Android aplikacija za praćenje dnevnog unosa kalorija

Mahmutović, Mateo

Supplement / Prilog

Publication year / Godina izdavanja: **2019**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:195:403850

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-29**

Sveučilište u Rijeci
Fakultet informatike
i digitalnih tehnologija

Repository / Repozitorij:

Repository of the University of Rijeka, Faculty of Informatics and Digital Technologies - INFORI Repository

DIGITALNA KNJIŽNICA

Image not found or type unknown

Sveučilište u Rijeci – Odjel za informatiku

Nastavnički smjer

Mateo Mahmutović

Android aplikacija za praćenje dnevnog unosa kalorija

Diplomski rad

Mentor: dr. sc. Božidar Kovačić doc.

Rijeka, 28.09.2019.

Sadržaj

Sažetak ... 1

Ključne riječi ... 1

1. Uvod ... 2

2. Android ... 3

2. 1 Arhitektura Androida ... 4

2.1.1. Aplikacijski sloj ... 6

2.1.2. Aplikacijski framework ... 6

2.1.3. Programske Biblioteke i Android Runtime ... 7

2.1.4. Linux Kernel ... 7

2.2. Verzije Android Operativnog sustava ... 8

2.3. Android Uređaji ... 10

3. Razvojno okruženje ... 11

3.1 Android Studio ... 13

3.1.1. Struktura projekta ... 13

3.1.2 Gradle Build sustav ... 17

3.2. SQLite ... 17

3.3. Stetho alat ... 19

4 Aplikacija ... 22

4.1. Sigurnost ... 23

4.2.Fragmenti ... 24

5. Razvoj aplikacije „MojPlan“ ... 26

5.1. Aktivnost unosa podataka i početan zaslon ... 26

5.3 Izmjena osobnih informacija ... 37

5.4 Prikaz prehrane ... 38

5.5 Dnevnik prehrane ... 42

5.6 Baza podataka ... 42

5.7 Budući razvoj aplikacije ... 47

6. Zaključak ... 48

Literatura 49
Tablice 51
Slike 52

Sažetak

Cilj ovog rada je istražiti i opisati tehnologije potrebne za razvoj fitness Android aplikacije koja koristi SQLite bazu podataka te sama analiza razvijene aplikacije. Ideja aplikacije je da se koristi kao personalizirani dnevnik prehrane s vizualnim elementima koji pomažu korisniku u praćenju svog energetskog unosa te ostvarivanju određenog cilja. U teoretskom dijelu rada opisana je arhitektura, povijest i sigurnost Android-a. Analizirano je razvojno okruženje Android Studio, korištene tehnologije te je definiran pojam aplikacije, aktivnosti i fragmenata. U praktičnom dijelu detaljno su demonstrirane funkcionalnosti i dizajn aplikacije.

Ključne riječi

Android, Android Studio, aplikacija

1. Uvod

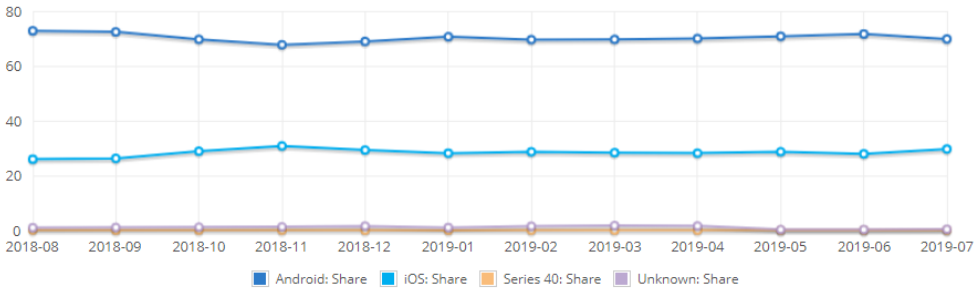
Nakon predstavljanja prvog modela iPhone pametnog telefona, te kasnije i Android operativnog sistema mobilni su uređaji postali vrlo popularni, te su u određenoj mjeri zauzeli ključan aspekt našeg poslovnog i privatnog života. Mobitel u današnjem društvu nije alat isključivo za upućivanje poziva i slanje SMS-ova, već je i uređaj koji pruža različite informacije i zabavu. Gotovo da i nema stvari kojoj korisnici više ne mogu pristupiti preko svojih mobitela. Važnost mobilnih uređaja pokrenula je snažnu konkurenciju među tehnološkim divovima kao što su Apple, Google i Microsoft. Tako je Google pokrenuo Android platformu za mobilne uređaje. Od svog predstavljanja, Android je privukao interes korisnika, tvrtki i programera te postao najrasprostranjeniji mobilni operativni sustav trenutno na tržištu. Sve je veća potreba za različitim aplikacijama koje mijenjaju način na koji živimo. Aplikacije za bankarstvo, dostavu, prijevoz, trgovinu, fitness, učenje i zabavu korisnici sve više prepoznaju i koriste kao alate koji nam pomažu u svakodnevnom životu i štede vrijeme.

Praktični zadatak ovog diplomskog rada je izrada Android aplikacije koja će služiti korisnicima za praćenje svoje prehrane kako bi lakše mogli regulirati vlastitu tjelesnu težinu. Aplikacija će na temelju različitih informacija koje korisnik unosi kao što su dob, spol, razina tjelesne aktivnosti i težina odrediti koliko bi kalorija korisnik trebao unijeti kako bi postigao svoj određeni cilj. U radu ćemo prije svega definirati što je Android, opisati arhitekturu OS-a, ključne komponente i prikazati povijest razvoja. U drugom dijelu rada analizirano je Android Studio razvojno okruženje te najvažnije programske knjižnice koje su korištene prilikom izrade aplikacije, kao što su SQLite i Stetho. Potrebno je definirati što je aplikacija, od kojih se dijelova svaka aplikacija sastoji te navesti neke mjere sigurnosti koje su ugrađene u svaku aplikaciju. U posljednjem dijelu rada opisan je razvoj te je prezentiran detaljan opis aplikacije pod nazivom „MojPlan“ koju sam izradio u sklopu ovog rada.

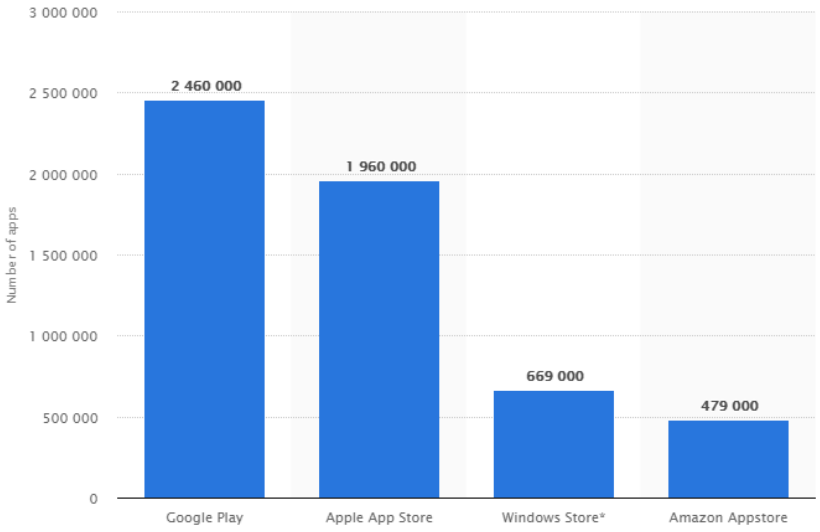
2. Android

Operativni sustavi za mobilne uređaje doživjeli su velike promjene u zadnjih 25 godina, od Palm OS-a, preko Blackberry OS-a do danas najpopularnijeg Androida. Android nije samo OS, već „stack“. To znači da se na njega ne odnosi samo operacijski sustav, već i skup programa, podsistema i samostalnih komponenti namijenjen za mobilne i ostale uređaje. Platforma je kreirana od strane Android Inc. koju je otkupio Google 2005. godine i izdana je kao Android Open Source Project (AOSP) 2007. godine. Grupa od 78 različitih kompanija tvori „Open Handset Alliance“ grupaciju koju čine tvrtke kao što su Dell, Samsung, Sony, Motorola i Nvidia. Cilj im je razvijati i distribuirati Android. Softver se može besplatno preuzeti sa službenih stranica i modificirati u skladu s Apache i BSD licencom. Razvoj Androida je relativno brz, veće zakrpe se isporučuju svakih nekoliko mjeseci. To stvara situaciju gdje informacije o platformi brzo postaju zastarjele i razni izvori poput knjiga i članaka teško prate razvoj. Izvori koji najbolje prate takav ritam su SDK dokumentacija te razni blogovi.

Sastoji se od Operativnog sustava, posredničkog softvera te ključnih aplikacija baziranih na Linux kernelu. Uključuje Java virtualnu mašinu, te Javu kao preporučeni programski jezik za većinu Android aplikacija**.** Omogućuje programerima da izrađuju aplikacije u Javi, te kasnije i upravljaju proizvodom preko Google-ove Java „depository“ programske biblioteke. Prema podacima sa slike 1 i 2 Android OS je najrasprostranjeniji mobilni OS s udjelom na tržištu od 70.24% prema podacima u razdoblju od 2018 – 08 do 2019 -07 (preuzeto sa stranice netmarketshare.com) te oko 2 460 000 aplikacija na Google Store-u (preuzeto sa statista.com).



