Izrada mobilne aplikacije za operacijski sustav Andorid na studijskom slučaju fitnes aplikacije

Ban, Stipe

VELEUČILIŠTE U ŠIBENIKU
ODJEL STUDIJA RAČUNARSTVA
PREDDIPLOMSKI STRUČNI STUDIJ

Stipe Ban

Izrada mobilne aplikacije za operacijski sustav Android na studijskom slučaju fitnes aplikacije

Završni rad

Šibenik, 2024 godina.

VELEUČILIŠTE U ŠIBENIKU
ODJEL STUDIJA RAČUNARSTVA
PREDDIPLOMSKI STRUČNI STUDIJ

Stipe Ban

Izrada mobilne aplikacije za operacijski sustav Android na studijskom slučaju fitnes aplikacije

Završni rad

Kolegij: Razvoj mobilnih aplikacija

Mentor: Marko Pavelić, mag. ing. inf. et comm. techn., pred.

Student: Stipe Ban

Matični broj studenta: 1219063975

Šibenik, rujan 2024.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, Stipe Bun, student/ica Veleučilišta u Šibeniku, JMBAG 1219063975 izjavljujem pod materijalnom i kaznenom odgovornošću i svojim potpisom potvrđujem da je moj završni/diplomski rad na stručnom prijediplomskom / stručnom diplomskom studiju Poslovna informatika pod naslovom: Izrada mobilne aplikacije za operacijski sustav Android na studijskom slučaju fitness aplikacije isključivo rezultat mog vlastitog rada koji se temelji na mojim istraživanjima i oslanja se na objavljenu literaturu, a što pokazuju korištene bilješke i bibliografija.

Izjavljujem da nijedan dio rada nije napisan na nedozvoljen način, odnosno da je prepisan iz necitiranog rada, te da nijedan dio rada ne krši bilo čija autorska prava.

Izjavljujem, također, da nijedan dio rada nije iskorišten za bilo koji drugi rad u bilo kojoj drugoj visokoškolskoj, znanstvenoj ili obrazovnoj ustanovi.

U Šibeniku, 30.9.2024

Student/ica:

Stipe Bun

TEMELJNA DOKUMENTACIJSKA KARTICA

Veleučilište u Šibeniku
Odjel studija računarstva
Preddiplomski stručni studij

Završni rad

Izrada mobilne aplikacije za operacijski sustav Android na studijskom slučaju fitnes aplikacije

STIPE BAN
sban@vus.hr

U ovom završnom radu opisan je operacijski sustav Android, od njegove povijesti pa sve do danas, njegov razvoj i arhitektura operacijskog sustava. Također opisano je razvojno okruženje Android Studio i programski jezik Java u kojem je napravljena aplikacija za ova završni rad. Opisan je sam proces izrade mobilne aplikacije i njene funkcionalnosti u obliku UML dijagrama. Aplikacija se sastoji od tri baze podataka, od kojih je jedna za registraciju/prijavu korisnika, druga za upis vježbi tog korisnika i treća za upis tjelesne težine korisnika kako bi pratio svoj put do ostvarenja svojih ciljeva u fitnesu.

(35 stranica / 12 slika / 8 literaturnih navoda / jezik izvornika: hrvatski)
Rad je pohranjen u digitalnom repozitoriju Knjižnice Veleučilišta u Šibeniku
Ključne riječi: Aplikacija, Java, Android
Mentor: Marko Pavelić, mag. ing. inf. et comm. techn., pred.
Rad je prihvaćen za obranu dana:

BASIC DOCUMENTATION CARD

Polytechnic of Šibenik Batchelor

Department of Computer Studies

Professional Undergraduate

Creation of a mobile application for the Android operating system based on a case study of a fitness application

STIPE BAN

sban@vus.hr

This undergraduate thesis describes the Android operating system, from its history to the present day, including its development and architecture. It also describes the development environment Android Studio and the Java programming language used to create the application for this undergraduate thesis. The process of creating the mobile application and its functionalities are illustrated with UML diagrams. The application consists of three databases: one for registration/login, second one inserting users workouts and the third one for inserting users body weight to help track their progress toward fitness goals.

(35 pages / 12 figures / 8 references / original in Croatian language)

Thesis deposited in Polytechnic of Šibenik Library digital repository

Keywords: Application, Android, Java

Supervisor: Marko Pavelić, mag. ing. inf. et comm. techn., pred.

Paper accepted:

SADRŽAJ

1. UVOD

U suvremenom načinu života, sve više ljudi posvećuje pažnju zdravlju i fizičkoj aktivnosti. Tehnološki napredak omogućio je razvoj brojnih digitalnih alata koji pomažu korisnicima da jednostavnije prate i poboljšavaju svoje fitness navike. Jedno od najpopularnijih rješenja u ovom području su mobilne aplikacije posvećene fitnessu, koje omogućuju korisnicima praćenje fizičke aktivnosti, postavljanje ciljeva, praćenje napretka te dobivanje personaliziranih savjeta. Cilj ovog završnog rada je prikazati razvoj fitness aplikacije za mobilne uređaje na Android platformi koristeći programski jezik Java i razvojno okruženje Android Studio. Aplikacija omogućuje korisnicima vođenje evidencije o treninzima, praćenje rezultata. Korištenje Jave u razvoju pruža aplikaciji visoku fleksibilnost, što omogućava daljnje nadogradnje i prilagodbe u skladu s potrebama korisnika. Rad će obuhvatiti detaljnu analizu ključnih aspekata razvoja Android aplikacije, strukture aplikacije, pohrane podataka o korisnicima i vježbama u bazu podataka, kao i integraciju softverskih alata i funkcionalnost koje korisnicima pomažu da prate svoj napredak.

2. OPERACIJSKI SUSTAV ANDROID

Operacijski sustav Android je sustav otvorenog koda koji je izrađen na *Linux* jezgri operacijskog sustava (engl.*kernel*). Prvenstveno je napravljen za pametne mobilne uređaje. Razvijen je od strane *Open Handset Alliance*, a njegova najkorištenija verzija je razvijena od strane *Googlea*. Prvi put je predstavljen u studenome 2007. godine, a prvi komercijalni uređaj na kojem je bio operacijski sustav Android lansiran je u rujnu 2008. godine. *Android Open Source Project (AOSP)* je besplatni softver otvorenog koda koji je licenciran pod *Apache* licencom. Iako, većina uređaja koristi Gooleovu verziju Android operacijskog sustava, koja dolazi s unaprijed instaliranim softverom zatvorenog koda, posebno *Google Mobile Services (GMS). Google Mobile Services (GMS)* uključuje osnovne aplikacije poput *Google Chromea, Google Play* trgovina i platformu *Google Play Services.* Iako je *AOSP* besplatan, naziv „ *Android* " i njegov logo zaštitni su znakovi *Googlea,* koji postavljaju pravila o upotrebi *Android* brendiranja na „necertificiranim" uređajima izvan njihovog ekosustava. Više od 70% *Android* uređaja koristi *Googleov* ekosustav, dok su alternativne verzije poput *Fire OS-a (Amazon)* i *ColorOS-a (Oppo).* Operacijski sustav *Android* koristi se i na pametnim uređajima poput televizora i satova. *Android* je od 2011. godine najprodavaniji operacijski sustav na pametnim mobilnim uređajima, a od 2013. godine na tabletima. U svibnju 2021. godine, imao je preko tri milijarde aktivnih korisnika mjesečno. Trenutno *Google Play Store* ima 1,7 milijuna dostupnih aplikacija, što je pad u odnosu na nekadašnjih 3 milijuna aplikacija. Android 14, najnovija verzija, objavljena je 4. listopada 2023. godine.

2.1. Povijest operacijskog sustava Android

Android Inc. Osnovali su *Andy Rubin, Rich Miner, Nick Sears* i *Chris White* u Palo Alto, Kalifornija, u listopadu 2003. godine. *Andy Rubin* je opisao projekt *Android* kao projekt s ogromnim potencijalom za razvoj pametnih mobilnih uređaja. U početku tvrtka je planirala razviti operativni sustav za digitalne kamere, što su i predstavili investitorima u travnju 2004. godine. Ubrzo su zaključili da tržište digitalnih kamera nije dovoljno veliko za njihove ciljeve, pa su se nakon par mjeseci preusmjerili na operativni sustav za pametne mobilne uređaje.U početku tvrtka je imala problema prikupljanjem investitora i bili su suočeni s izbacivanjem iz poslovnog prostora. Međutim, Steve Perlman, Rubinov bliski prijatelj donio je 10.000 dolara koji su iskoristili kao početni kapital. Perlman je odbio udio u njihovoj tvrtki i izjavio je kako

je on to učinio jer je vjerovao u njihov projekt i jer je htio pomoći svom prijatelju Andyju. Godine 2005., Rubin je započeo pregovore s tvrtkama *Samsung* i *HTC*. Nedugo nakon tih pregovora *Google* je kupio njihovu tvrtku za najmanje 50 milijuna dolara u srpnju 2005. godine. Ključni članovi tima iz tvrtke *Android Inc.* Prešli su u *Google* kao dio akvizicije. U *Googleu,* tim koj je vodio Rubin razvio je platformu za mobilne uređaje temeljenu na *Linux* kernelu. *Google* je ponudio tu platformu proizvođačima mobilnih uređaja s obećanjem fleksibilnog i nadogradivog operacijskog sustava. Dolaskom *Appelovog iPhonea* 2007. godine prisilio je *Google* da preusjmeri razvoj na podršku za ekrane za dodir. Prvi *Android* uređaj, *HTC Dream,* lansiran je 23. rujna 2008. godine i označio je početak nove ere pametnih uređaja. Od 2008. godine, *Android* je doživio brojne nadogradnje kooje su potpuno poboljšale operativni sustav i popravile brojne greške iz prošlih verzija. Svaka glavna verzija operacijskog sustava *Android* nosila je ime prema slatkišu po abecednom redu. Prve verzije operacijskog sustava *Android* imale su ime *CupCake, Donut, Eclair* i *Froyo.* Tijekom najave verzije *KitKata* 2013. godine *Google* je izjavio da uređaji čine naš život slađima pa da svaka verzija operacijskog sustava *Android* doviva ime po desertu. *Google* je 2010.godine pokrenuo seriju *Nexus* seriju mobilnih uređaja kako bi predstavio nove verzije operacijskog sustava *Android*. Na konferenciji 2013. godine, *Google* je najavio verziju *Samsung Galaxy S4* pametnog uređaja koja je umjesto *Samsungove* verzije operacijskog sustava koristila „čisti Android“. Ovaj mobilni uređaj označio je početak *Google Play Edition* programa, koji su kasnije slijedili drugi uređaji. S *Androidom 4.4, KitKat,* pristup pisanju na *MicroSD* memorijske kartice je bio ograničen. Pisanje je bilo moguće samo unutar posebnih direktorija s nazivima paketa *Android/dana/.* Pisanje je vraćeno s verzijom *Androida 5 Lollipop.* U svibnju 2019. godine operacijski sustav našao se u središtu rata između Kine i Sjedinjenih Američkih Država, zbog *Huawei-a* koji je ovisio o njihovoj platformi *Android.* Objavljeno je da će verzija operacijskog sustava *Android Q* biti preimenovana u *Android 10* čime se prekinuo niz imenovanja po desertima.

