj adapter u svaki objekt liste „*RecyclerView*“ implementira „*TextChangeListener*“ prikazan prema programskom kôdu 3.12. koji prilikom unosa mase za pojedinu namirnicu postavlja unesenu vrijednost u polje brojeva „*podatci_2*“ na točno onome indexu na kojem se nalazi ID te namirnice u polju „*namirnice*“.

```
btn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {

            if(mAdapter.broj_zakacenih==0){
                Toast.makeText(getApplicationContext(),
                "Morate odabrati barem jednu namirnicu !!",
                Toast.LENGTH_LONG).show();
            }
            else {
                Intent intent = new Intent(Meal_Create.this, Meal_Populate.class);
                intent.putExtra("sve", mAdapter.namirnice);
                intent.putExtra("broj_zakacenih", mAdapter.broj_zakacenih);
                String DATUM = getIntent().getStringExtra(MainActivity.EXTRA_TEXT);
                intent.putExtra(EXTRA_TEXT, DATUM);
                startActivity(intent);
                finish();
            }

        }
    });
```

Programski kôd 3.11. *Kôd onClickListener za tipku "Potvrdi" sa zaslona za odabir namirnica.*

```
public void onTextChanged(CharSequence s, int start, int before, int count) {

    for(int i = 0;i<namirnice.length;i++){
        if(namirnice[i]==id){
            if(holder.grami.getText().toString().isEmpty()){
              //
            }
            else {
                podatci_2[i] = Integer.parseInt(holder.grami.getText().toString());
            }
        }
    }
```

Programski kôd 3.12. *Kôd onTextChange metode u objektu liste RecyclerView.*

Neposredno poslije unosa mase za namirnice pritiskom na tipku „*Dodaj obrok*“ poziva se *boolean* funkcija „*Calculate_Meal*“ prikazana prema programskom kôdu 3.13. , koja kreira objekt „*meal*“ klase „*Meal*“ i u konstruktor daje parametre: kontekst trenutnog zaslona, polje brojeva „*podatci_2*“ (koje sadrži unesenu masu pojedinih namirnica), polje brojeva „*namirnice*“ (koje sadrži ID-ove odabranih namirnica) i trenutni odabrani datum.

```
public boolean Calculate_Meal(){
    final String DATUM = getIntent().getStringExtra(MainActivity.EXTRA_TEXT);
    Intent intent = getIntent();
    namirnice = intent.getIntArrayExtra("sve");

        Meal meal = new Meal(
                getApplicationContext(),
                mAdapter.podatci_2,
                namirnice,
                DATUM);

        if(meal.AddMeal()){
            return true;
        }else {
            return false;
        }
}
```

Programski kôd 3.13. *Kôd Calculate_Meal funkcije.*

Nakon kreiranja objekta „*meal*“ klase „*Meal*“ poziva se funkcija „*AddMeal*“ klase „*Meal*“ prema programskom kôdu 3.14. Koja ukoliko su za sve namirnice unesene mase prolazi kroz polje „*namirnice*“ i za svaku namirnicu ovisno o njenoj masi i nutritivnoj vrijednosti računa iznos kalorija, proteina, ugljikohidrata i masti, te vrijednost dodaje varijablama „*total_kalories*“, „*total_proteins*“, „*total_uh*“ i „*total_masti*“ koje zbrajaju ukupne nutritivne vrijednosti za taj čitav