SLIKA 1 - TRŽIŠNI UDIO MOBILNIH OPERATIVNIH SISTEMA



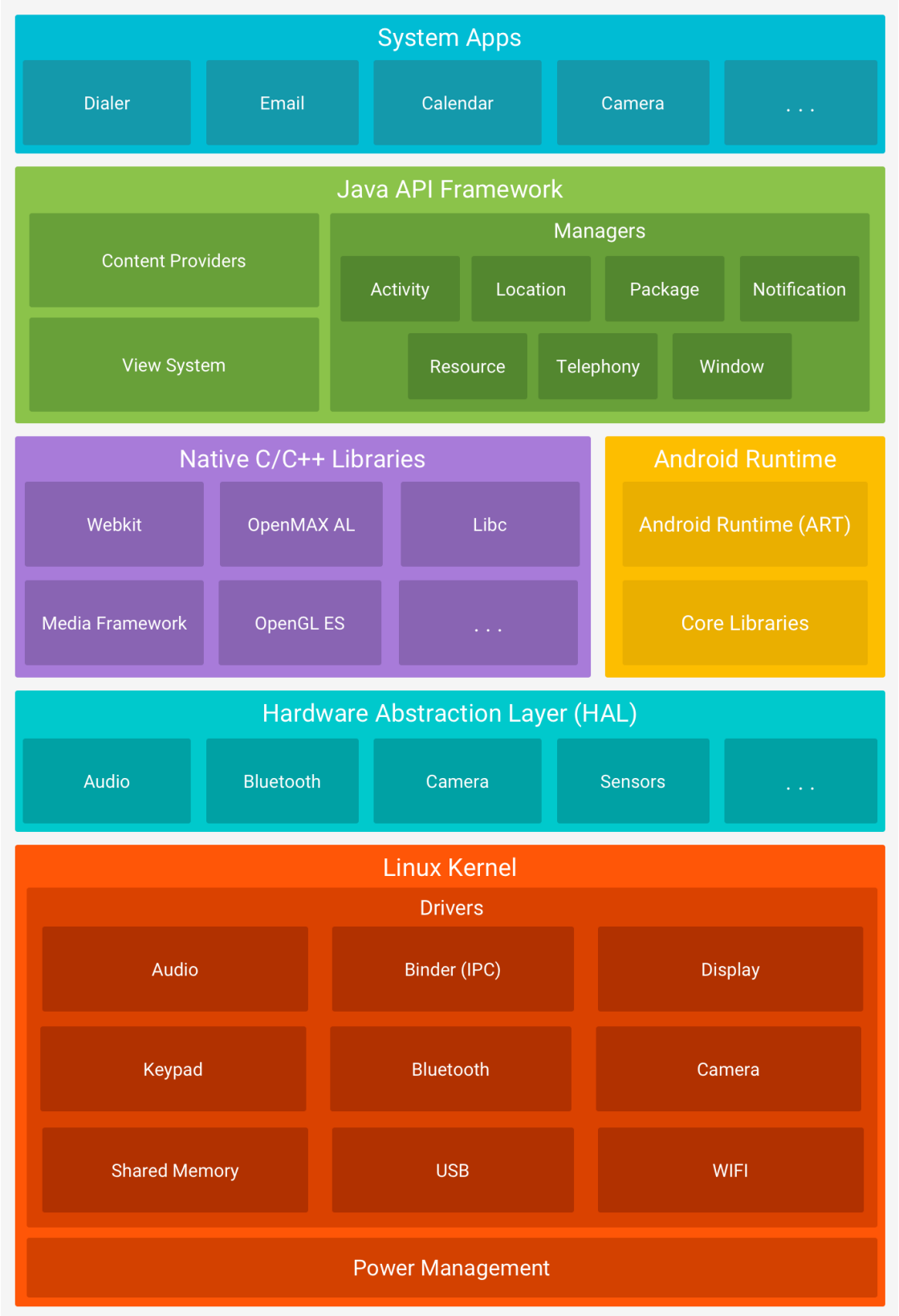
SLIKA 2 - BROJ APLIKACIJA NA NAJPOZNATIJIM MREŽNIM TRGOVINAMA

2. 1 Arhitektura Androida

Google Android OS je baziran na Linux kernel-u te koristi softverski dizajn u slojevima. Na vrhu „stack-a“ od 5 slojeva nalaze se aplikacije koje se pokreću preko virtualne mašine. Sistemske komponente napisane su u Java, C, C++ te XML programskim jezicima. Android arhitektura je prikazana dijagramom na slici 3**.**

Arhitektura Androida sastoji se od slojeva:

- Aplikacijski sloj
- Aplikacijski framework
- Programske knjižnice s Android Runtime među-slojem
- Apstrakcijski sloj fizičkih komponenti (HAL)
- Linux kernel



SLIKA 3 - ARHITEKTURA ANDROIDA

2.1.1. Aplikacijski sloj

Aplikacijski sloj je najviši sloj koji sadrži neke od osnovnih aplikacija koje uključuju elektroničku poštu, SMS program, kalendar, mape, preglednik, kontakte i slične aplikacije. Sve aplikacije u tom sloju napisane su koristeći Java programski jezik. Te aplikacije nemaju poseban status u usporedbi s aplikacijama koje korisnik odluči instalirati na uređaj. Tako korisnik može promijeniti svoj zadani preglednik, tipkovnicu ili SMS program. Jedan od izuzetaka je aplikacija Postavke.

2.1.2. Aplikacijski framework

Aplikacijski framework je softverski framework koji se koristi za implementaciju standardne strukture aplikacije za određeni operativni sustav. Uz pomoć različitih upravitelja, pružatelja sadržaja i ostalih usluga programeri mogu iskoristiti funkcije drugih aplikacija za svoje potrebe. Sve funkcionalnosti Android operativnog sistema dostupne su preko programskih sučelja napisanih u Java programskom jeziku. Dostupni su razvojnim programerima te tvore blokove od kojih je Android aplikacija izgrađena tako da pojednostavljuju kod, te dijele sistemske komponente i usluge u zasebne module, koji uključuju:

- „View System“ koji se može koristiti za izgradnju korisničkog sučelja
- Upravitelj Resursa
- Upravitelj Obavijesti
- Upravitelj Aktivnosti
- Pružatelji Sadržaja

2.1.3. Programske Biblioteke i Android Runtime

Sloj ispod se sastoji od dva dijela. Prvi su programske biblioteke koje su napisane ili-ili C++ programskom jeziku. One se pozivaju preko Java sučelja te omogućuju većinu funkcionalnosti koje su dostupne u osnovnim bibliotekama Java programskom jeziku. Android Runtime (ART) je drugi dio koji je od verzije Androida 4.4 KitKat zamijenio Dalvik virtualnu mašinu. Najveća razlika Dalvika i ART-a je ta što je Dalvik baziran na Just-in-Time (JIT) kompilaciji, dok je ART baziran na Ahead-of-Time (AOT) kompilaciji. S Dalvik JIT kompajlerom, svaki put kad se aplikacija pokreće, dio bytecodea se prevodi u strojni jezik. Kako se izvršenje obavlja, sve se više bytecodea kompajlira i sprema u cache memoriju. S druge strane, ART AOT kompajler statistički prevodi bytecode u strojni jezik i sprema ga u memoriju uređaja. To se dešava samo jednom, što utječe na poboljšanje performansi, štednju baterije, no duže vrijeme instalacije aplikacije u odnosu na Dalvik virtualnu mašinu

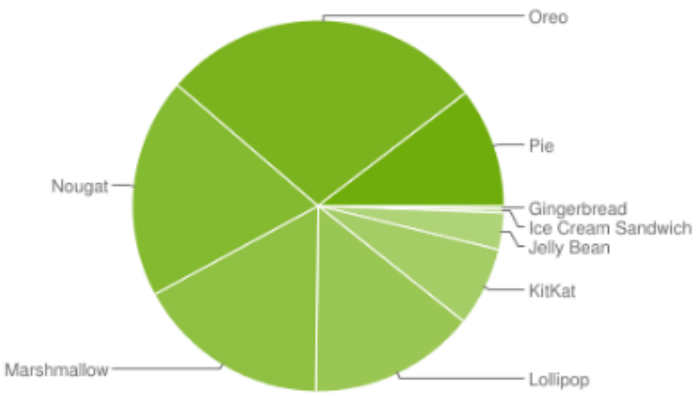
2.1.4. Linux Kernel

Android se oslanja na Linux 2.6 verziju kernel-a za osnovne sistemske servise kao što su sigurnost, upravljanje memorijom na nižim razinama, procesima, mrežnim zahtjevima i verzijama drivera. Kernel služi i kao apstraktni sloj između hardvera i ostatka softverskih slojeva te omogućava proizvođačima da razvijaju driver-e za svoje uređaje.

2.2. Verzije Android Operativnog sustava

Povijest Android Operativnog sustava započela je s Android beta verzijom petog studenog 2007. godine. Distribucija prve komercijalne verzije, Android 1.0 izdana je 23. Rujna 2008. godine. Android 1.5 Cupcake, izdan 27. travnja 2009. je prva verzija Androida koja je imala naziv po desertu, što će postati standard za sve ostale zakrpe i verzije Androida. Honeycomb je prva zakrpa Androida namijenjena isključivo za tablete. Izdana je 22. veljače 2011. godine. Android 4.0 Ice Cream Sandwich zakrpa, bazirana na Linux kernel verziji 3.0.1. izdana je 19. listopada 2011. To je posljednja verzija koja je službeno podržavala Flash player Adobe Systems-a. Jelly Bean verzija izdana u srpnju 2012. godine imala je zadaću poboljšati funkcionalnost i performanse korisničkog sučelja. Poboljšanje performansi uključivalo je „Projekt Butter“ koji koristi predviđanje dodira, trostruki buffering, vsync te fiksiranih 60 sličica u sekundi kako bi se stvorilo fluidno korisničko sučelje. KitKat verzija izdana u rujnu 2013. optimizirana je za pokretanje na većem broju uređaja nego prijašnje verzije. Lollipop je sljedeća verzija Androida. Neke od najbitnijih promjena su redizajniranog korisničkog sučelja, poboljšanje notifikacija, te je Android Runtime službeno zamijenio Dalvik kako bi se dodatno poboljšale performanse. Android 6.0 Marshmallow verzija, izdana u kolovozu 2015. je donijela dodatne funkcionalnosti te ponovni redizajn sučelja. Nougat izdana je u kolovozu 2016 dok je Oreo izdan godinu kasnije te je osmo veliko izdanje Android OS-a. Pie je zadnja izdana verzija, te deveto izdanje Android OS-a. U ožujku 2019. najavljena je najnovija verzija Androida 10.0 Q. Na slici 4 pruženi su detaljni podaci o relativnom broju uređaja koji pokreću datu verziju platforme Android.