2.2. Arhitektura operacijskog sustava *Android*

Operacijski sustav *Android* baziran je na *Linux 2.6* jezgri i napisan je u programskom jeziku *C/C++*. Zbog otvorenog programskog koda, aplikacije imaju mogućnost komuniciranja i pokretanja drugih aplikacija putem *middlewarea.* Iako su programski jezici *C i C++* primjenjivi za radno okruženje (engl. *framework*), većina aplikacija napisana je u programskom

jeziku *Javi* rabeći *Android Software Development Kit (SDK).* Moguća je i izrada aplikacija s programskim jezikom *C/C++*, ali u tom slučaju upotrebljava se *Android Native Code Development Kit (NDK).* Ovim postupkom omogućuje se mnogo bolje raspolaganje resurima i uporaba knjižnica programa iz jezgre. Složenije je pisanje same aplikacije, ali sama aplikacija je brža i do 10 puta. Arhitektura operacijskog sustava *Android* može se promatrati kao programski stog s nekoliko slojeva. Na dnu tog stoga je *Linux 2.6* jezgra koja sadrži *drivere,* od kojih su najvažniji oni za međuprocesnu komunikaciju (*IPC*) i upravljanje napajanjem (engl. *Power Management*). Iznad jezgre su knjižice napisane u programskom jeziku *C/C++:*

- *Surface Manager* – knjižnica koja nadzire iscrtavanje grafičkog sučelja
- *OpenGL | ES* – knjižnica za skopovsko ubrzavanje 3D prikaza i za visoku optimiziranu 3D softversku rasterizaciju
- *SGL – 2D* – knjižnica koja se upotorebljava za večinu aplikacija
- *Media Framework* – knjižnica temeljena na *OpenCORE* koja podržava snimanje i reproduciranje poznatih audio/video formata
- *FreeType* – kjižica namijenjena iscrtavanju fontova
- *SSL (Secure Sockets Layer)* – knjižnica za sigurnosnu komunikaciju putem interneta
- *SQLite* – knjižnica za upravljanje bazama podataka dostupna svim aplikacijama
- *WebKit – engine* za preglednike
- *Libc* – sistemska *C* knjižnica prilagođena za ugradbene sustave zasnovane na *Linux OS-u* [6].

2.3. Razvojno okruženje Android studio

Android Studio je službeno integrirano razvojno okruženje (*IDE*) za razvoj *Android* aplikacija, koje je razvila *Google-ova Android* ekipa. Pokrenut je 2013. godine i temelji se na *IntelliJ IDEA,* popularnom *Java IDE-u*. *Android Studio* pruža kompletan set alata za razvoj *Android* aplikacija, uključujući *editor* koda, alate za vizualni dizajn korisničkog sučelja (*UI*), emulator za testiranje aplikacija te napredne funkcije poput refaktorizacije i automatskog generiranja koda.

Neke od ključnih značajki *Android Studia* su:

- *Gradle* sistem za gradnju koji olakšava upravljanje zavisnostima i automatizira proces gradnje aplikacija
- *Emulator Android* uređaja za testiranje aplikacija na različitim verzijama Androida i

uređajima

- Profiler performansi za praćenje i optimizaciju upotrebe memorije, CPU-a, mrežnih resursa i baterije
- Podrška za *Kotlin* programski jezik, uz standardni *Java* programski jezik, čime se omogućuje jednostavniji i moderniji razvoj aplikacija

Razvojno okruženje *Android Studio* kontinuirano se ažurira i poboljšava, prateći najnovije verzije Android sustava i razvojne trendove, što ga čini ključnim alatom za sve Android programere [4].

Struktura projekta u razvojnom okruženju *Android Studia.* Svaki projekt u razvojnom okruženju *Android Studio* sadrži jedan ili više modula s datotekama izvornog koda i datotekama resursa. Tipovi modula uključuju:

- *Android App* module
- *Library* module
- *Google App Engine* module

Po zadanim postavkama, razvojno okruženje *Android Studio* prikazuje datoteke projekta u *Android Project View.* Ovaj prikaz organiziran je po modulima kako bi osigurao brzi pristup ključnim izvornim datotekama projekta. Svaki modul aplikacije sadrži:

- *Manifest* – sadrži *AndroidManifest.xml* datoteku
- *Java* – sadrži datoteke izvornog koda *Kotlin* i *Java*
- *Res* – sadrži sve nekodirane resurse [7].

3. PROGRAMSKI JEZIK JAVA

Java je programski jezik koji je razvio James Gosling zajedno s timom inženjera u tvrtki *Sun Microsystems*. Razvoj programskog jezika Jave započeo je 1991. godine kroz projekt pod nazivom " *Green Project* ". Prvotni cilj bio je prenosivi jezik koji je mogao raditi na raznim uređajima, poput pametnih televizora i kućnih uređaja. Jezik je prvotno nazvan *Oak,* ali je kasnije preimenovan u *Java*, prema sorti kave. Programski jezik *Java* je službeno predstavljena 1995. godine, u vrijeme kad su *web* preglednici postajali sve popularniji. U to vrijeme, programski jezik Java se istaknula svojom ključnom filozofijom *" Write Once, Run Anywhere " (WORA),* što znači da su program napisani u Javi mogli raditi na bilo kojoj platformi koja podržava *Java Virtual Machine (JVM),* bez potrebe za prilagodbama. Njezina prenosivost I sugurnosne značajke učinile su taj programski jezik izuzetno popularnom za razvoj web aplikacija I poslovnih sustava. Također, zahvaljujući objektno orijentiranom pristupu, podršci za višezadaćnost (*multitherading*) I mrežno programiranje, Java je postala jedan od najraširenijih I najutjecajnijih programskih jezika u svijetu te je I danas temelj mnogih tehnoloških sustava [5].

Programski jezik *Java* ima nekoliko ključnih značajki koje su je učinile jednim od najpopularnijih programskih jezika:

- **Platformska neovisnost:** Programski jezik *Java* slijedi princip (*WORA*). Programi se prevode u *bytecode*, koji može raditi na bilo kojem sustavu koji ima *Java Virtual Machine (JVM),* neovisno o operativnom sustavu ili hardveru.
- **Objektno orijentirana:** Programski jezik *Java* se temelji na objektno orijentiranom pristupu, što znači da sve u *Javi* predstavlja objekte. To omogućuje modularnost, ponovnu upotrebu koda I jednostavnije održavanje
- **Sigurnost:** *Java* ima ugrađene sigurnosne značajke koje štite od zlonamjernih aplikacija, uključujući upravljanje memorijom, sandbox I verifikaciju *bytecode-a*. Time se smanjuje mogućnost grešaka poput prepisivanja memorije ili zlonamjernog koda.
- **Automatsko upravljanje memorijom:** Java koristi *garbage collector* koji automatski oslobađa memoriju koju više ne koristi objekti, čime smanjuje rizik od curenja memorije I pogrešaka.
- **Multithreading:** Java omogućuje višezadaćnost (*multithreading*), što znači da se više zadataka može izvoditi paralelno unutar istog programa, što poboljšava performance I efikasnost.

- **Dinamičnost I prenosivost:** *Java* se dinamički povezuje s bibliotekama u vrijeme izvršavanja, što omogućava fleksibilnost I prilagodbu. Biblioteke I aplikacije napisane u programskom jeziku *Javi* mogu se lako prenositi na tazličite platforme.
- **Visoke performance:** Iako nije toliko brza kao jezici koji se prevode direktno u strojni kod (poput *C/C++), Java* koristi *Just-In-Time (JIT)* kompajler unutar *JVM*-a kako bi ubrzala izvršavanje programa.
- **Robusnost:** Programski jezik *Java* je dizajniran da bude pouzdan I robusan. Nudi provjere pogrešaka u vremenu kompilacije I izvršavanja te ima snažan sustav iznimaka (*Exception handling*) koji olakšava otkrivanje I rješavanje pogrešaka [2].

Programski jezik *Java* omogućava izradu aplikacija na razini poduzeća, poput mrežnih aplikacija, *web* aplikacija i pristupa bazama podataka. Zbog svoje kompatibilnosti s različitim platformama poznata je i u izradi mobilnih aplikacija i smatra se univerzalnim programskim jezikom.

- Razvoj mobilnih aplikacija

Razvoj mobilnih aplikacija jedno je od najpopularnijih područja primjene *Jave* s pojavom pametnih telefona. Najčešće korišteni mobilni operativni sustav temelji se na *Javi*. Više od 80% pametnih telefona koristi *Android*, što stvara veliku potražnju za tvrtkama koje razvijaju mobilne aplikacije koristeci *Javu*.