obrok. Kada su zbrojene sve nutritivne vrijednosti funkcijom „*insertData*“ se upisuju u bazu podataka.

```
public boolean AddMeal() {
    for (int i = 0; i<namirnice.length;i++) {
        Cursor mrki = myDb.getAllData_byID(namirnice[i]);

        if (mrki.moveToFirst()) {
            if (podatci_2[i] == 0) {
                Toast.makeText(kontext,
                "Morate unijeti težinu za sve odabrane namirnice !!",
                Toast.LENGTH_LONG).show();
                return false;
            }
            total_kalories += calories(mrki.getString(1), podatci_2[i]);
            total_proteins += proteins(mrki.getString(1), podatci_2[i]);
            total_uh += carbohydrate(mrki.getString(1), podatci_2[i]);
            total_masti += fat(mrki.getString(1), podatci_2[i]);
            gramaza = gramaza + podatci_2[i];
            }
    }
    int count = myDb_add.count_meals(datum) + 1;
    naziv_jela = count + ". Obrok";
    boolean isInserted = myDb_add.insertData(
        getDatum(),
        getNaziv_jela(),
        getKalorije(),
        getProteini(),
        getUgljikohidrati(),
        getMasti(),
        getGramaza());

        return true;
}
```

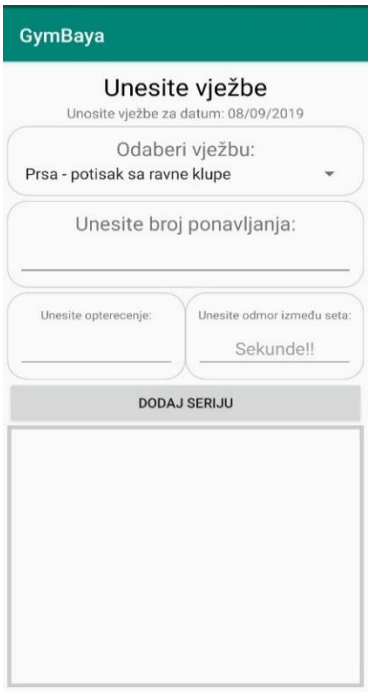
Programski kôd 3.14. *Kôd AddMeal funkcije*.

Poslije unosa prehrane sljedeći segment aplikacije je unos vježbi na koji se prelazi pritiskom tipke „*Unesite vježbe*“ prema slici 3.5. Pritiskom te tipke prebacujemo se na zaslon za unos vježbi prikazan prema slici 3.11.

Da bi se dodala odrađena serija neke vježbe mora se prvo iz padajućeg izbornika (engl. *Drop Down Menu*) odabrati o kojoj vježbi je riječ, onda se unosi broj odrađenih ponavljanja, iznos opterećenja i iznos sekundi odmora prije serije. Te se onda pritiskom na tipku „*Dodaj seriju*“ pokreće metoda „*AddData*“ klase „*VjezbeView*“ prema programskom kôdu 3.17. koja prvo provjerava jesu li sva polja popunjana te ukoliko nisu preko „*Toast*“ poruke obavještava korisnika.

Ako je sve popunjeno funkcijom „*insertData*„ se u bazu podataka unose informacije o odrađenoj seriji (trenutni datum, ime vježbe, broj ponavljanja, iznos opterećenja i vrijeme odmora

). Unesene serije se ispisuju identično preko „*RecyclerView*“ kao i kod obroka kako je to prethodno objašnjeno.



Sl. 3.11. *Izgled zaslona za unos vježbi.*

Ako se želi ukloniti neka dodana serija (isto vrijedi i za obrok) potrebo je samo klizno pomaknuti objekt (engl. *swipe*) u „*RecyclerView*“ listi lijevo ili desno i tako aktiviramo „*ItemTouchHelper*“ prema programskom kôdu 3.16. koji onda poziva metodu „*removeItem*“ prikazanu prema programskom kôdu 3.15., koja kao parametar prima ID objekta koji je pomaknut lijevo ili desno te ga briše iz baze podataka.

```
private void removeItem(int id) {
    String DATUM= getIntent().getStringExtra(MainActivity.EXTRA_TEXT);
    myDb.deleteData(id);
    Cursor kursor = myDb.getAllData(DATUM);
    mAdapter.swapCursor(kursor);
}
```