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x	Jelly Bean	16	1.2%
4.2.x		17	1.5%
4.3		18	0.5%
4.4	KitKat	19	6.9%
5.0	Lollipop	21	3.0%
5.1		22	11.5%
6.0	Marshmallow	23	16.9%
7.0	Nougat	24	11.4%
7.1		25	7.8%
8.0	Oreo	26	12.9%
8.1		27	15.4%
9	Pie	28	10.4%



SLIKA 4 - RELATIVAN BROJ UREĐAJA KOJI POKREĆU ODREĐENU VERZIJU PLATFORME ANDROID

2.3. Android Uređaji

Uređaji namijenjeni za Android su većinom pametni telefoni i tableti, dok se uporaba Android tehnologije koristi i u raznim ostalim područjima, uglavnom za pametne satove, automobilske kontrolne ploče i slične uređaje. Takva namjena nosi sa sobom razne prednosti i nedostatke u razvoju mobilnih aplikacija. Iako su Internet usluge bile dostupne na mobilnim uređajima od sredine devedesetih godina s razvojem HDML jezika (Handheld device markup language), tek sredinom prošlog desetljeća su mobiteli s mogućnošću Internet pristupa postali masovno popularni. Zahvaljujući trendovima kao što su tekstualne poruke i proizvodima poput iPhone-a, telefoni koji mogu poslužiti kao uređaji za pristup internetu su naglo stekli popularnost te tako pridonesli razvoju sličnih uređaja. Dakle, kao neke od prednosti su mogućnost rada sa zanimljivim novim tehnologijama, te rad u brzo-rastućem tržišnom segmentu. Neki od najvećih od nedostataka su problemi s razvojem softvera namijenjen za male ekrane, te su korisnička sučelja i tipkovnice također mali što čini rad s takvim uređajima otežan. Procesorska brzina i memorija su još uvijek na većini uređaja slabiji uspoređujući ih sa stolnim računalima ili laptopima.

3. Razvojno okruženje

Integrirano razvojno okruženje ili IDE omogućava kodiranje na osobnom računalu te olakšava programeru proces razvoja nekog softvera. Neke od danas najraširenijih funkcionalnosti raznih razvojnih okruženja su automatsko predviđanje koda i označavanje sintakse programskog jezika različitim bojama. Riječi u kodu koje imaju neko posebno značenje kao što su imena klasa, metoda, varijabli i ostalih ključnih riječi biti će označeno različitim bojama kao što je prikazano na slici 5.

```
@Override
public void onAttach(Context context) {
    super.onAttach(context);
    if (context instanceof OnFragmentInteractionListener) {
        mListener = (OnFragmentInteractionListener) context;
    } else {
        throw new RuntimeException(context.toString()
                + " must implement OnFragmentInteractionListener");
    }
}
```

SLIKA 5 - PRIMJER OZNAČAVANJA SINTAKSE

U Java programskom jeziku, prije nego što se program pokrene**,** datoteka sa .java sufiksom mora biti transformirana u izvršnu datoteku sa .class sufiksom pomoću kompajlera. Nakon što je datoteka kompajlirana, program se pokreće u terminalu. IDE nudi i razne alate za otkrivanje i ispravljanje pogreški u kodu. Za razvoj android aplikacija danas se većinom koristi Android studio koji je zamijenio Eclipse kao službeno razvojno okruženje za Android. Neka od ostalih razvojnih okruženja koja se mogu koristiti su Visual Studio, IntelliJ IDEA, NetBeans, Komodo i ostali. Njihova usporedba prikazana je u tablici 1.

TABLICA 1 – USPOREDBA RAZVOJNIH OKRUŽENJA ZA RAZVOJ ANDROID APLIKACIJA

RAZVOJNA OKRUŽENJA ZA ANDROID	**PODRŽANI PROGRAMSKI JEZICI**	**PODRŽANI OPERATIVNI SISTEMI**	**LICENCA**	**CIJENA**
Android Studio	Java C C++ Kotlin	Windows MacOS Linux	Besplatan softver	Besplatan
Eclipse	Java C C++ C# JavaScript Python	Svi Operativni sistemi koji podržavaju Java programski jezik	Eclipse javna licenca	Besplatan
Visual Studio	C++ C C# Visual Basic PHP JavaScript	Windows MacOS Linux	Open Source MIT	Osnovna verzija besplatna, Jednokratno plaćanje do 2999$
IntelliJ IDEA	Java Scala Groovy Kotlin JavaScript TypeScript SQL	Svi Operativni sistemi koji podržavaju Java programski jezik	Apache 2.0 licenca	Osnovna verzija besplatna, Godišnja naknada do 499$
NetBeans	Java C C++ HTML PHP JavaScript	Wjndows MacOS Linux Solaris	CDDL 1.0 i GPL2	Besplatan
Komodo	Java JavaScript Python PHP HTML Ruby	Windows MacOS Linux	Mozilla javna licenca	Osnovna verzija besplatna, Jednokratno plaćanje do 394$

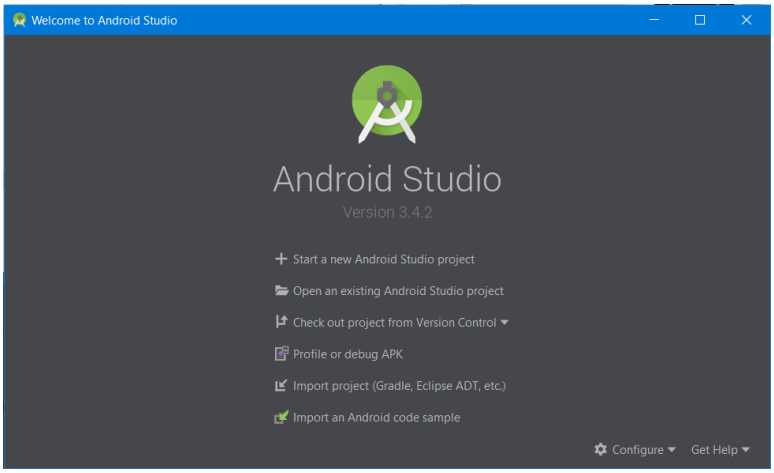
3.1 Android Studio

Android Studio je preporučeno razvojno okruženje za razvoj Android aplikacija te je besplatan za preuzimanje i korištenje s web sjedišta developer.android.com. Podržane su verzije za Windows, Mac OS te Linux Operativne sisteme. Android Studio je baziran na IntelliJ IDEA razvojnom okruženju što se može primijetiti po sličnosti sučelja. Android Studio podržava i iste programske jezike kao IntelliJ IDEA, odnosno Java i C++, te od Android Studio verzije 3.0, izdane u listopadu 2017. i programski jezik Kotlin. Razvijen je od strane Google-a. Prva stabilna verzija izdana je u prosincu 2014. godine te je od tada zamijenio Eclipse kao primarni IDE za razvoj Android aplikacija.

Sistemski zahtjevi za Android Studio 3.0 su novije verzije Windows, Mac OS, Linux te Chrome OS operacijskih sustava, instaliran JDK 8, rezolucija zaslona od barem 1280x800 piksela, minimalno 4GB RAM-a i 6GB slobodne memorije na tvrdom disku, dok je preporučeno 8GB RAM-a te 16GB memorije na tvrdom disku. Postavljanje Android razvojnog okruženja je prilično jednostavno. Koraci su slični za sve podržane operativne sustave (s manjim varijacijama), a to su instalacija JDK verzije za određeni OS i procesor, preuzimanje sa službene stranice te instalacija.

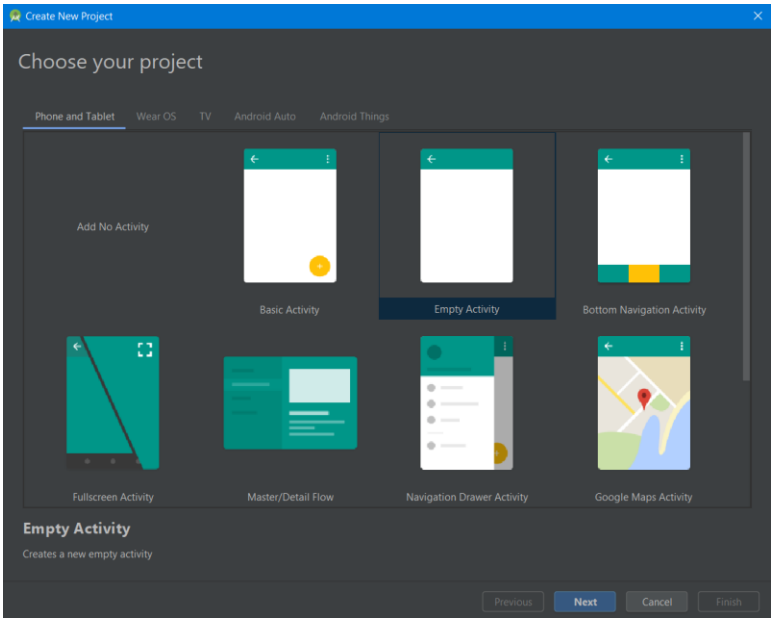
3.1.1. Struktura projekta

Nakon pokretanja Android Studio-a, između ostalih opcija imamo mogućnost kreiranja novog projekta. Prozor dobrodošlice je prikazan svaki put kada je Android Studio pokrenut bez otvorenih projekata te je prikazan na slici 6. Projekt se može zatvoriti odabirom Close project opcije. Ako je korisnik izašao iz Android Studio-a bez prethodnog zatvaranja projekta, alat će zaobići taj prozor te će automatski otvoriti prethodno aktivan projekt.