- Razvoj web aplikacija

Jedna od čestih primjena Jave u programiranju. Tvrtke za razvoj web aplikacija stvaraju softverske aplikacije koje rade na web poslužitelju, a korisnici pristupaju putem preglednika (*Google Chrome, Mozilla Firefox*). Jedne od najpoznatijih web aplikacija koje su napravljene koristeći *Javu* su *Spotify*(streaming glazbe) i *Twitter* (sadašnje X).

- Razvoj softvera za poduzeća

Java je čest izbor za razvoj softverskih rješenja. Softver za poduzeća pomaže tvrtkama u ostvarivanju njihovih ciljeva te obuhvaća sustave za upravljanje odnosima s klijentima (*CRM*),

sustave za ljudske resurse (*HR*) i sustave za upravljanje opskrbnim lancem (*SCM*). Najpoznatiji *CRM* sustav napravljen u javi je *Salesforce*.

- Aplikacije za računala

Omogućuje izradu aplikacija za stolna računala. Aplikacija je softverski program koji se instalira na računalo i radi unutar operativnog sustava, bez potrebe za internetskom vezom. Alat za 3D vizualizaciju geografskih analiza, *Nasa World Wind*, razvijen je u Javi. Taj alat otvorenog koda koristi se za praćenje kretanja vozila, vizualizaciju gradova i analizu vremenskih obrazaca.

- Igre i engine

Koristi se i za razvoj videoigara. Neke od najpoznatijih igarica napravljenih u Javi su *Minecraft* i *Clash of Clans*. Također koristi se i za izradu *Game Engine*-a, softvera koji olakšava izradu videoigara pružajući grafićke i audio alate, kao i druge potrebne funkcionalnosti.

- Ugrađeni sustavi

Java se koristi i za razvoj ugrađenih sustava, koji su specijalizirani računalni sustavi s određenom funkcijom. Obično se nalaze u uređajima poput automobila, televizora i zrakoplova [1].

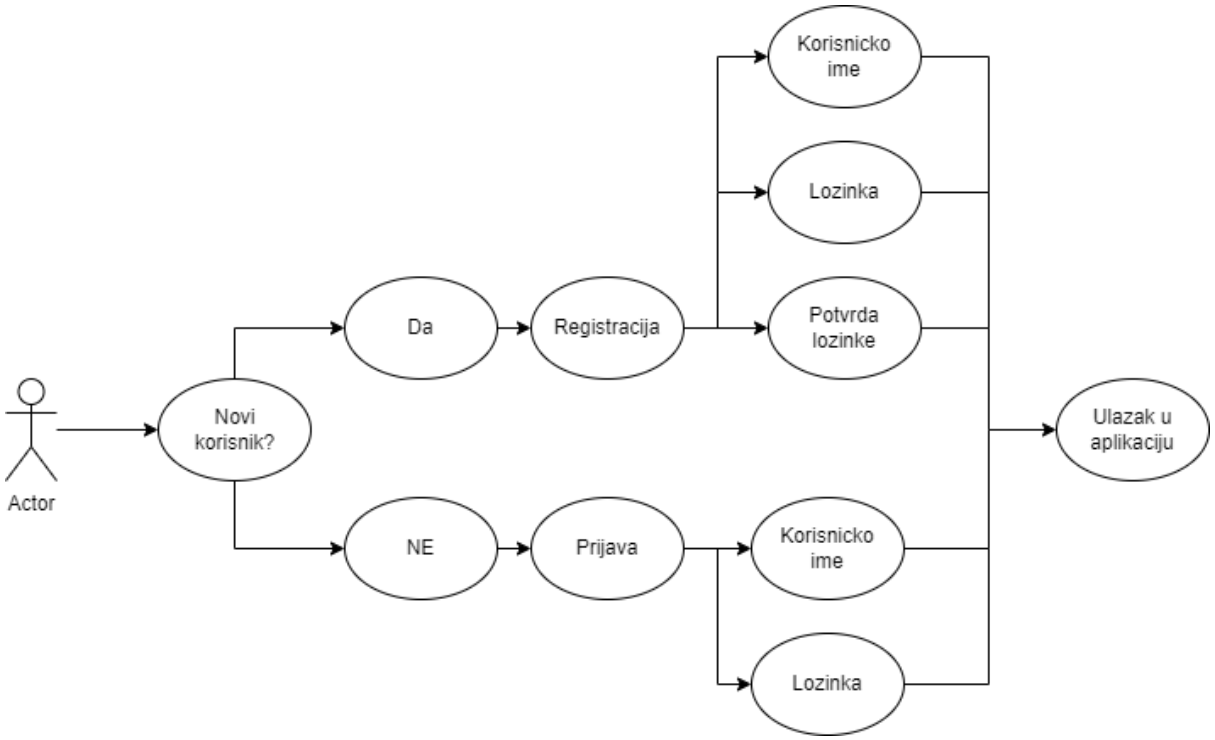
4. ARHITEKTURA SUSTAVA

UML (engl. *Unified Modeling Language*) je normirani vizualni jezik koji se koristi za modeliranje programske potpore. Pomoću njega izrađuju se i UML dijagrami koji se koriste za vizualno prikazivanje oblikovanja, ponašanja složenih programskih sustava i njihove arhitekture. Pomažu u održavanju jasnoće i dosljednosti u dokumentaciji projekata [8].

4.1. UML dijagrami

4.1.1. Dijagram slučaja uporabe prijave i registracije

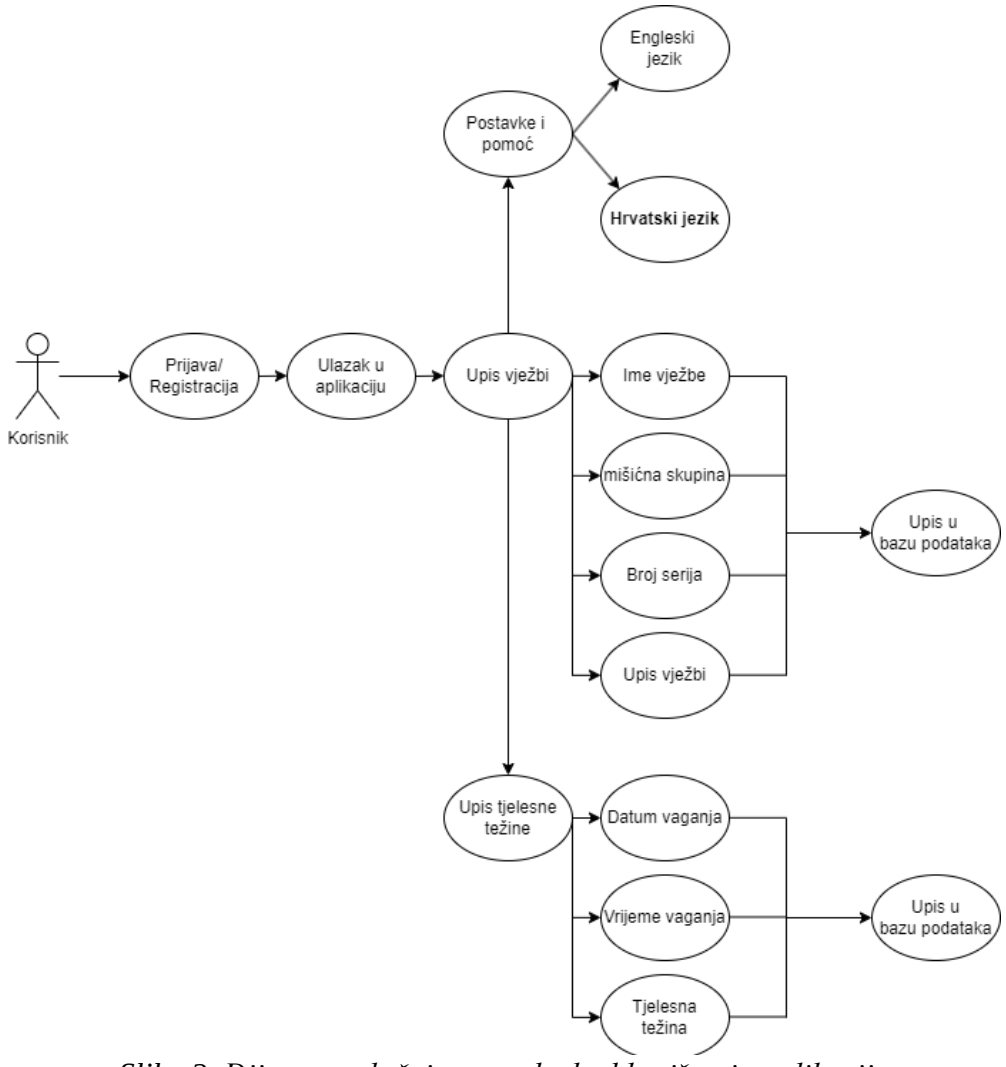
Na Slici 1. prikazan je dijagram slučaja uporabe prijave i registracije. Korisnik ulazi u aplikaciju i ako je prvi put u aplikaciji, mora napraviti korisnički račun za ulazak u aplikaciju. Mora odabrati unikatno korisničko ime, svoju lozinku i potvrditi svoju lozinku. Ako je registracija uspješna, korisnik ima pristup aplikaciji. U slučaju da korisnik ima korisnički račun u aplikaciji, treba samo upisati svoje korisničko ime i odgovarajuću lozinku,ako su korisničko ime i lozinka točni, korisnik ima pristup aplikaciji.