Programski kôd 3.15. *Kôd removeItem metode.*

```
        new ItemTouchHelper(new ItemTouchHelper.SimpleCallback(0,
                ItemTouchHelper.LEFT | ItemTouchHelper.RIGHT) {
            @Override
            public boolean onMove(RecyclerView recyclerView,
                                  RecyclerView.ViewHolder viewHolder,
                                  RecyclerView.ViewHolder target) {
                return false;
            }

            @Override
            public void onSwiped(RecyclerView.ViewHolder viewHolder, int direction) {
                removeItem((int) viewHolder.itemView.getTag());
            }
        }).attachToRecyclerView(recyclerView);
```

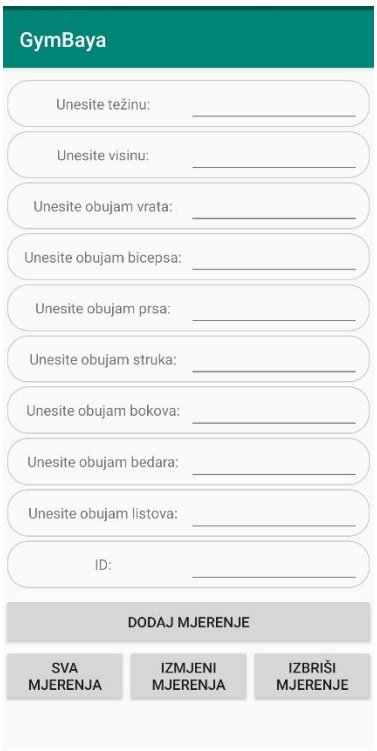
Programski kôd 3.16. *ItemTouchHelper.*

```
public  void AddData() {
    btnAddData.setOnClickListener(
        new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (mySpinner.getSelectedItem() == null) {
                    Toast.makeText(getApplicationContext(), "Odaberite vježbu !!",
                    Toast.LENGTH_LONG).show();
                }
                else if (broj_ponavljanja.getText().toString().isEmpty()){
                    Toast.makeText(getApplicationContext(), "Unesite broj ponavljanja !!",
                    Toast.LENGTH_LONG).show();
                }
                else if (opterecenje.getText().toString().isEmpty()){
                    Toast.makeText(getApplicationContext(), "Unesite opterećenje !!",
                    Toast.LENGTH_LONG).show();
                }
                else if (odmor.getText().toString().isEmpty()){
                    Toast.makeText(getApplicationContext(), "Unesite odmor između seta u sekundama !!",
                    Toast.LENGTH_LONG).show();
                }
                else {
                    mySpinner = findViewById(R.id.spinner_vjezbe);
                    broj_ponavljanja = findViewById(R.id.editText_broj_ponavljanja);
                    opterecenje = findViewById(R.id.editText_opterecenje);
                    String DATUM = getIntent().getStringExtra(MainActivity.EXTRA_TEXT);
                    String IME_VJEZBE = mySpinner.getSelectedItem().toString();
                    int BROJ_PONAVLJANJA = (Integer.parseInt(broj_ponavljanja.getText().toString()));
                    int OPTERECENJE = (Integer.parseInt(opterecenje.getText().toString()));
                    boolean isInserted = myDb.insertData(DATUM, IME_VJEZBE, BROJ_PONAVLJANJA,
                    OPTERECENJE, Integer.parseInt(odmor.getText().toString()));
                    if (isInserted == true)
                        Toast.makeText(VjezbeView.this, getString(R.string.extercise_added),
                    Toast.LENGTH_LONG).show();
                    else
                        Toast.makeText(VjezbeView.this, getString(R.string.error_toast),
                    Toast.LENGTH_LONG).show();

                    Refresh_RecyclerViewa();
                }}});
}
```

Programski kôd 3.17. *Kôd AddData funkcije klase VjezbeView.*

Posljednji segment aplikacije je dodavanje mjerenja, pritiskom tipke „*Unesite mjerenje*“ na zaslonu prikazan slikom 3.5., otvara se zaslon za dodavanje mjerenja prikazan slikom 3.12. Da bi se unijelo mjerenje prvo se moraju popuniti sva polja prikazana na slici.