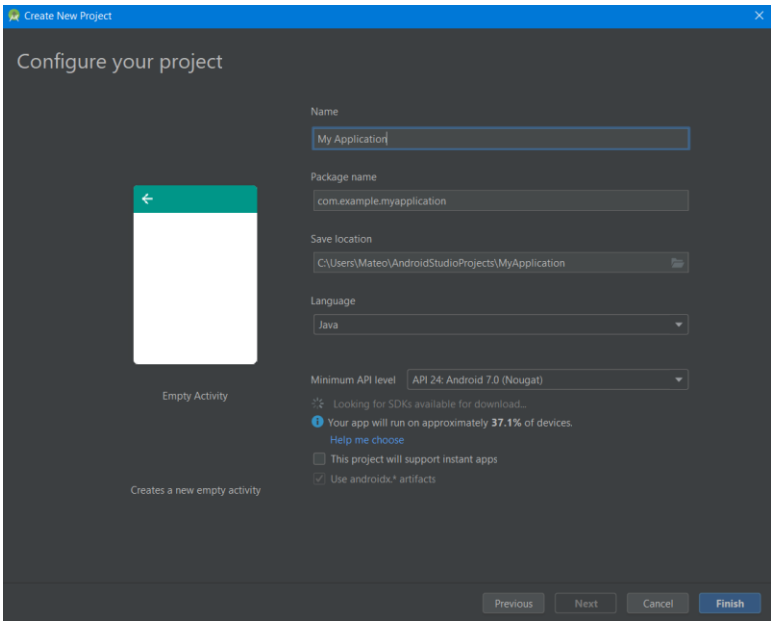
SLIKA 6 - PROZOR DOBRODOŠLICE

Prvi korak je prikazan na slici 7 gdje odabiremo uređaj za koji smo namijenili svoj projekt. Imamo mogućnost izrade aplikacije namijenjene za mobitele i tablete, pametne satove, TV, automobilska računala ili Android Things. Ispod odabiremo tip glavne aktivnosti za koju želimo izraditi određeni projekt.



SLIKA 7 - PRVI KORAK, KONFIGURACIJA PROJEKTA

Sljedeći korak, prikazan na slici 8, uključuje konfiguraciju projekta, odabir imena projekta, naziv paketa, lokacija, programski jezik se minimalni SDK, odnosno API razina na kojoj će se moći pokrenuti aplikacija.

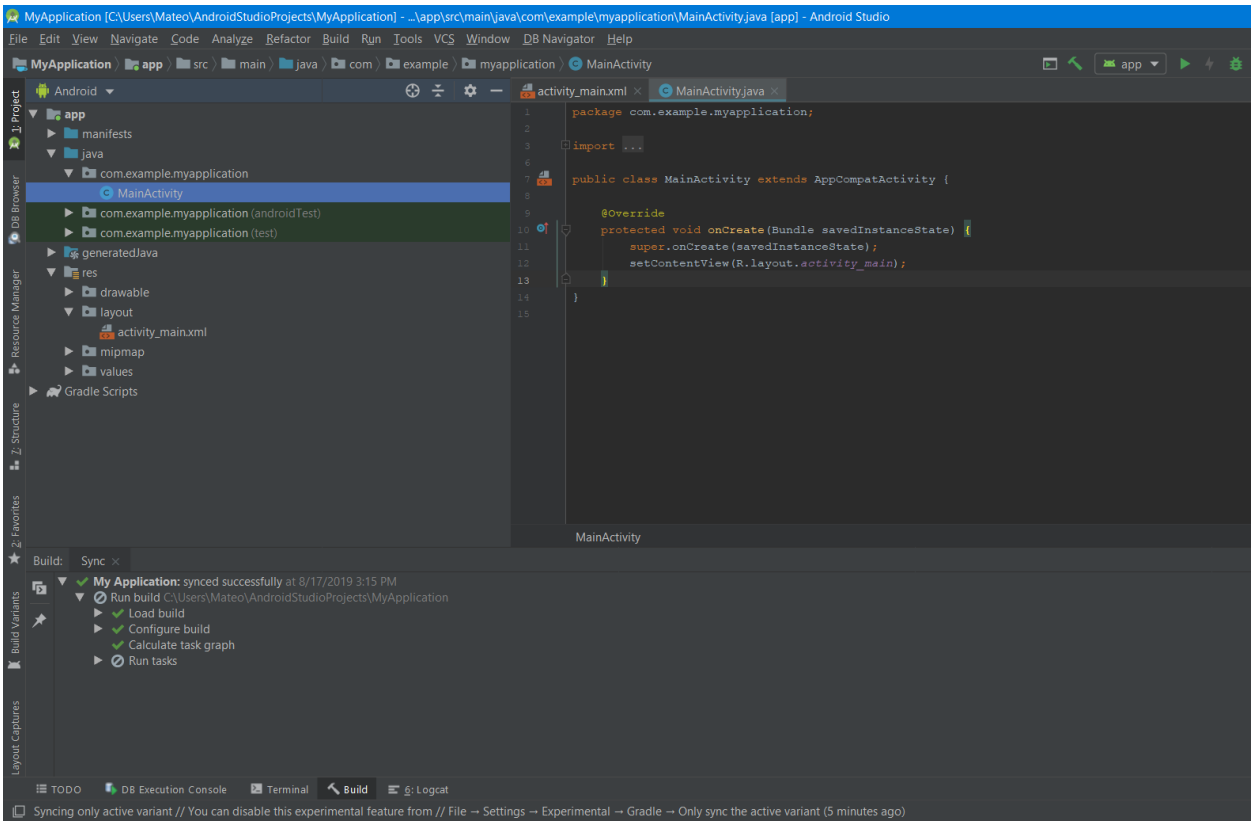


SLIKA 8 - DRUGI KORAK, KONFIGURACIJA PROJEKTA

Svaki projekt sadrži jedan ili više modula s izvornim kodom i resursnim datotekama. Vrste modula uključuju: Android app moduli, moduli biblioteka, Google App Engine moduli. Kao što se primijeti na slici 9 Android Studio prikazuje projektne datoteke po početnim postavkama s lijeve strane prozora. Taj pogled je organiziran tako da se omogućuje brzi pristup ključnim datotekama projekta:

- manifesti: sadrži AndroidManifest.xml datoteku koja služi za pokretanje i konfiguraciju projekta.
- java: Sadrži Java datoteke s izvornim kodom, uključujući Junit testni kod.
- res: Sadrži dizajn u XML-u, slike i sve ostale resurse koji nisu Java, C++ ili Kotlin kod unutar više podmapa. Drawable mapa sadrži slikovne datoteke, layout sadrži dizajn u XML jeziku za svaku aktivnost ili fragment. Menu sadrži informacije o izborniku dok values

mapa sadrži strings, styles i color datoteke. Strings mapa sadrži String vrijednosti koje se mogu pozivati u aplikacije, dok styles i colors utječu na daljnji dizajn aplikacije.



SLIKA 9 - PROZOR ANDROID PROJEKTA

Korisničko sučelje Android Studio-a se sastoji od:

1. Alatne trake koja omogućuje pokretanje aplikacije i ostalih alata.
2. Navigacijske trake koja omogućuje navigaciju kroz projektne datoteke, te kompaktniji pregled strukture projekta u odnosu na glavni prozor.
3. Prozora za uređivanje koda u kojem programer kreira i modificira programski kod ili dizajn, ovisno o vrsti datoteke.
4. Traka s alatima nalazi se s vanjske strane IDE prozora, te sadrži gumbe za proširenje ili sažimanje određenih prozora/pogleda.
5. Statusna traka prikazuje trenutno stanje projekta i razvojnog okruženja, te javlja korisniku različite obavijesti, poruke i upozorenja.

Korisničko sučelje se može prerasporediti po želji kako bi se dodatno olakšao rad programera. Za većinu funkcionalnosti postoje različite kratice na tipkovnici.

3.1.2 Gradle Build sustav

Kompajleri, Osim kompajliranja – prevođenja programskog koda u objektni kod**,** vrše i povezivanje, odnosno pretvaranje u datoteku koja se može pokrenuti na računalu. Build-anje projekta uključuje dodatne radnje koje omogućuju provedbu testova kompajliranog koda. Build sustav služi kako bi na izvornim programskim datotekama primijenio neke alate kompajliranja i povezivanja te grupirao sve u jedinstvenu datoteku s .apk sufiksom kojom Android OS zna upravljati.

Build sustav koji se koristi unutar Android Studija naziva se Gradle. On se pokreće kao integrirani alat unutar Android Studija, pokretan nezavisno od komandne linije. Funkcionalnosti Gradle Build sustava uključuju izmjenu i konfiguraciju build procesa, kreiranje više APK-ova za aplikaciju, te ponovno korištenje koda. Omogućuje spremanje aplikacije za daljnju distribuciju.