Slika 1. Dijagram slučaja uporabe prijave i registracije

4.1.2. Dijagram slučaja uporabe kod korištenja aplikacije

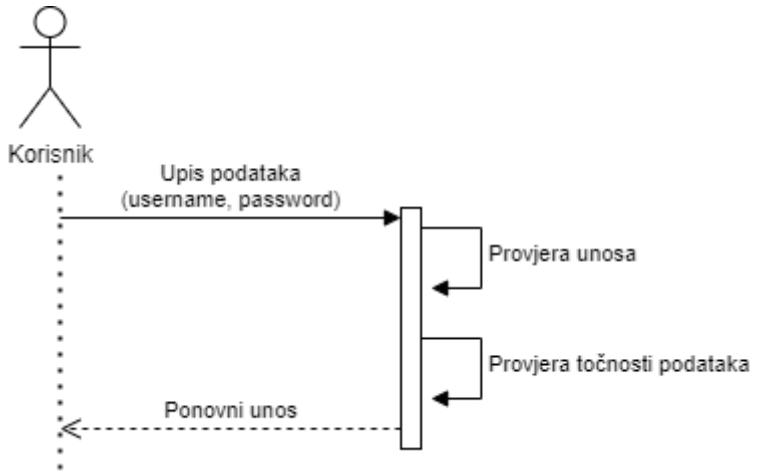
Na slici 2. prikazaj je dijagram slučaja uporabe (engl. *UseCase*) kod korištenja aplikacije. Opisuje što se korisnik može raditi u aplikaciji. Nakon prijave ili registracije, korisnik ulazi u aplikaciju, prva stranica koja se otvara je ona za upis vježbi. Iz stranice za upis vježbi, korisnik bira hoće li ići na stranicu za upis tjelesne težine ili na stranicu gdje su postavke i pomoć. Na stranici za upis vježbi korisnik mora upisati ime vježbe, mišiće koji radu s tom vježbom, broje serija i broj ponavljanja koliko će korisnik odraditi s kojom vježbom. Na stranici za unos tjelesne težine, korisnik upisuje datum vaganja, vrijeme u kojem se vagao i tjelesnu težinu koju je imao u to vrijeme. Stranica za postavke i pomoć služi korisniku za odabir jezika aplikacije. Korisnik može birati engleski ili hrvatski jezik, također mu je objašnjeno što treba upisati u koje polje na kojoj stranici.



Slika 2. Dijagram slučaja uporabe kod korištenja aplikacije

4.1.3. Sekvencijski dijagrami prijave u aplikaciju

Na Slici 3. prikazan je sekvencijski dijagram kod prijave postojećeg korisnika. Korisnik upisuje svoje korisničko ime i lozinku. Podaci se provjeravaju u bazi podataka, kad su podaci netočni, korisniku dolazi poruka da ponovi unos podataka.



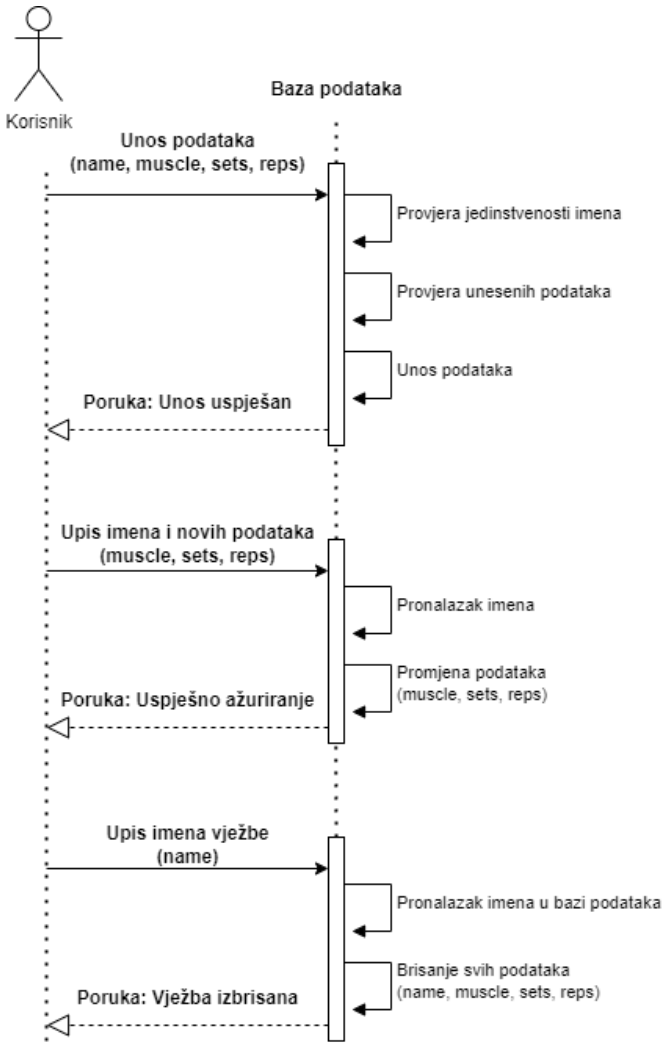
Slika 3. Sekvencijski dijagram kod pogrešnog unosa podataka pri prijavi

Na Slici 4. prikazan je sekvencijski dijagram kod prijave postojećeg korisnika. Korisnik upisuje svoje korisničko ime i lozinku. Podaci se provjeravaju u bazi podataka, kad se provjere korisniku dolazi poruka da je prijava uspješna i prosljeđuje ga u aplikaciju.

Slika 4. Sekvencijski dijagram kod točnog unosa podataka pri prijavi

4.1.4. Sekvencijski dijagram kod unosa, ažuriranja i brisanja podataka

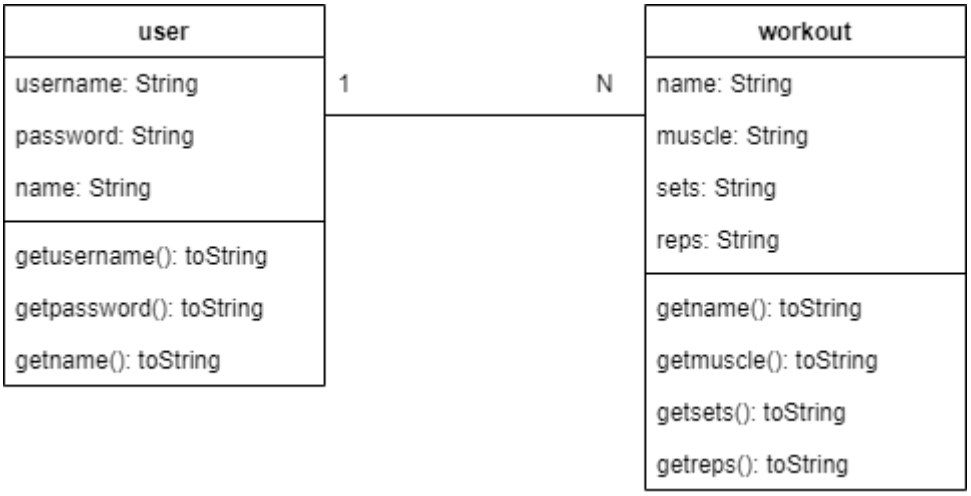
Na Slici 5. prikazan je sekvencijski dijagram unosa, ažuriranja i brisanja podataka upisanih vježbi. Korisnik unosi podatke o vježbi, provjerava se jedinstvenost imena vježbe, nakon provjere upisuju se podaci u bazu podataka. Korisniku dolazi poruka da je unos bio uspješan. Za ažuriranje podataka, korisnik upisuje ime vježbe čije podatke želi ažurirati i upisuje nove podatke(*muscle, sets, reps*). Provjerava se postoji li upisano to ime vježbe, ako postoji onda se mijenjaju podaci te vježbe i korisniku dolazi poruka da je ažuriranje uspješno. Za brisanje vježe korisnik samo upisuje ime vježbe koju želi izbrisati. U bazi podataka se provjerava postoji li zapisano ime te vježbe u bazi podataka, ako postoji onda se brišu svi podaci o toj vježbi. Nakon brisanja korisniku dolazi poruka da je brisanje uspješno.



Slika 5. Sekvencijski dijagram upisa, ažurirnja i brisanja vježbi iz baze podataka

4.1.5. Dijagram klasa

Na Slici 6. prikazan je dijagrama klasa. Prikazane su dvije klase, jedna se zove korisnik (engl. *user*), a druga se zove vježba (engl. *workout*). Klasa *user* ima entitete *username* koji je ujedno i primarni ključ, *password, name,* a klasa *workout* ima entitete *name, muscle, sets i reps*. Entitet *name* je primarni ključ klase *workout* i strani ključ klase *users.*



Slika 6. Dijagram klasa user i workout

4.2. Baza podataka za login i registraciju

U ovom projektu korištena je već ugrađena baza podataka *SQLite*. Ona ne zahtjeva dodatnu instalaciju servera, što je čini praktičnu za aplikaciju koja treba pohraniti podatke lokalno.