Tipka „*Dodaj mjerenje*“ pritiskom aktivira metodu „*AddData*“ koja dohvaća sva polja i kreira objekt klase „*Mjerenje*“ koji prima za parametre dohvaćene vrijednosti iz polja za unos. Te se poziva metoda „*AddMjerenje*“ klase „*Mjerenje*“ koja provjerava jesu li unesena sva polja te ukoliko nisu „*Toast*“ porukom obavještava korisnika, ako je sve ispravno uneseno poziva se *boolean* funkcija „*insertData*“ koja upisuje podatke o mjerenju u tablicu.



Sl. 3.12. *Izgled zaslona za dodavanje mjerenja.*

4. NAČIN KORIŠTENJA APLIKACIJE

U četvrtom poglavlju prikazan je način korištenja aplikacije, što se sve može uraditi s aplikacijom, na koji način i sl.

4.1. Korisnički zahtjevi

Tablicom Tab. 4.1. prikazani su korisnički zahtjevi koji moraju biti ispunjeni prilikom izrade mobilne aplikacije, a oni se ogledaju kroz zadatak završnog rada.

ID	Status	Prioritet	Opis	UC
			Generalni zahtjevi korisnika	
1	A	1	Postoji mogućnost unošenja vlastitih namirnica	UC2
2	A	1	Postoji mogućnost kreiranja obroka	UC1
3	A	1	Postoji mogućnost dnevnog praćenja	-
4	A	1	Postoji mogućnost vođenja evidencije odrađenih vježbi	UC3
5	A	1	Korisnik može unositi ponavljanja, opterećenje i vrijeme odmora za određenu seriju vježbe	UC3
6	A	1	Korisnik može brisati namirnice, obroke i vježbe	-
7	A	2	Sve se sprema u bazu podataka	-
8	A	1	Postoji mogućnost unosa, izmjene i brisanja fizičkih mjerenja	UC4

Tab. 4.1. *Tablica sa zahtjevima prilikom izrade aplikacije.*

4.2. Slučajevi korištenja

Slučajevi korištenja aplikacije UC1, UC2, UC3 i UC4 koji objašnjavaju glavne korisničke zahtjeve iz tablice Tab. 4.1. prikazani su kroz scenarij koji bi korisnik trebao slijediti kako bi izvršio radnju koju taj slučaj pokriva.

Slučajevi korištenja:

- Slučaj UC1 pokriva scenarij kada korisnik kreira obrok, a prikazan je tablicom u prilogu P.4.1.
- Slučaj UC2 pokriva scenarij kada korisnik unosi namirnicu, a prikazan je tablicom u prilogu P.4.2.
- Slučaj UC3 pokriva scenarij kada korisnik unosi seriju za vježbu koju je odradio, a prikazan je tablicom u prilogu P.4.3.
- Slučaj UC4 pokriva scenarij kada korisnik unosi mjerenje, a prikazan je tablicom u prilogu P.3.7.

4.3. Upute za korištenje

Prilikom prvog pokretanja aplikacije korisnik mora unijeti svoje osobne podatke kako bi nastavio sa daljnjim korištenjem aplikacije, kao što je detaljnije prikazano u odjeljku 4.3.1. Prilikom svakog sljedećeg pokretanja aplikacije korisniku se prikazuje kalendar za odabir datuma i tri kategorije, te ukoliko je uneseno barem jedno mjerenje i postotak potkožnog masnog tkiva prikazano prema slici 3.5.

4.3.1. Unos korisničkih podataka

Nakon instalacije aplikacije prilikom prvog pokretanja pokreće se zaslon za unos korisničkih podataka prikazan slikom 3.4. Na tom zaslonu korisnik unosi osobne podatke kao što je ime, prezime, spol i godine starosti. Kada se unesu svi traženi podatci i potvrde pritiskom na tipku „*Potvrdi unos*“ otvara se početni zaslon aplikacije prethodno prikazan slikom 3.5.