3.2. SQLite

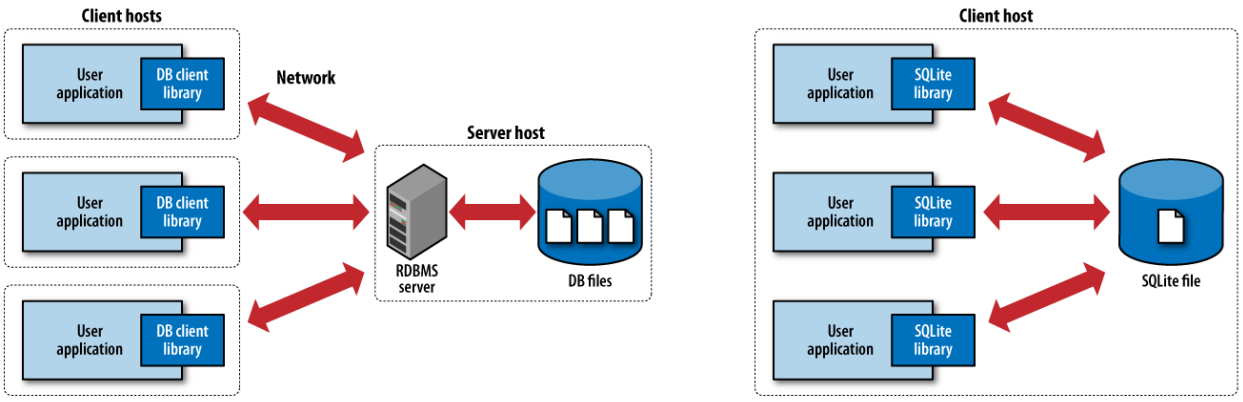
SQLITE je softverski paket koji pruža sustav za upravljanje relacijskim bazama podataka. Relacijske baze podataka se koriste za pohranu podataka korisnika u obliku tablica. Dodatno, paket pruža složene upite za kombiniranje podataka iz više tablica za generiranje izvješća i sažetaka o podacima. Ostali popularni komercijalni sustavi za upravljanje bazama podataka su Oracle Database, IBM-ov DB2 te Microsoftov SQL Server, dok su najpopularniji sustavi otvorenog koda MySQL i PostgreSQL. SQLite ima reputaciju kao prenosiv, lak za korištenje, kompaktan, učinkovit i pouzdan sustav za upravljanje relacijskim bazama podataka. SQLite je ugrađena baza

podataka. To znači da se pokreće zajedno, odnosno kod je isprepleten, ugrađen kao dio programa koji je sadrži.

Svojstva SQLITE softverskog paketa:

- Nije potreban odvojeni serverski proces ili sistem za upravljanje. SQLite knjižnica pristupa pohranjenim datotekama direktno.
- Bez konfiguracije - izostanak servera znači da nema podešavanja postavki.
- Nije potrebna administracija pošto se čitava baza podataka nalazi u jednoj datoteci.
- Jedna programska knjižnica sadrži cijeli sistem baze podataka, koji se integrira direktno u aplikaciju
- Paket je u osnovnom obliku manji od megabajta te zahtijeva tek nekoliko megabajta memorije.
- Transakcije dopuštaju pristup s više procesa ili procesnih niti.
- SQLite podržava većinu funkcionalnosti koje se nalaze unutar SQL92 standarda.

SQLite programska knjižnica je vrlo pouzdana zbog razvojnog tima koji konstantno nadograđuje i testira svoj kod. Na slici 10 prikazana je usporedba klijent – server arhitekture i SQLite arhitekture.



SLIKA 10 - KLIENT - SERVER ARHITEKTURA I SQLITE ARHITERKTURA

3.3. Stetho alat

Stetho je programska knjižnica otvorenog koda koja povezuje Android aplikaciju s Chrome razvojnim alatima, koje su dio Chrome preglednika. Kreirana od strane Facebook-a te služi za različite operacije za otklanjanje softverskih pogreški nad Android aplikacijom. Stetho se može koristiti za pregled baze podataka, datoteka, te mrežnih zahtjeva i provjeru statusnih kodova.

Jedan od razloga odabira Stetho alata je lakoća korištenja i integriranja u aplikaciju. Za integraciju jednostavno dodamo linije koda sukladno uputama na službenoj stranici, uvrstimo Stetho u „dependencies“ datoteku te omogućimo mrežnu provjeru baze podataka. Prikazane su linije koda za „dependencies“ datoteku te integraciju u aplikaciju na slici 11.

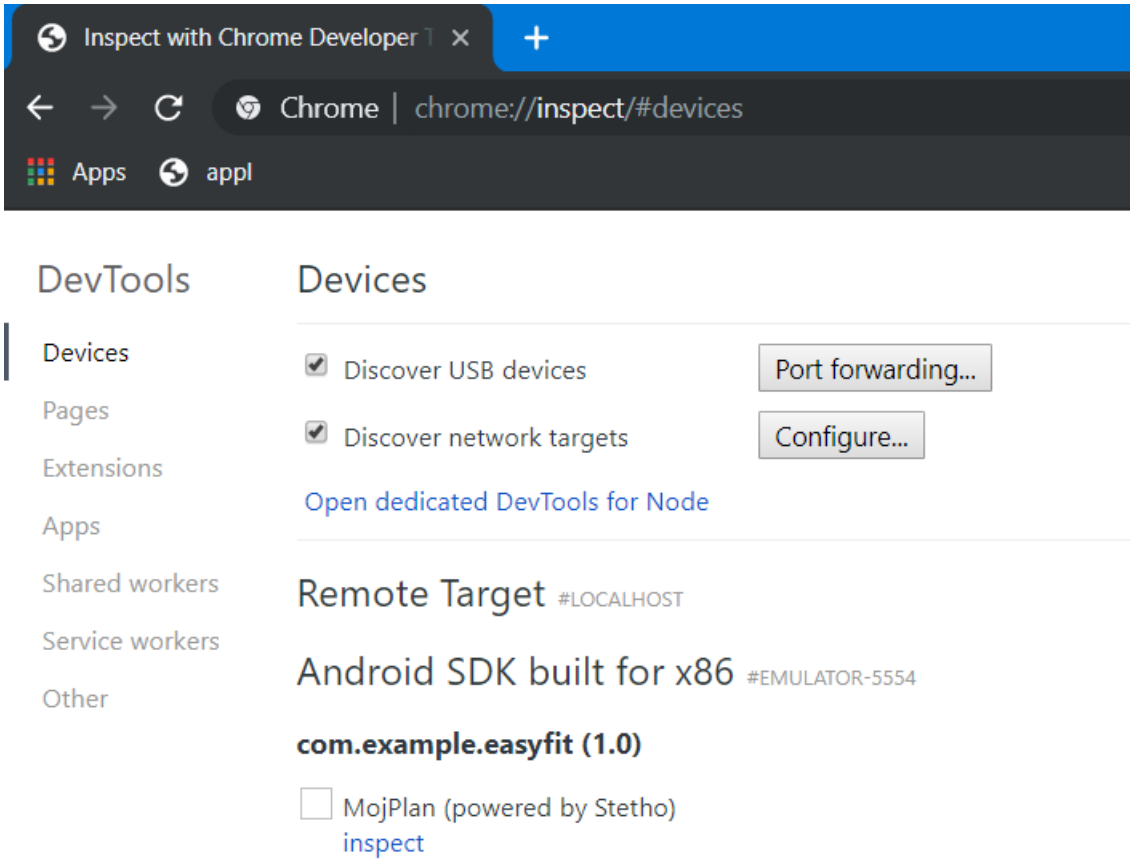
```
implementation 'com.facebook.stetho:stetho:1.5.1'
implementation 'com.facebook.stetho:stetho-okhttp3:1.5.1'
```

SLIKA 11 - GRADLE DEPENDENCIES

```
//STETHO
Stetho.initializeWithDefaults( context: this);
new OkHttpClient.Builder()
        .addNetworkInterceptor(new StethoInterceptor())
        .build();
```

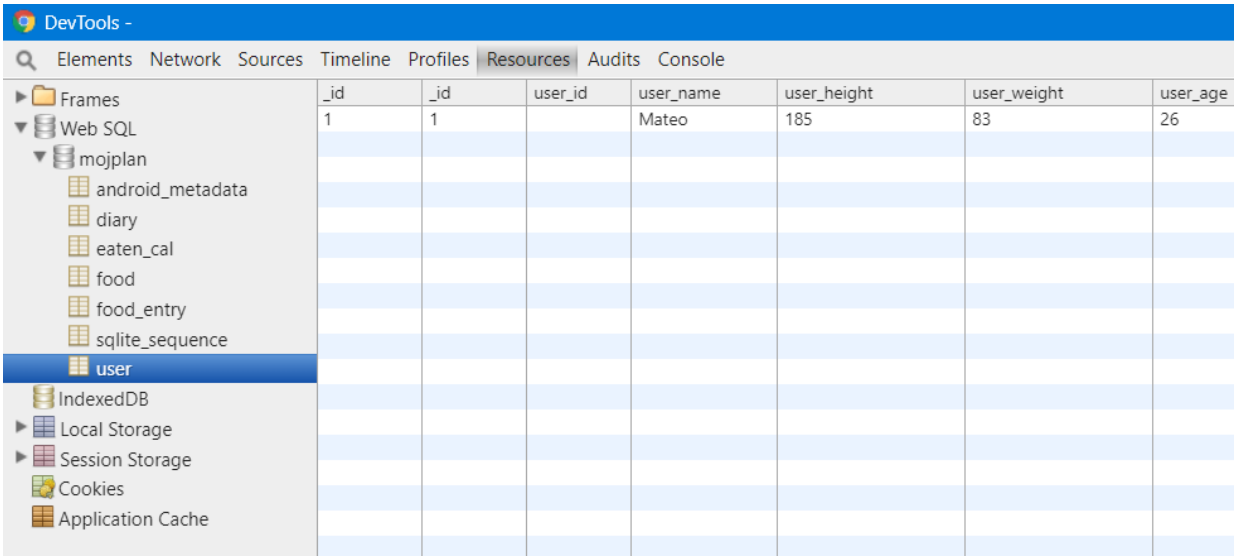
SLIKA 12 - INTEGRACIJA STETHO ALATA U APLIKACIJU

Nakon uspješne integracije programske knjižnice u projekt, pokrenemo emulator, te u Chrome preglednik upišemo Chrome://inspect što nas odvodi do Chrome razvojnih alata. Naš uređaj će postati vidljiv te ćemo ga moći pregledati klikom na poveznicu inspect koja se nalazi ispod imena našeg projekta kao što je i prikazano na slici 13.