Dio koda koji predstavlja izradu baze podataka *usersdb.db* u kojoj se nalazi tablica *users* koja se sastoji od entiteta *username* i entiteta *password.* Entitet username je primarni ključ tablice. Tablica se koristi za upis podataka novih korisnika i za prijavu već postojećih korisnika.

```
public class DBreg extends SQLiteOpenHelper {

   public static final String databaseName = "userdb.db";

    public DBreg(@Nullable Context context) {super(context, "userdb.db",
null, 1);}
    @Override
    public void onCreate(SQLiteDatabase db) {db.execSQL("create Table
users(username TEXT primary key, password TEXT)");}

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {db.execSQL("drop Table if exists users");}
```

Kod 1. Izrada baze podataka usersdb i tablice users

Metoda nazvana *insertdata()* koju koristimo za unos podataka u tablicu *users*. Prva linija koda u metodi *insertdata()* otvara bazu podataka i omogućuje nam da u nju dodajemo nove podatke. Linija koda *ContentValues* kreira objekt *contentValues* i u njega se dodaju vrijednosti *username* i *password*. Kasnije se vrijednosti iz *contentValues* upisuju u bazu podataka. Ako je upis bio nuspješan metoda vraća (-1).

```
public Boolean insertdata (String username, String password)
{
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues contentValues = new ContentValues();
    contentValues.put("username",username);
    contentValues.put("password",password);
    long result = db.insert("users",null, contentValues);
    if (result == -1)
    {
        return false;
    }else return true;}
```

Kod 2. Metoda za unos u bazu podataka usersdb

4.2.1. Provjera podataka u bazi podataka *usersdb*

Kod za dvije metode *checkusername()* i *checkuserpass()*. Metoda *checkusername()* provjerava postoji li u bazi podataka onaj *username* koji korisnik upisao. U slučaju da taj *username* već postoji tj. Ako je već upisan u bazu podataka *cursor* vraća *true*, a ako ne postoji metoda vraća *false*. Metoda *checkuserpass()* provjerava postoji li ista kombinacija *username* i *password* upisana u bazi podataka. Nakon provjere, ako kombinacija postoji metoda vraća *true*, u suprotnom vraća *false*.

```
public Boolean checkeusername (String username)
{
    SQLiteDatabase db =this.getWritableDatabase();
    Cursor cursor = db.rawQuery("Select * from users where username =
?",new String[]{username});
    if(cursor.getCount()>0)
    {return true;}
    else {return false;}
}
public Boolean checkuserpass (String username, String password)
{
    SQLiteDatabase db =this.getWritableDatabase();
    Cursor cursor = db.rawQuery("Select * from users where username = ? and
password=?",new String[]{username, password});
    if(cursor.getCount()>0)
    {return true;}
    else {return false;}
}}
```

Kod 3. Metode za provjeru podataka u bazi podataka usersdb

4.3. Baza podataka za upis vježbi

Dio koda koji predstavlja izradu baze podataka *Exercise.db* u kojoj se nalazi tablica *exertb* u kojoj ćemo unositi podatke vezane za vježbe. Tablica *exertb* sastoji se od od četiri entiteta. Entiteti u tablici su *name*, *muscle*, *sets* i *reps*. Entitet *name* predstavlja ime vježbe koju će korisnik upisati, entitet *muscle* predstavlja onu mišićnu skupinu koju će korisnik raditi s tom vježbom, entitet *sets* predstavlja broj serija koje će korisnik odraditi na toj vježbi i entitet *reps* predstavlja koliko ponavljanja je korisnik odradio po jednoj seriji.

```
public class DBHelper extends SQLiteOpenHelper {
    public DBHelper(Context context) {
        super(context, "Exercise.db", null, 1);
    }
    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("create Table exertb(name TEXT primary key,muscle TEXT,
sets TEXT, reps TEXT)");}
    @Override
    public void onUpgrade(SQLiteDatabase db, int i, int i1) {
        db.execSQL("drop Table if exists exertb");}
```

Kod 4. Izrada baze podataka Exercise i tablice exertb

4.3.1. Upis podataka u tablicu *exertb*

Metodu *insertuserdata()* koristimo za upis podataka u tablicu *exertb*. Prva linija koda u metodi otvara bazu podataka i omogućuje upis novih podataka u tablicu. Linija *ContentValues* kreira objekt *contentValues* u kojeg se dodaju vrijednosti *name*, *muscle*, *sets* i *reps*. Kasnije

vrijednosti iz objekta *contentValues* upisuju se u bazu podataka. Ako upis u bazu nije bio uspješan metoda vraća negativan broj (-1).

```
public Boolean insertuserdata (String name, String muscle, String sets,
String reps)
{
    SQLiteDatabase db=this.getWritableDatabase();
    ContentValues contentValues = new ContentValues();
    contentValues.put("Name", name);
    contentValues.put("Muscle", muscle);
    contentValues.put("Sets", sets);
    contentValues.put("Reps", reps);
    long result=db.insert("exertb", null, contentValues);
    if (result==-1){return false;}
    else {return true;}}
```

Kod 5. Unos podataka u tablicu exertb

4.3.2. Ažuriranje podataka iz tablice *exertb*

Metodu *updateuserdata()* koristimo za ažuriranje upisanih podataka u tablicu *exertb.* Kod otvara bazu podataka i priprema je za ažuriranje. Korisnik upisuje ime vježbe i nove podatke koje želi ažurirati. Provjeravaju se sve upisane vježbe u tablici *exertb.* Ako nađe vježbu s istim imenom onda ažurira ostale podatke te vježbe (*muscle, sets, reps*).

```
public Boolean updateuserdata (String name, String muscle, String sets,
String reps)
{
    SQLiteDatabase db=this.getWritableDatabase();
    ContentValues contentValues = new ContentValues();
    contentValues.put("Muscle", muscle);
    contentValues.put("Sets", sets);
    contentValues.put("Reps", reps);
    Cursor cursor = db.rawQuery("Select * from exertb where name=?", new
String[]{name});
    if(cursor.getCount()>0) {
        long result = db.update("exertb", contentValues, "name=?", new
String[]{name});
        if (result == -1) {
            return false;
        } else {
            return true;
        }}else {return false;}}
```

Kod 6. Ažuriranje podataka iz tablice exertb

4.3.3. Brisanje podataka iz tablice *exertb*

Metoda *deletedata()* nam služi za brisanje podataka iz tablice *exertb*. Korisnik upisuje ime vježbe koju želi izbrisati. Kod u metodi *deletedata()* provjerava postoji li ta vježba u tablici. U slučaju da postoji, vježba i svi njeni podaci se uklanjaju iz tablice.

```
public Boolean deletedata (String name)
{
    SQLiteDatabase db=this.getWritableDatabase();
    Cursor cursor = db.rawQuery("Select * from exertb where name=?", new
String[]{name});
    if(cursor.getCount()>0) {
        long result = db.delete("exertb", "name=?", new String[]{name});
        if (result == -1) {
            return false;
        } else {return true;}
    }else {return false;}}
```

Kod 7. Brisanje podataka iz tablice exertb

4.3.4. Pregled upisanih podataka iz tablice *exertb*

Kod iz metode *getdata()* nam omogućuje da čitamo sve upisane podatke u tablici *exertb*. Prva linija koda otvara bazu podataka i omogućuje čitanje iz nje. SQL upit "Select * from exertb" vraća sve podatke iz tablice *exertb*. Linija *return cursor;* vraća rezultat upita u obliku objekta *cursor*.

```
public Cursor getdata ()
{
    SQLiteDatabase db=this.getWritableDatabase();
    Cursor cursor = db.rawQuery("Select * from exertb", null);
    return cursor;
}}
```

Kod 8. Kod za pregled upisanih podataka iz tablice exertb

4.4. Baza podataka za upis tjelesne težine

Dio koda koji predstavlja izradu baze podataka *Weight.db* u kojoj se nalazi tablica *WeightDetails* u kojoj ćemo unositi podatke vezane za tjelesnu težinu. Tablica *WeightDetails* sastoji se od tri entiteta. Entiteti su *date, time* i *weight.* Entitet *date* predstavlja datum na koji se korisnik izvagao, entitet *time* predstavlja vrijeme kada se korisnik izvagao i entitet *weight* predstavlja tjelesnu težinu koju je korisnik imao u vrijeme vaganja.

```
public class DBHelper2 extends SQLiteOpenHelper {
    public DBHelper2(Context context) {
        super(context, "Weight.db" , null, 1);
    }
    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("create Table WeightDetails(date Text primary key,time
TEXT, weight TEXT)");
    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
        db.execSQL("drop Table if exists WeightDetails");
    }
```

Kod 9. Izrada baze podataka Weight.db i tablice WeightDetails

4.4.1. Upis podataka u tablicu *WeightDetails*

Metodu *insertdata()* koristimo za upis podataka u tablicu *WeightDetails*. Prva linija koda u metodi otvara bazu podataka i omogućuje upis novih podataka u tablicu. Linija *ContentValues* kreira objekt *contentValues* u kojeg se dodaju vrijednosti *date, time i weight.* Kasnije vrijednosti iz objekta *contentValues* upisuju se u bazu podataka. Ako upis u bazu nije bio uspješan metoda vraća negativan broj (-1).

```
public Boolean insertdata(String date,String time, String weight) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues contentValues = new ContentValues();
    contentValues.put("Date", date);
    contentValues.put("Time", time);
    contentValues.put("Weight", weight);
    long result = db.insert("WeightDetails", null, contentValues);
    if (result == -1) {
        return false;
    } else {return true;}}
```

Kod 10. Unos podataka o tjelesnoj težini

4.4.2. Ažuriranje podataka iz tablice *WeightDetails*

Metodu *updatedata()* koristimo za ažuriranje upisanih podataka u tablicu *WeightDetails.* Kod otvara bazu podataka i priprema je za ažuriranje. Korisnik upisuje datum vaganja koje želi promijeniti i nove podatke koje želi ažurirati. Provjeravaju se svi upisani datumi u tablici *WeightDetails.* Ako nađe isti datum u tablici kao onaj što je korisnik tražio, onda ažurira ostale podatke na taj datum (*time, weight*).

```
public Boolean updatedata (String date, String time, String weight)
{
    SQLiteDatabase db=this.getWritableDatabase();
    ContentValues contentValues = new ContentValues();
    contentValues.put("Time", time);
    contentValues.put("Weight", weight);
    Cursor cursor = db.rawQuery("Select * from WeightDetails where date=?",
new String[]{date});
    if(cursor.getCount()>0) {
        long result = db.update("WeightDetails", contentValues, "date=?",
new String[]{date});
        if (result == -1) {
            return false;
        } else {
            return true;
        }
    }else {return false;}}
```

Kod 11. Ažuriranje podataka tjelesne težine

4.4.3. Pregled upisanih tjelesnih težina iz tablice *WeightDetails*

Kod iz metode *getdata()* nam omogućuje da čitamo sve upisane podatke u tablici *WeightDetails*. Prva linija koda otvara bazu podataka i omogućuje čitanje iz nje. SQL upit "*Select * from WeightDetails*" vraća sve podatke iz tablice *WeightDetails.* Linija *return cursor;* vraća rezultat upita u obliku objekta *cursor.*

```
public Cursor getdataa ()
{
    SQLiteDatabase db=this.getWritableDatabase();
    Cursor cursor = db.rawQuery("Select * from WeightDetails", null);
    return cursor;}}
```

Kod 12. Kod za pregled upisanih podataka o tjelesnoj težini iz tablice WeightDetails

4.5. Studijski slučaj Android aplikacije – „FITNES APLIKACIJA“

Fitnes aplikacija napravljena je tako da se novi korisnik mora registrirati kako bi uopče koristio aplikaciju. Nakon registracije novog korisnika ili prijave već postojećeg korisnika, otvara se aktivnost za upis vježbe. Korisnik može napraviti svoj plan treninga i odrediti koliko će vježbi odraditi i koliko će svaka vježba imati serija i koliko će svaka serija imati ponavljanja. Ima mogućnost praćenja svoje tjelesne težine, također korisnik može birati hoće li mu aplikacija biti na hrvatskom ili engleskom jeziku.