4.3.2. Unos namirnica

Kako bi se dodale namirnice potrebno je pritisnuti tipku „*Dodaj namirnice*“ prikazanu prema slici 3.6., nakon čega nam se otvara zaslon za dodavanje namirnica prikazan slikom 3.7.

U polja prikazana slikom 3.7. potrebno je unijeti tražene podatke, a to su naziv namirnice, iznos grama proteina na 100 grama namirnice, iznos grama ugljikohidrata na 100 grama namirnice i iznos grama masti na 100 grama namirnice, te na kraju kako bi se potvrdio unos potrebno je pritisnuti tipku „*Dodaj namirnicu*“.

Da bi izmijenili ili izbrisali dodane namirnice pritisnemo tipku „*Izmjeni namirnicu*“ prikazanu prema slici 3.6., nakon čega nam se otvara zaslon za izmjenu i brisanje namirnica prikazan slikom 3.8. Brisanje namirnice se izvodi unošenjem ID-a dodane namirnice u polje za unos ID-a prema slici 3.8. i na kraju pritiskom tipke „*Izbriši*“. Da bi saznali ID namirnice potrebno je prema slici 3.8. pritisnuti tipku „*Ispis namirnica*“ koja će onda u dijaloškom okviru ispisati informacije o svim dodanim namirnicama.

4.3.3. Unos prehrane

Obroci se dodaju odnosno kreiraju tako da se nakon što u bazi podataka imamo barem jednu namirnicu pritisne tipka „*Kreiraj obrok*“ na zaslonu za unos prehrane prema slici 3.6. Potom se otvara zaslon prema slici 3.9. za odabir namirnica koje se nalaze u našem obroku. Na tom zaslonu se označuju potvrdni okviri pored namirnica koje imamo u obroku. Te kad smo završili potvrđujemo odabir pritiskom na tipku „*Potvrdi*“ koja vodi na sljedeći zaslon za unos mase pojedine odabrane namirnice prikazan prema slici 3.10. Na tom zaslonu se unosi masa u gramima u polje za unos pored naziva svake izabrane namirnice. Nakon što smo unijeli masu za svaku odabranu namirnicu pritiskom na tipku „*Dodaj obrok*“, obrok se dodaje u bazu podataka i ispisuje u „*RecyclerView*“ na zaslonu prikazan slikom 3.6.

Ukoliko korisnik nije dobro unio obrok ili iz nekog drugog razloga želi izbrisati obrok to čini povlačenjem obroka lijevo ili desno sa liste u „*RecyclerView“* prikazana prema slici 3.6.

4.3.4. Unos vježbi

Ukoliko korisnik odabere kategoriju unos vježbi odnosno pritisne tipku „*Unesite vježbe*“ otvori mu se zaslon prema slici 3.11. Na kojem unosi seriju po seriju koju je odradio. Prvenstveno u padajućem izborniku korisnik izabire vježbu koju je odradio, onda unosi broj ponavljanja koji je odradio, masu opterećenja u kilogramima koje je koristio i na kraju vrijeme odmora u sekundama prije izvođenja serije. Nakon što je sve uneseno pritiskom na tipku „*Dodaj seriju*“ serija se dodaje u bazu podataka i automatski ispisuje na listu „*RecyclerView*“ prikazana na zaslonu prema slici 3.11.

4.3.5. Unos mjerenja

Ukoliko korisnik odabere kategoriju unos mjerenja odnosno pritisne tipku „*Unesite mjerenje*“ otvori mu se zaslon prema slici 3.12. Na kojem korisnik da bi unio mjerenje mora unijeti sljedeće mjere: masu u kilogramima, visinu u centimetrima, obujam vrata u centimetrima, obujam bicepsa u centimetrima, obujam prsa u centimetrima, obujam struka u centimetrima, obujam bokova u centimetrima, obujam bedara u centimetrima i obujam listo u centimetrima. Nakon unosa korisnik potvrđuje unos pritiskom na tipku „*Dodaj mjerenje*“ koja mjerenje sprema u bazu podataka i automatski računa novi postotak potkožnog masnog tkiva shodno posljednjem unesenom mjerenju.