SLIKA 13 - CHROME DEVTOOLS

Na slici 14 možemo primijetiti da Stetho pruža detaljno korisničko sučelje putem Chrome razvojnih alata kako bi razvoj aplikacija bio brži i efektivniji. U mojem slučaju koristio sam Stetho za provjeru podataka unutar tablica svoje baze podataka.



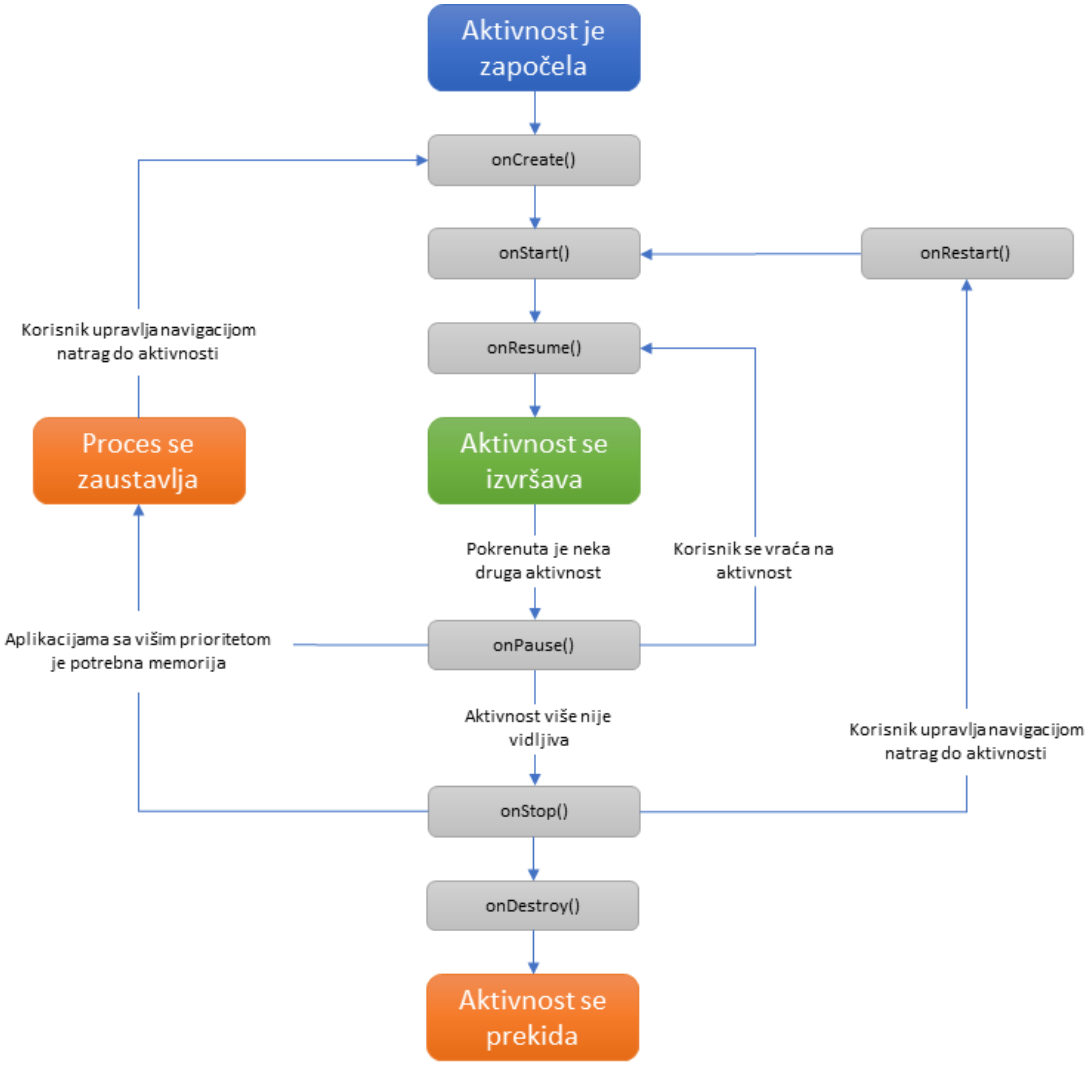
SLIKA 14 - WEB SQL

4 Aplikacija

Aplikacija unutar Android Studio-a može se napisati koristeći Java, Kotlin ili C++ programski jezik. Kod se kompajlira zajedno sa svim podacima i resursnim datotekama u Android paket, što je datoteka sa .apk sufiksom. Komponente aplikacije su blokovi od kojih je svaka aplikacija izgrađena. Postoje četiri vrste komponenti, koje moraju biti napisane u android manifestu:

1. Aktivnosti - predstavljaju ekran na kojem se vrši interakcija korisnika s aplikacijom.
2. Usluge - Nema korisničkog sučelja, omogućavaju rad aplikacije u pozadini radi izvršenja dugoročnih operacija, prikupljanja podataka ili ostalih usluga za koje je aplikacija namijenjena
3. Primatelji emitiranja - Vrše primanje vanjskih podataka. Omogućavaju dohvaćanje podataka aplikacijama koje nisu ni pokrenute, kako bi aplikacija notificirala korisnika o nekom događaju.
4. Pružatelji sadržaja - upravljaju zajedničkim aplikacijskim podacima koji se mogu spremati u SQLite bazu podataka, na web ili bilo koju lokaciju koju aplikacija može dohvatiti.

Na slici 15 prikazan je životni ciklus Android aktivnosti.

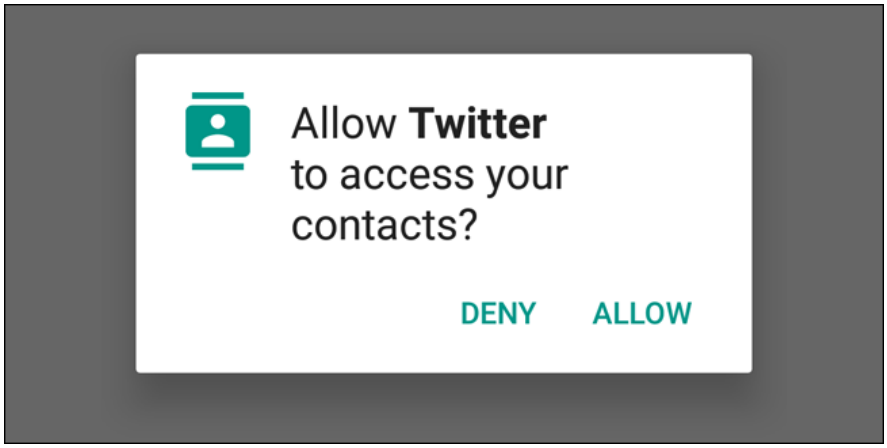


SLIKA 15 - ŽIVOTNI CIKLUS AKTIVNOSTI

4.1. Sigurnost

Svaka Android aplikacija se nalazi unutar svog sigurnosnog okvira, zaštićena pomoću različitih Android funkcionalnosti. Svaka se aplikacija unutar Android OS-a smatra različitim korisnikom. Sistem svakoj aplikaciji dodijeli jedinstveni korisnički ID. Sistem zadaje dozvole za rad s datotekama unutar aplikacije isključivo za zadani ID. Svaki proces se pokreće izoliran sa

vlastitom virtualnom mašinom, dok se svaka aplikacija pokreće unutar vlastitog Linux procesa. Android OS pokreće proces kad se određena komponenta treba izvršiti, te zaustavlja proces kad više nije potreban ili kad je sustavu potrebna memorija za druge aplikacije. Android OS radi po principu „minimum privilegija“. To znači da aplikacija ima pristup samo onim komponentama sustava koje su potrebne za rad te aplikacije. Postoje načini za pristup sistemskim komponentama, no za to korisnik treba dati eksplicitnu dozvolu, na primjer za pristup kameri, lokaciji, kalendaru, kontaktima itd. kao što je prikazano na slici 16.