4.5.1. Ekran za učitavanje

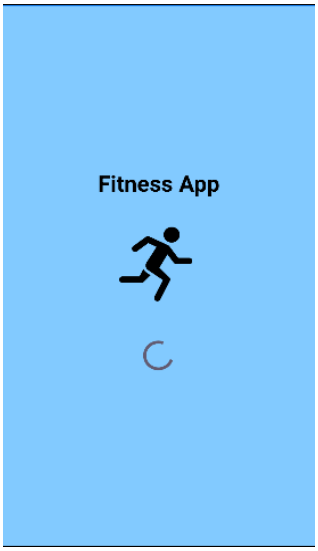
Aktivnost koja se pokreće pri pokretanju aplikacije. Nakon određenog vremena aktivnost se gasi i pokreće se aktivnost za registraciju korisnika.

Kod ekrana za učitavanje. Prva aktivnost koja se pokreće pri pokretanju aplikacije. Nakon pokretanja aktivnosti pojavljuje se logo aplikacije i tekst. Učitavanje aktivnosti traje 3000 milisekundi (3 sekunde), te nakon toga pokreće se aktivnost za registraciju novog korisnika ili *login* već postojećeg korisnika.

```
public class loadingscreen extends AppCompatActivity {

    Handler handler;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_loadingscreen);

        handler = new Handler();
        handler. postDelayed(new Runnable() {
            @Override
            public void run() {
                Intent intent = new Intent(loadingscreen.this,
MainActivity3.class);
                startActivity(intent);
                finish();
            }},3000);};}
```

Kod 13. Kod ekrana za učitavanje

Slika 1 prikazuje izgled ekrana za učitavanje. Ekran za učitavanje se sastoji od komponenti grafičkog sučelja: *TextView, ImageView I ProgressBar*. Tekst *Fitness App* crne je boje I ima veličinu fonta 30dp I podebljan je. Logo aplikacije je smješten po sredini ekrana I ispod njega nalazi se *ProgressBar* koji se učitava 3000 milisekundi.



Slika 7. Izgled ekrana za učitavanje

4.5.2. Registracija korisnika

Kod počinje postavljanjem listenera za klik na gumb (*Register*). Klikom na gumb (*Register*) izvršava se kod unutar metode *onClick().* Linije koda gdje je *String* služe za dohvaćanje vrijednosti koje su upisane u komponentu grafičkog sučelja *EditText* (*username, password* i *confirm password).* U slučaju da su polja ostala prazna tj. Da korisnik nije popunio sva polja dolazi poruka korisniku (*Fill all fields*) koja ga napominje da mora popuniti sva polja za registraciju. Ako su sva polja popunjena, onda se provjerava jesu li password i confirm password iste, ako nisu prikazuje se poruka (*Invalid Password*). Kod podudaranja passworda poziva se metoda *checkusername().* Ako korisničko ime već postoji, prikazuje se poruka *(Username already exists, Please login.),* ako ne postoji to korisničko ime, registracija se nastavlja. Kada je korisničko ime slobodno, poziva se metoda *insertdata()* koja dodaje korisničko ime i lozinku u bazu podataka, dolazi poruka (*Registration successfull*) i prelazi se na aktivnost (*Loading screen*), potom na aktivnost (*Add Workout*).

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

        String username = binding.username.getText().toString();
        String password = binding.password.getText().toString();
        String confirmpass = binding.confirmpass.getText().toString();
        if (username.equals("") || password.equals("") ||
confirmpass.equals(""))
            Toast.makeText(MainActivity3.this, "Fill all fields",
Toast.LENGTH_SHORT).show();
        else
        {
            if (password.equals(confirmpass)) {
                Boolean checkusername =
databasehelper.checkeusername(username);
                if (checkusername == false) {
                    Boolean insert = databasehelper.insertdata(username,
password);
                    if (insert == true) {
                        Toast.makeText(MainActivity3.this, "Register
Successfull", Toast.LENGTH_SHORT).show();
                        Intent intent = new Intent(getApplicationContext(),
MainActivityLS2.class);
                        startActivity(intent);
                    } else {Toast.makeText(MainActivity3.this, "Register
failed", Toast.LENGTH_SHORT).show();}
                } else {Toast.makeText(MainActivity3.this, "Username
already exists, Please login", Toast.LENGTH_SHORT).show();}
            } else {Toast.makeText(MainActivity3.this, "Invalid Password",
Toast.LENGTH_SHORT).show();}}}});
```

Kod 14. Kod za gumb Register

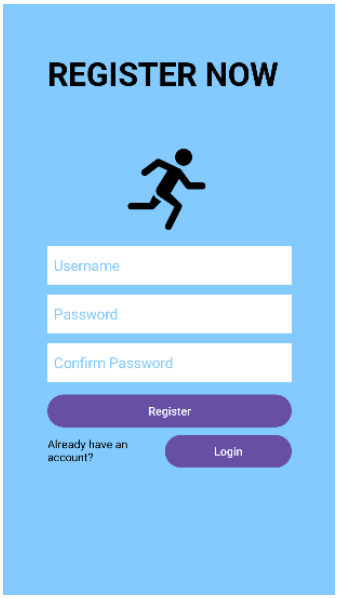
Kod listenera za klik na gumb (*Login*) u slučaju da korisnik već ima postojeći račun. Klik na gumb vodi na aktivnost (*Login*) za prijavu korisnika u aplikaciju.

```
binding.login.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(getApplicationContext(),
MainActivity4.class);
        startActivity(intent);
    }});}}
```

Kod 15. Kod za gumb Login

Frame Layout pokriva cijeli zaslon i postavljena je boja s kodom(*#82CAFF*). Komponenta grafičkog sučelja *TextView* (*REGISTER NOW*) ima veličinu teksta 40dp, podebljan je, centriran i crne boje. Komponenta grafičkog sučelja *EditText* (*Username, Password i Confirm Password*)

polja za unos podataka. Polja imaju bijelu pozadinu sa slovima boje pozadine. Ispod polja za unos podataka nalaze se 2 gumba (*Register i Login*). Gumb (*Register*) služi za registraciju korisnika, a gumb *(Login)* za prijelaz na novu aktivnost (*Login*).



Slika 8. Izgled Register Now aktivnosti

4.5.3. Prijava korisnika

Kod počinje s postavljenim listenerom za klik na gumb (*Login*). Pritiskom pokreće se metoda *onClick().* Dohvaćaju se uneseni podaci za korisničko ime i lozinku iz odgovarajućih polja.
U slučaju da su polja ostala prazna dolazi poruka korisniku da popuni sva polja (*Fill all fields*), ako su polja popunjena, podaci se provjeravaju metodom *checkuserpass().* Ako su korisničko ime i lozinka točni, dolazi poruka korisniku da je prijava uspješna (*Login Successfull*) i korisnik se preusmjerava na aktivnost (*Loading screen*), potom na aktivnost (*Add Exersice*), ako su netočni podaci dolazi poruka Nevažeći podaci *(Invalid data).*

```
binding.login.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String username = binding.username.getText().toString();
        String password = binding.password.getText().toString();
        if (username.equals("") || password.equals(""))
            Toast.makeText(MainActivity4.this, "Fill all fields",
Toast.LENGTH_SHORT).show();
            else{Boolean checkCredentials =
db.checkuserpass(username,password);
                if(checkCredentials == true)
                {
                    Toast.makeText(MainActivity4.this, "Login Successfull",
Toast.LENGTH_SHORT).show();
                    Intent intent = new Intent(getApplicationContext(),
MainActivityLS2.class);
                    startActivity(intent);
                }else {Toast.makeText(MainActivity4.this, "Invalid data",
Toast.LENGTH_SHORT).show();}}}});
```

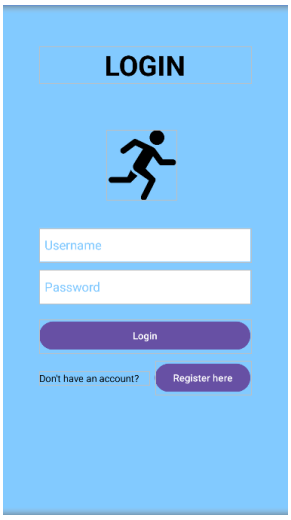
Kod 16. Kod za gumb Login

Metoda za pritisak na gumb (*Register here*), preusmjerava korisnika nazad na aktivnost (*Register*).