5. ZAKLJUČAK

Kako i sam naziv rada kaže „*Android aplikacija za praćenje dnevnog treninga*“ to je mobilna aplikacija koja korisniku omogućava praćenje njegovog napretka na dnevnoj bazi. Prvi dio zadatka je bio izraditi rješenje kako bi korisnik mogao unositi informacije o sebi i svom tijelu te vidjeti kroz usporednu nekoliko zadnjih mjerenja napredak koji je postignut. Taj problem je riješen na jedan jednostavan, ali praktičan način, tako da se mjerenja unose i upisuju u bazu te je uvijek moguće vidjeti ispis svih mjerenja i usporediti ih. Tu je i pokazatelj koji reprezentira mjerenje u vidu postotka potkožnog masnog tkiva preko kojeg se uvijek može vidjeti napredak treninga i prehrane. Drugi dio zadatka je bio napraviti jedan oblik evidencije o prehrani koju smo unijeli na dnevnoj bazi. To rješenje je malo zahtjevnije za izradu, ali lako za korištenje. Prvobitno je zamišljeno da korisnik sam sebi pravi svoju lokalnu bazu namirnica, kako ne bi došao u situaciju da se u popisu namirnica nalaze one namirnice koje korisnik nikada i ne upotrebljava. Te onda kroz korištenje ima mogućnost uvijek nadograđivati tu bazu sa novim namirnicama. Kada postoji baza s namirnicama tada je moguće na jednostavan način kreirati obrok koji sadrži iznos nutritivnih vrijednosti. Treći i zadnji dio zadatka je implementiranje mogućnosti praćenja odrađenih vježbi tijekom dana. To je napravljen slično kao i prehrana samo na malo jednostavniji način jer ne moramo unositi vježbe nego su one već kroz aplikaciju implementirane, te je potrebno samo izabrati vježbu, unijeti broj povaljanija, opterećenje i vrijeme odmora kako bi se mogla dodati serija neke odrađene vježbe. Kad se rezimira sve, ova aplikacija i sve aplikacije ovog karaktera imaju veliku mogućnost za razvoj, jer zbog sve ubrzanijeg tempa života puno se više unosi loša hrana i ne vodi se računa o tijelu i zdravlju. Dok bi se korištenjem ovakve aplikacije stvorila zdrava navika za vođenje računa o onome što se svakodnevno radi i čime se hrani.

LITERATURA

[1] Damir Kirasić, XML tehnologija i primjena u sustavima procesne informatike [online], IEEE, dostupno na: https://www.ieee.hr/_download/repository/mipro_xml_tekst.pdf [08.09.2019.]

[2] Cory Schmidt, What is Android ? [online], AndroidPIT, dostupno na: https://www.androidpit.com/what-is-android [08.09.2019.]

[3] Skupina autora, About Android [online], Android Developers, dostupno na: https://developer.android.com/about [08.09.2019.]

[4] Skupina autora, Meet Android Studio [online], Android Developers, dostupno na: https://developer.android.com/studio/intro [08.09.2019.]

[5] Skupina autora, Android Studio [online], Android Developers, dostupno na: https://developer.android.com/studio/index.html [08.09.2019.]

[6] Skupina autora, How many Java developers are there in the world ? [online], Plumbr, dostupno na: https://plumbr.io/blog/java/how-many-java-developers-in-the-world [08.09.2019.]

[7] Oracle Corporation, What is Java technology and why do I need it ? [online], Oracle Corporation, dostupno na: https://www.java.com/en/download/faq/whatis_java.xml [08.09.2019.]