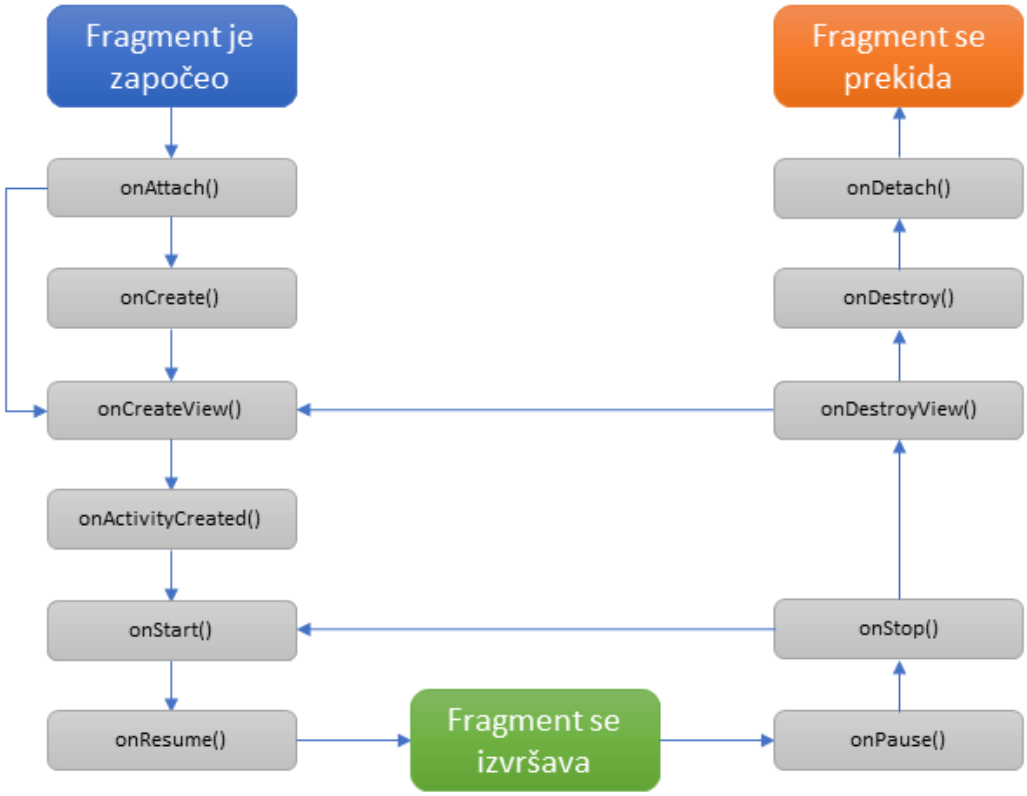


SLIKA 16 - NOTIFIKACIJA ZA DODJELU DOZVOLE APLIKACIJI

4.2.Fragmenti

Unutar aplikacije koju sam izradio u sklopu diplomskog rada korišteni su fragmenti. Fragmenti predstavljaju jedan dio korisničkog sučelja. Fragment se može smatrati dijelom aktivnosti, odnosno zasebnom pod-aktivnosti, sa vlastitim ciklusom, koji može započeti ili završiti tijekom aktivnosti unutar koje je pozvan. Prednost korištenja fragmenata je taj što se jedan fragment može iskoristiti unutar više različitih aktivnosti. Fragment se mora uvijek stvoriti unutar neke aktivnosti koja na njega direktno utječe.

Fragmenti su uvedeni u Android verziji 3.0 kao podrška fleksibilnijim i dinamičnijim sučeljima za veće ekrane, kao što su tableti. Na primjer fragmenti se mogu prikazati zajedno unutar jedne aktivnosti na većem ekranu, ili zasebno na manjem. Na slici 17 prikazan je životni ciklus fragmenta.



SLIKA 17 - ŽIVOTNI CIKLUS FRAGMENTA

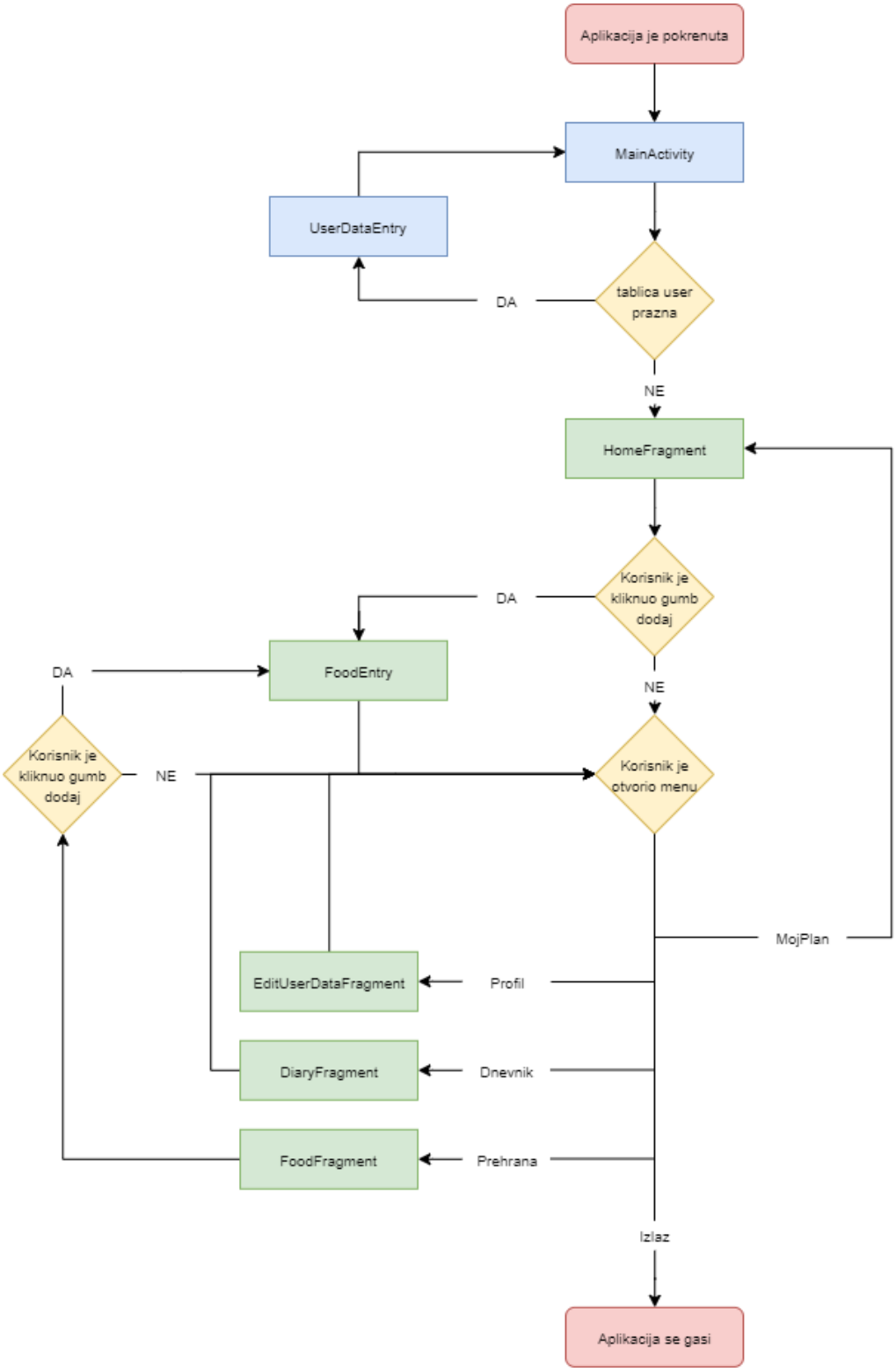
5. Razvoj aplikacije „MojPlan“

Aplikacija je smišljena kao dnevnik i podsjetnik za praćenje dnevnog unosa kalorija. Aplikaciju sam nazvao „MojPlan“. Zamišljena da pomaže korisniku u ostvarivanju svog cilja vezanog za kontrolu tjelesne težine. Aplikacija prati dnevni unos kalorija te se prilagođava korisniku ovisno o njegovoj dobi, visini, težini, spolu te razini fizičke aktivnosti i cilju koji želi postići. Na slici 18 prikazan je flowchart aplikcije kreiran u besplatnom web alatu Draw.io. Aktivnosti su prikazane plavom, dok su fragmenti prikazani zelenom bojom.

5.1. Aktivnost unosa podataka i početan zaslon

Kod prvog pokretanja aplikacije, odnosno kada je broj korisnika u bazi podataka nula, kao što je prikazano na slici 19, pokrenuti će se aktivnost pod nazivom UserDataEntry gdje se od korisnika traže osobni podaci koji su potrebni za izračun indeksa tjelesne težine i potrebnog dnevnog unosa kalorija.

Aktivnost koja će se pokrenuti sadrži klasu pod nazivom UserDataEntry. UserDataEntry klasu sam postavio kao podklasu klase AppCompatActivity. AppCompatActivity je klasa u Androidu koja sadrži važne metode koje koristim unutar klasa UserDataEntry i MainActivity te je dio Android support programske knjižnice. Aplikacija će zatražiti od korisnika unos imena, visine, težine i godina, odabir spola, razine aktivnosti i definiranje cilja. U slučaju da korisnik ne unese neku od zatraženih informacija, aplikacija će obavijestiti korisnika porukom te označiti prvo prazno polje, te provjerava ako su unosi za visinu, težinu i godine u određenom rangu. U nastavku na slici 20 i 21 je prikazana UserDataSubmit metoda koja provjerava i unosi podatke o korisniku u tablicu user.



SLIKA 18 - FLOWCHART APLIKACIJE

```
//ako je broj redova u tablici food == 0, ispuni tablicu slijedećim vrijednostima
int nrow = db.countAllRecordsNotes( table: "food");
if (nrow == 0) {
    db.add( table: "food",  fields: "_id, food_name, food_description, food_calories",  values: "NULL, 'Šunka', '100 grama narezak', '350'");
    db.add( table: "food",  fields: "_id, food_name, food_description, food_calories",  values: "NULL, 'Jaje', 'Tvrdo kuhano', '200'");
    db.add( table: "food",  fields: "_id, food_name, food_description, food_calories",  values: "NULL, 'Čokolada', '100 grama', '300'");
    db.add( table: "food",  fields: "_id, food_name, food_description, food_calories", values: "NULL, 'Zobene pahuljice', '50 grama', '100'");
}
```