```
binding.regf.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(getApplicationContext(),
MainActivity3.class);
        startActivity(intent);}});};}
```

Kod 17. Kod za gumb Register Here

Frame Layout prekriva cijeli zaslon i postavljena mu je boja pozadine s kodom (*#82CAFF*). Komponenta grafičkog sučelja *TextView (LOGIN)* ima veličinu fonta 40dp, podebljan je, crne je boje i centriran. Komponenta grafičkog sučelja *EditText (Username, Password)* polja za unos podataka. Polja imaju bijelu pozadinu sa slovima boje pozadine. Ispod polja nalaze se 2 gumba (*Login i Register here*). Gumb (*Login*) služi za prijavu korisnika, a gumb (*Register here*) vodi korisnika na aktivnost za registraciju ukoliko nema račun za aplikaciju.



Slika 9. Izgled aktivnosti za login

4.5.4. Unos vježbi

Dio koda u kojem su deklarirani komponente grafičkog sučelja *EditText, Button*. Komponente grafičkog sučelja *EditText (name, muscle, sets, reps)* služe za unos imena vježbi, mišića koji se koriste pri toj vježbi, broj setova koje korisnik želi odraditi i broj ponavljanja koje će korisnik odraditi. Komponente grafičkog sučelja *Button (insert, update, delete, view)* izvršavaju unos, brisanje, uređivanje i pregled upisanih podataka u bazu podataka.

```
EditText name, muscle, sets, reps;
Button insert, update, delete, view;
DBHelper db;
```

Kod 18. Kod za deklaraciju u aktivnosti Add Workout

Kod za *Floating button* i za *Button* koji se koriste za prijelaz u druge aktivnosti. *Floating button* je za prelazak u aktivnost *(Settings/Help)*, a gumb *Weight* su za prelazak u aktivnost *(Add Weight)*.

```
Button btn=findViewById(R.id.button);
btn.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent=new Intent(MainActivity.this, MainActivity2.class);
        startActivity(intent);}});



FloatingActionButton fab = findViewById(R.id.fab);
fab.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent=new Intent(MainActivity.this, MainActivity3.class );
        startActivity(intent);}});
```

Kod 19. Kod za gumb Weight i leteći gumb

Metoda kojom se pronalaze komponente grafičkog sučelja *EditText* i *Button* po ID-u iz XML-a i kreira se nova instanca klase *DBHelper*.

```
name = findViewById(R.id.name);
muscle = findViewById(R.id.muscle);
sets = findViewById(R.id.sets);
reps = findViewById(R.id.reps);


insert = findViewById(R.id.button4);
update = findViewById(R.id.button5);
delete = findViewById(R.id.button6);
view = findViewById(R.id.button2);
db = new DBHelper(this);
```

Kod 20. Pronalazak po ID-u u aktivnosti Add Workout

Kod počinje s postavljenim listenerom za klik na gumb (*Insert*). Nakon pritiska na gumb pokreće se *onClick* metoda unutar koje se izvršava kod. Linije (*String*) uzimaju vrijednosti koje je korisnik upisao na zadana polja (*Name, Muscle, Sets i Reps*). Dohvaćaju tekst iz tih polja i pretvara ih u *String*. Linija koda poziva metodu *insertuserdata()* iz klase *DBHelper* koja upisuje podatke u bazu podataka. Nakon toga ide provjera unosa podataka, ako je upis uspješan na zaslonu se pojavljuje poruka (*Workout added*), u suprotnom dolazi poruka (*Try again*).

```
insert.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String nameTXT = name.getText().toString();
        String muscleTXT = muscle.getText().toString();
        String setsTXT = sets.getText().toString();
        String repsTXT = reps.getText().toString();
        Boolean checkinsertdata =
db.insertuserdata(nameTXT,muscleTXT,setsTXT,repsTXT);
        {
            if(checkinsertdata==true)
                Toast.makeText(MainActivity.this, "Workout
added",Toast.LENGTH_SHORT).show();
            else
                Toast.makeText(MainActivity.this, "Try
again",Toast.LENGTH_SHORT).show();
        }}});
```

Kod 21. Kod za gumb Add

Postavljen je listener za klik na gumb (*Update*). Pritiskom na gumb pokreće se metoda *onClick* te se izvršava kod. Linije (*String*) dohvaćaju tekst koji je korisnik upisao u polja (*Name, Muscle, Sets i Reps)* i pretvara ih u *String*. Linija poziva metodu *upadateuserdata()* iz klase *DBHelper* koja ažurira podatke (*Muscle, Sets i Reps*) one vježbe koja je već upisana s tim istim imenom u bazu podataka. Varijabla *checkupdatedata()* provjerava je li ažuriranje bilo uspješno. Ako je bilo uspješno korisniku dolazi poruka (*Workout updated*), a ako nije prikazuje se (*Workout not updated).*

```
update.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String nameTXT = name.getText().toString();
        String muscleTXT = muscle.getText().toString();
        String setsTXT = sets.getText().toString();
        String repsTXT = reps.getText().toString();

        Boolean checkupdatedata =
db.updateuserdata(nameTXT,muscleTXT,setsTXT,repsTXT);
        {
            if(checkupdatedata==true)
                Toast.makeText(MainActivity.this, "Workout
updated",Toast.LENGTH_SHORT).show();
            else
                Toast.makeText(MainActivity.this, "Workout not
updated",Toast.LENGTH_SHORT).show();
        }}});
```

Kod 22. Kod za gumb Change

Postavljen je listener za klik na gumb (*Delete*). Pritiskom na gumb pokreće se metoda i izvršava se kod. Dohvaća se tekst iz polja (*Name*) i pretvara u *String*. Metoda *deleteuser()* provjerava

postoji li upisana vježba s tim imenom u bazi podataka u slučaju da postoji svi podaci o toj vježbi se brišu uključujući ime vježbe i korisniku dolazi poruka (*Workout deleted*), a ako brisanje nije uspješno dolazi poruka (*Workout not deleted*).

```
delete.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String nameTXT = name.getText().toString();
        Boolean checkdeletedata = db.deletedata(nameTXT);
        {
            if(checkdeletedata==true)
                Toast.makeText(MainActivity.this, "Workout
deleted",Toast.LENGTH_SHORT).show();
            else
                Toast.makeText(MainActivity.this, "Workout not
deleted",Toast.LENGTH_SHORT).show();
        }}});
```

Kod 23. Kod za gumb Delete

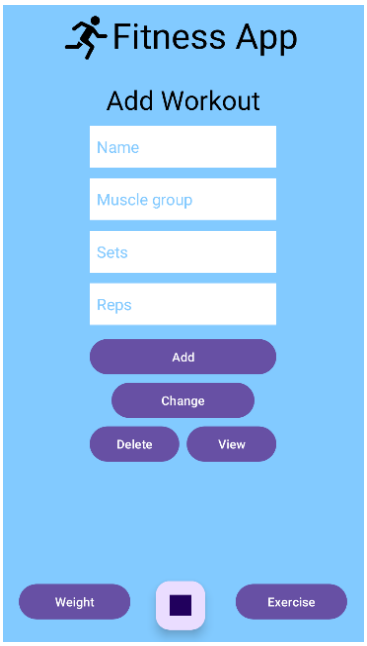
Klikom na gumb (*View*) pokreće se kod. Linija koda poziva metodu *getdata()* iz klase *DBHelper* i vraća rezultat u obliku *Cursor* objekta. *Cursor* je struktura koja sadrži rezultate pretrage iz baze podataka. Provjerava se ima li zapisa u bazi podataka. Ako nema, prikazuje se poruka (*No workouts added*) i vraća se iz funkcije. *StringBuffer* koristi se za dodavanje i spremanje podataka kako bi ih se prikazalo u dijalogu. Prolazi se kroz rezultate *Cursor*-a, pomiču se red po red i dodaju u *buffer*. Nakon toga kreira se novi *AlterDialog*, postavlja se naslov dijaloga i prikazuju se svi prikupljeni podaci iz baze podataka.

```
view.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View view) {
        Cursor res = db.getdata();
        if (res.getCount()==0)
        {
            Toast.makeText( MainActivity.this,"No workouts added",
Toast.LENGTH_SHORT).show();
            return;
        }
        StringBuffer buffer = new StringBuffer();
        while(res.moveToNext())
        {
            buffer.append("Name :"+res.getString(0)+"\n");
            buffer.append("Muscle :"+res.getString(1)+"\n");
            buffer.append("Sets :"+res.getString(2)+"\n");
            buffer.append("Reps :"+res.getString(3)+"\n");
        }
        AlertDialog.Builder builder = new AlertDialog.Builder(
MainActivity.this);
        builder.setCancelable(true);
        builder.setTitle("Exercise");
        builder.setMessage(buffer.toString());
```

```
        builder.show();
    }});}}
```

Kod 24. Kod za gumb View

Frame Layout prekriva cijeli zaslon i postavljena mu je boja pozadine s kodom (*#82CAFF*). Komponenta grafičkog sučelja *TextView (Add Workout)* ima veličinu fonta 30dp, podebljan je, crne je boje i centriran. Komponenta grafičkog sučelja *EditText (Name, Muscle group, Sets, Reps)* imaju bijelu pozadinu s slovima u boji pozadine. Ispod komponente grafičkog sučelja *EditText* nalaze se komponente *Button (Add, Change, Delete, View).* Komponenta grafičkog sučelja *Button(Add*) korisiti se za dodavanje podataka koji su upisani u gornja polja u bazu podataka, s komponentom *button(Change)* mijenjamo unesene podatke, s komponentom *button (Delete)* brišemo podatke, a s komponentom *button (View)* pregledavamo sve unesene podatke iz baze podataka. Komponente grafičkog sučelja *Buttoni (Weight , Exercise*) i *Floating button* koriste se kao meni za prijelaz između različitih aktivnosti.



Slika 10. Izgled aktivnosti Add Workout

4.5.5. Unos tjelesne težine

Dio koda u kojem su deklarirane komponente grafičkog sučelja *EditText (date, time, weight)* koji služe za unos datuma, vremena izračuna tjelesne težine i tjelesne težine. Komponente grafičkog sučelja *Button (insert, change, view)* koji izvršavaju unos, ažuriranje i pregled upisanih podataka u bazu podataka.