[8] Skupina autora, XML (Extensible Markup Language) [online], TechTarget, dostupno na: https://whatis.techtarget.com/definition/XML-Extensible-Markup-Language [08.09.2019.]

[9] IBM, Introduction to XML [online], IBM, dostupno na: https://www.ibm.com/developerworks/xml/tutorials/xmlintro/xmlintro.html [08.09.2019.]

[10] Goran Rudman, Tablica prehrambene vrijednosti namirnica [online], Fitness, dostupno na: https://www.fitness.com.hr/prehrana/nutricionizam/Tablica-kalorija.aspx [08.09.2019.]

[11] Skupina autora, Body Fat Calculator [online], Calculator.net, dostupno na: https://www.calculator.net/body-fat-calculator.html [08.09.2019.]

[12] FluidUI, Mobile Prototrypes [online], FluidUI, dostupno na: https://www.fluidui.com/ [08.09.2019.]

SAŽETAK

Naslov: Android aplikacija za praćenje dnevnog treninga

Zadatak i cilj ovog završnog rada bio je kreirati Android aplikaciju za praćenje dnevnog treninga. Aplikacija je napravljena da omogućava unos informacija o korisniku aplikacije, njegova fizička mjerenja kao što su visina, masa, obujam ključnih dijelova tijela itd. Implementirana je mogućnost unosa prehrane tj. praćenja unosa nutritivnih vrijednosti na dnevnoj bazi. Tako da korisnik može dodavati namirnice po želji i kreirati obroke koji se poslije spremaju u bazu podataka i kojima se u svakom trenutku može pristupiti i vidjeti unesenu prehranu za određeni dan. Aplikacija ima mogućnost praćenja vježbi na način da korisnik izabere vježbu te ispuni potrebne informacije o seriji koju je izvodio. Kompletna aplikacija je izrađena uz pomoć Android studija te tehnologija koje on sadrži, implementacija i funkcionalnost aplikacije je pisana u Java programskom jeziku dok je dizajn i izgled pisan u XML opisnom jeziku. Rezultat ovog završnog rada je mobilna aplikacija koja je u potpunosti funkcionalna i ima sve mogućnosti i ispunjava sve tražene zahtjeve.

Ključne riječi: Android, Baza podataka, Java, vježbanje, prehrana

ABSTRACT

Title: Android application for tracking daily training

The task and objective of this final thesis was to create an Android application for tracking daily workouts and food/beverage intake. The application is designed to allow user's information to be entered. For example; his/her physical measurements such as height, weight, circumference of key body parts, etc. This application has the ability to track exercises in such a way that the user selects the exercise and fills in the necessary information about the sets he/she performed. In addition to the application tracking daily workouts, it was also designed to monitor dietary intake on a daily basis. Users have the ability to add their daily food consumption and save that information in the application database, which can be accessed at any time to view the diet entered for any given day. The complete application was created with the help of Android Studio and the technologies it contains. Implementations and functionalities of the application are written in Java programming language while the designs and layouts are written in XML descriptive language. The result of this final work is a mobile application that is fully functional and has all capabilities and meets all the required requirements.

Keywords: Android, Database, Diet, Java, Workout

ŽIVOTOPIS

Denis Ivanović rođen je 28.8.1998. godine u Požegi. Osnovnu školu Braće Radić pohađao je u Grebnica. Od 2012. do 2016. godine pohađao je srednju školu u Školskom centru fra Martina Nedića u Orašju te stiče zvanje Ekonomskog tehničara. Od 2016. godine pohađa preddiplomski stručni studij Elektrotehnike smjer Informatika na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. Tijekom pohađanja fakulteta stiče znanja iz područja programskog inženjerstva, koja se odnose najviše na programiranje s programskim jezicima Java, C i Python. Također na fakultetu pohađa predmet „*Razvoj mobilnih aplikacija*“ na kojem stiče znanja iz područja Androida, te same izrade Android mobilnih aplikacija u Android studiju.