```
EditText date, time, weight;
Button insert, change, view;
DBHelper2 db;
```

Kod 25. Kod za deklaracija u aktivnosti Add Weight

Metoda kojom se pronalaze komponente grafičkog sučelja *EditText* i *Button* po *ID-u* iz XML-a i kreira nova instanca klase *DBHelper2.*

```
date = findViewById(R.id.date1);
time = findViewById(R.id.time);
weight = findViewById(R.id.weight);
insert = findViewById(R.id.insert);
change = findViewById(R.id.change);
view = findViewById(R.id.view);
db = new DBHelper2(this);
```

Kod 26. Pronalazak po ID-u u aktivnosti Add Workout

Kod počinje s postavljenim listenerom za klik na gumb (*Insert*). Nakon pritiska na gumb pokreće se *onClick* metoda unutar koje se izvršava kod. Linije (*String*) uzimaju vrijednosti koje je korisnik upisao na zadana polja (*Date, Time i Weight*). Dohvaćaju tekst iz tih polja i pretvara ih u *String*. Linija koda poziva metodu *insertuserdata()* iz klase *DBHelper2* koja upisuje podatke u bazu podataka. Nakon toga ide provjera unosa podataka, ako je upis uspješan na zaslonu se pojavljuje poruka (*Weight added*), u suprotnom (*Try again*).

```
insert.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String dateTXT = date.getText().toString();
        String timeTXT = time.getText().toString();
        String weightTXT = weight.getText().toString();
        Boolean checkinsertdata = db.insertdata(dateTXT,timeTXT,
weightTXT);
        {
            if(checkinsertdata==true)
                Toast.makeText(MainActivity2.this, "Weight
added",Toast.LENGTH_SHORT).show();
            else
                Toast.makeText(MainActivity2.this, "Try
again",Toast.LENGTH_SHORT).show();}}});
```

Kod 27. Kod za gumb Insert weight

Postavljen je listener za klik na gumb (*change*). Pritiskom na gumb pokreće se metoda *onClick* te se izvršava kod. Linije (*String*) dohvaćaju tekst koji je korisnik upisao u polja (*Date, Time i Weight*) i pretvara ih u *String*. Linija koda poziva metodu *upadatedata()* iz klase *DBHelper2* koja ažurira podatke (*Time i Weight*) one težine koja je upisana na taj datum. Varijabla *checkupdatedata()* provjerava je li ažuriranje bilo uspješno. Ako je bilo uspješno korisniku dolazi poruka (*Weight updated*), a ako nije prikazuje se (*Weight not updated*).

```
change.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String dateTXT = date.getText().toString();
        String timeTXT = time.getText().toString();
        String weightTXT = weight.getText().toString();
        Boolean checkupdatedata = db.updatedata(dateTXT,timeTXT,weightTXT);
        {
            if(checkupdatedata==true)
                Toast.makeText(MainActivity2.this, "Weight
updated",Toast.LENGTH_SHORT).show();
            else
                Toast.makeText(MainActivity2.this, "Weight not
updated",Toast.LENGTH_SHORT).show();}}});
```

Kod 28. Kod za gumb Edit weight

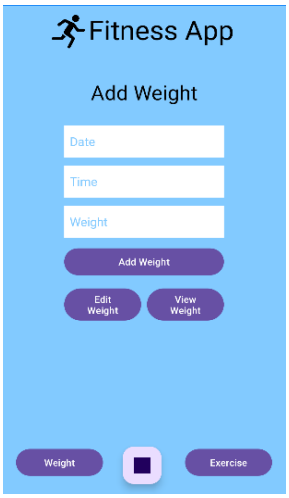
Klikom na gumb (*View*) pokreće se kod. Linija poziva metodu *getdata()* iz klase *DBHelper2* i vraća rezultat u obliku *Cursor* objekta. Provjerava se ima li zapisa u bazi podataka. Ako nema, prikazuje se poruka (*Please add weight*) i vraća se iz funkcije. *StringBuffer* koristi se za

dodavanje i spremanje podataka kako bi ih se prikazalo u dijalogu. Prolazi se kroz rezultate *Cursor-a*, pomiču se red po red i dodaju u *buffer*. Nakon toga kreira se novi *AlterDialog*, postavlja se naslov dijaloga i prikazuju se svi prikupljeni podaci iz baze podataka.

```
view.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Cursor res = db.getdataa();
        if (res.getCount()==0)
        {
            Toast.makeText( MainActivity2.this,"Please add weight",
Toast.LENGTH_SHORT).show();
            return;
        }
        StringBuffer buffer = new StringBuffer();
        while(res.moveToNext())
        {
            buffer.append("Date : "+res.getString(0)+"\n");
            buffer.append("Time : "+res.getString(1)+"\n");
            buffer.append("Weight : "+res.getString(2)+"\n");
        }
        AlertDialog.Builder builder = new AlertDialog.Builder(
MainActivity2.this);
        builder.setCancelable(true);
        builder.setTitle("Weight");
        builder.setMessage(buffer.toString());
        builder.show();
    }});}}
```

Kod 29. Kod za gumb View weight

.Frame Layout prekriva cijeli zaslon i postavljena mu je boja pozadine s kodom (*#82CAFF*). Komponenta grafičkog sučelja *TextView* (*Add Weight*) ima veličinu fonta 30dp, podebljan je, crne je boje i centriran. Komponenta grafičkog sučelja *EditText* (*Date, Time, Weight*) imaju bijelu pozadinu s slovima u boji pozadine. Ispod komponente *EditText* nalaze se komponente



Slika 11. Izgled aktivnosti Add Weight

Button (*Add Weight, Edit Weight, View Weight*). Komponente grafičkog sučelja Button (*Add Weight)* korisiti se za dodavanje podataka koji su upisani u gornja polja u bazu podataka, s komponentom *Button* (*Edit Weight*) mijenjamo unesene podatke, s komponentom *Button* (*View Weight)* pregledavamo sve unesene podatke iz baze podataka. Komponente grafičkog sučelja *Button (Weight , Exercise)* i *Floating button* koriste se kao meni za prijelaz između različitih aktivnosti.

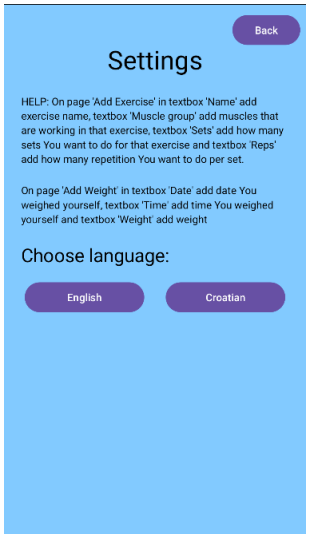
4.5.6. Postavke i pomoć

Kod za komponentu grafičkog sučelja *Button* na aktivnosti za postavke i pomoć. Komponenta *Button* s ID (*button3*) koristi se za povratak na početnu aktivnost (*Add Workout*) nakon ulaska u aktivnost (*Help i Settings*), a komponenta *Button* s ID (*cro*) služi za prijevod aplikacije na hrvatski jezik.

```
public class MainActivity3 extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_main3);
        Button btn = findViewById(R.id.button3);
        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(MainActivity3.this,
MainActivity.class);
                startActivity(intent);}});

        Button btncro = findViewById(R.id.cro);
        btncro.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intentcro = new Intent(MainActivity3.this,
MainActivitycro.class);
                startActivity(intentcro);}});};}
```

Kod 30. Kod za gumbove Back, English i Croatian

U aktivnosti *(Settings i Help)* opisano je što treba upisati u koje polje (*EditText*) na kojoj aktivnosti. Gumbovi (*English i Croatian*) služe za odabir jezika aplikacije, gumb (*Back*) služi za povratak na aktivnost (*Add Workout*).



Slika 12. Izgled aktivnosti Settings i Help

5. ZAKLJUČAK

Razvoj fitnes aplikacije za operacijski sustav Android u razvojnom okruženju Android Studio pokazao je zažnost planiranja i poznavanja programskog jezika Java. Kroz proces izrade mobilne aplikacije za operacijski sustav Android, usvojene su vještine, kao što su rad sa korisničkim sučeljem, rješavanje problema koji nailaze u procesu izrade mobilne aplikacije, upravljanje bazama podataka i tablicama u njima. Rezultat toga je funkcionalna mobilna aplikacija za operacijski sustav Android koja korisnicima omogućava upisivanje vježbi i praćenje napretka. Iako aplikacija zadovoljava osnovne zahtjeve, postoji veliki potencijal za dodatna poboljšanja. Neki od njih su integracija s pametnim uređajima, izrada personaliziranog plana treninga, funkcije za praćenje prehrane. Ovaj projekt postavio je čvrste temelje za budući razvoj mobilnih aplikacija za operacijski sustav Android.

LITERATURA

[1] Finoit. *What is Java Used for: Top 6 Uses of Java Programing*. Dohvaćeno iz https://www.finoit.com/articles/what-is-java-used-for/. (Pristupljeno dana 21.09.2024.)

[2] AWS. Amazon. *What is Java?*. Dohvaćeno iz https://aws.amazon.com/what-is/java/. (21.09.2024.)

[3] IBM. *What is Java?*. Dohvaćeno iz https://www.ibm.com/topics/java. (Pristupljeno dana 21.09.2024.)

[4] Wikipedia. *Android Studio.* Dohvaćeno iz https://en.wikipedia.org/wiki/Android_Studio. (Pristupljeno dana 20.09.2024.)

[5] Wikipedia. *Java (programming language).* Dohvaćeno iz https://en.wikipedia.org/wiki/Java_(programming_language). (Pristupljeno dana 21.09.2024.)

[6] Wikiwand. Android (operating system). Dohvaćeno iz https://www.wikiwand.com/en/articles/Android_(operating_system)#. (Pristupljeno dana 24.09.2024.)

[7] Developers Android. *Meet Android Studio.* Dohvaćeno iz https://developer.android.com/studio/intro. (Pristupljeno dana 25.09.2024.)

[8] Frid N., Jović A. *Modeliranje programske potpore UML – dijagramima, (2023.)*

Popis slika

Popis kodova