_____

Denis Ivanović

PRILOZI

Projektna mapa s izvornim kôdom i dokumentima u word i pdf formatu nalazi se na optičkom disku koji je priložen uz ispisanu verziju završnog rada.

ID slučaja	UC1
Ime	Korisnik kreira obrok
Opis	Korisnik kreira obrok od prethodno unesenih namirnica
Preduvjet	Prethodno dodane namirnice
Scenarij	1. Korisnik odabire datum 2. Korisnik pritisne tipku „Unesi prehranu“ 3. Korisnika se prebacuje na zaslon za dodavanje obroka 4. Korisnik pritisne tipku „Kreiraj obrok“ 5. Korisnik se prebacuje na zaslon za odabir namirnica 6. Korisnik označuje potvrdne okvire pored namirnica koje želi u obroku 7. Korisnik pritisne tipku „Potvrdi“ 8. Korisnik se prebacuje na zaslon za unos mase za pojedinu namirnicu 9. Korisnik pored pojedine namirnice unosi masu za tu namirnicu u gramima 10. Korisnik dodaje obrok klikom na tipku „Dodaj obrok“

P.4.1. *Tablica koja opisuje UC1 slučaj korištenja.*

ID slučaja	UC2
Ime	Korisnik unosi namirnicu
Opis	Korisnik unosi namirnicu sa njenim nutritivnim vrijednostima
Preduvjet	Nema
Scenarij	1. Korisnik odabire datum 2. Korisnik klikne na tipku „Unesite prehranu“ 3. Korisnika se prebacuje na zaslon za dodavanje obroka 4. Korisnik klikne na tipku „Dodaj namirnicu“ 5. Korisnika se prebacuje na zaslon za dodavanje namirnica 6. Korisnik unosi ime namirnice 7. Korisnik unosi nutritivnu vrijednost kalorija jela na 100 grama 8. Korisnik unosi nutritivnu vrijednost proteina jela na 100 grama 9. Korisnik unosi nutritivnu vrijednost ugljikohidrata jela na 100 grama 10. Korisnik unosi nutritivnu vrijednost masti jela na 100 grama 11. Korisnik klikne na tipku „Dodaj jelo“

P.4.2. *Tablica koja opisuje UC2 slučaj korištenja.*

ID slučaja	UC3
Ime	Korisnik unosi vježbu
Opis	Korisnik unosi seriju za određenu vježbu
Preduvjet	Nema
Scenarij	1. Korisnik odabire datum 2. Korisnik klikne na tipku „Unesite vježbe“ 3. Korisnika se prebacuje na zaslon za dodavanje vježbi 4. Korisnik odabire vježbu iz padajućeg izbornika 5. Korisnik unosi broj ponavljanja 6. Korisnik unosi opterećenje 7. Korisnik unosi vrijeme odmora u sekundama 8. Korisnik klikne na tipku „Dodaj vježbu“

P.4.3. *Tablica koja opisuje UC3 slučaj korištenja.*

ID slučaja	UC4
Ime	Korisnik unosi mjerenje
Opis	Korisnik unosi mjerenje fizičkih proporcija
Preduvjet	Nema
Scenarij	1. Korisnik odabire datum 2. Korisnik klikne na tipku „Unesite mjerenje“ 3. Korisnika se prebacuje na zaslon za dodavanje mjerenja 4. Korisnik unosi masu 5. Korisnik unosi visinu 6. Korisnik unosi obujam vrata 7. Korisnik unosi obujam bicepsa 8. Korisnik unosi obujam prsa 9. Korisnik unosi obujam struka 10. Korisnik unosi obujam bokova 11. Korisnik unosi obujam bedara 12. Korisnik unosi obujam listova 13. Korisnik klikne na tipku „Dodaj mjerenje“

P.4.4. *Tablica koja opisuje UC4 slučaj korištenja.*