SLIKA 19 - POKRETANJE AKTIVNOSTI ZA UNOS INFORMACIJA

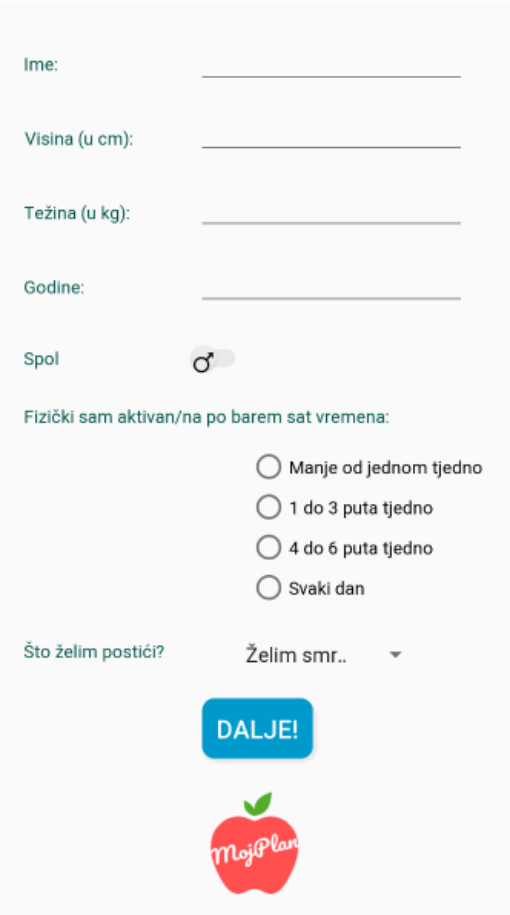
```
public void userDataSubmit() {
    //1. Ime
    EditText editTextName = findViewById(R.id.name_edittext);
    //2. Visina
    EditText editTextHeight = findViewById(R.id.height_edittext);
    //3. Težina
    EditText editTextWeight = findViewById(R.id.weight_edittext);
    //4. Godine
    EditText editTextAge = findViewById(R.id.age_edittext);
    //5. Spol
    Switch switchGender = findViewById(R.id.gender_switch);
    //6. Aktivnost
    RadioGroup radioGroupAL = findViewById(R.id.radioGroup);
    RadioButton lvl1 = findViewById(R.id.radButton1);
    RadioButton lvl2 = findViewById(R.id.radButton2);
    RadioButton lvl3 = findViewById(R.id.radButton3);
    RadioButton lvl4 = findViewById(R.id.radButton4);
    //7. Cilj
    Spinner spinnerGoal = findViewById(R.id.spinner_goal);
    //provjera ako ima praznih unosa
    if (TextUtils.isEmpty(editTextName.getText().toString())) {
        editTextName.setError("Ovo polje ne može biti prazno!");
        Toast.makeText(UserDataEntry.this, "Unesi svoje ime!", Toast.LENGTH_LONG).show();
        return;
    } else if (TextUtils.isEmpty(editTextHeight.getText().toString())) {
        editTextHeight.setError("Ovo polje ne može biti prazno!");
        Toast.makeText(UserDataEntry.this, "Unesi svoju visinu!", Toast.LENGTH_LONG).show();
        return;
    } else if (TextUtils.isEmpty(editTextWeight.getText().toString())) {
        editTextWeight.setError("Ovo polje ne može biti prazno!");
        Toast.makeText(UserDataEntry.this, "Unesi svoju težinu!", Toast.LENGTH_LONG).show();
        return;
    } else if (TextUtils.isEmpty(editTextAge.getText().toString())) {
        editTextAge.setError("Ovo polje ne može biti prazno!");
        Toast.makeText(UserDataEntry.this, "Unesi svoje godine!", Toast.LENGTH_LONG).show();
        return;
    } else if (!lvl1.isChecked() && !lvl2.isChecked() && !lvl3.isChecked() && !lvl4.isChecked()) {
        Toast.makeText(UserDataEntry.this, "Odaberi razinu tjelesne aktivnosti!", Toast.LENGTH_LONG).show();
        return;
    }
    String stringName = editTextName.getText().toString();
    int heightInt = Integer.parseInt(editTextHeight.getText().toString());
    int weightInt = Integer.parseInt(editTextWeight.getText().toString());
    int ageInt = Integer.parseInt(editTextAge.getText().toString());
    int genderInt = 0;
    int activityLevelInt = 0;
    int goalInt;
```

SLIKA 20 - USERDATASUBMIT 1/2

```
    //Provjera ako su unosi u određenom rangu
    if ((heightInt < 100) || (heightInt > 250)) {
        Toast.makeText(UserDataEntry.this, "Nepravilan unos - visina!", Toast.LENGTH_LONG).show();
    }
    else if ((weightInt < 30) || (weightInt > 250)) {
        Toast.makeText(UserDataEntry.this, "Nepravilan unos - težina!", Toast.LENGTH_LONG).show();
    }
    else if (ageInt < 18 && ageInt > 0) {
        Toast.makeText(UserDataEntry.this, "Aplikacija nije namijenjena mladima od 18!", Toast.LENGTH_LONG).show();
    }
    else if (ageInt > 100) {
        Toast.makeText(UserDataEntry.this, "Nepravilan unos - godine!", Toast.LENGTH_LONG).show();
    }
    else if (ageInt <= 0) {
        Toast.makeText(UserDataEntry.this, "Godine ne mogu biti 0 ili negativan broj!", Toast.LENGTH_LONG).show();
    }
    else {
        if (switchGender.isChecked()) genderInt = 1;       //FEMALE - ON == 1
        if (lvl1.isChecked()) activityLevelInt = 0;
        else if (lvl2.isChecked()) activityLevelInt = 1;
        else if (lvl3.isChecked()) activityLevelInt = 2;
        else if (lvl4.isChecked()) activityLevelInt = 3;
        //String StringGoal = spinnerGoal.getSelectedItem().toString();
        goalInt = spinnerGoal.getSelectedItemPosition();
        //BMR i BMI metode
        int BMR = calculateBMR(heightInt, weightInt, ageInt, genderInt, activityLevelInt, goalInt);
        double BMI = calculateBMI(heightInt, weightInt);
        //Datum
        String stringDate = getCurrentDate();
        // Unos u bazu podataka - tablica user
        SQLiteAdapter sqLiteAdapter = new SQLiteAdapter(this);
        sqLiteAdapter.open();
        String stringNameSQL = sqLiteAdapter.checkEntry(stringName);
        int heightIntSQL = sqLiteAdapter.checkEntry(heightInt);
        int weightIntSQL = sqLiteAdapter.checkEntry(weightInt);
        int ageIntSQL = sqLiteAdapter.checkEntry(ageInt);
        int genderIntSQL = sqLiteAdapter.checkEntry(genderInt);
        int activityLevelIntSQL = sqLiteAdapter.checkEntry(activityLevelInt);
        int goalIntSQL = sqLiteAdapter.checkEntry(goalInt);
        int BMRSQL = sqLiteAdapter.checkEntry(BMR);
        double BMISQL = sqLiteAdapter.checkEntry(BMI);
        String stringDateSQL = sqLiteAdapter.checkEntry(stringDate);
        String stringInput = "NULL, " + stringNameSQL + "," + heightIntSQL + "," + weightIntSQL + "," + ageIntSQL + "," + genderIntSQL + "," +
activityLevelIntSQL + "," + goalIntSQL + "," + BMRSQL + "," + BMISQL + "," + stringDateSQL;
        sqLiteAdapter.add("user", "_id, user_name, user_height, user_weight, user_age, user_gender, user_activity, user_goal, user_BMR, user_BMI,
user_date", stringInput);
        sqLiteAdapter.close();
        //Natrag na MainActivity
        startActivity();s
    }
} // method userDataSubmit()
```

SLIKA 21 - USERDATASUBMIT 2/2

Na slici 22 je prikazan dizajn aktivnosti koja se pokreće kada je tablica user prazna.



SLIKA 22 - DIZAJN ZA AKTIVNOST UNOS PODATAKA

U slučaju da su svi podaci pravilno upisani, pritiskom na gumb “dalje!” aplikacija se vraća u MainActivity klasu unutar koje se nakon provjere ako je korisnik u tablici pokreće fragment HomeFragment. Na slici 23 je prikazan kod za pokretanje init() metode ako se korisnik nalazi u tablici user:

```
if (nrow != 0) {
      init();
   }

}

private void init() {

   Fragment fragment = null;
   Class fragmentClass = null;
   fragmentClass = HomeFragment.class;
   try {
      fragment = (Fragment) fragmentClass.newInstance();
   } catch (Exception e) {
      e.printStackTrace();
   }

   FragmentManager fragmentManager = getSupportFragmentManager();
   fragmentManager.beginTransaction().replace(R.id.frameLayout1, fragment).commit();
}
```

SLIKA 23 - INIT

HomeFragment je dio početnog zaslona aplikacije, unutar kojeg korisnik ima nekoliko dostupnih mogućnosti i informacija. Dizajn fragmenta je prikazan na slici 25.



SLIKA 24 – DIZAJN POČETNOG ZASLONA

Na vrhu aplikacije ispod datuma i prikaza koliko kalorija smije unijeti na temelju svog cilja i izračunatog BMR-a, korisnik može dodati novi unos, odnosno naziv hrane, iznos kalorija te dodatan opis hrane koju je unio u svoj organizam. Osim što se ovisno o unosu broj na vrhu smanjuje, korisniku se dinamički ispunjava i kružni „loading bar“ koji promijeni boju kada korisnik prekorači svoj dnevni unos. Unutar fragmenta HomeFragment poziva se više metoda koje su bitne za rad aplikacije. Metoda greetingsUser() na alatnoj traci pozdravlja korisnika. AutomateDate() ažurira datum na zaslonu, displayList() prikazuje u listi što smo unijeli za trenutni datum, dok updateDiary() metoda ažurira promjene. Unutar deleteList() i foodEaten() metoda nalaze se „listener“-i za gumbe kojima dodajemo ili brišemo elemente iz liste. Na slici 25 je prikazana onActivityCreated() metoda koja se poziva kod kreiranja fragmenta:

```
@Override
public void onActivityCreated(@Nullable Bundle savedInstanceState) {
    super.onActivityCreated(savedInstanceState);
    //Pozdravlja korisnika u naslovu
    greetingsUser();
    //Metoda za prikaz današnjeg datuma
    automateDate();
    //Button za dodati hranu
    foodEaten();
    //Button za izbrisati dnevni unos
    deleteList();
    //Prikaz što smo danas dodali
    displayList();
    //Ažuriranje promjena u dnevnik
    updateDiary();
}
```

SLIKA 25 - ONACTIVITYCREATED

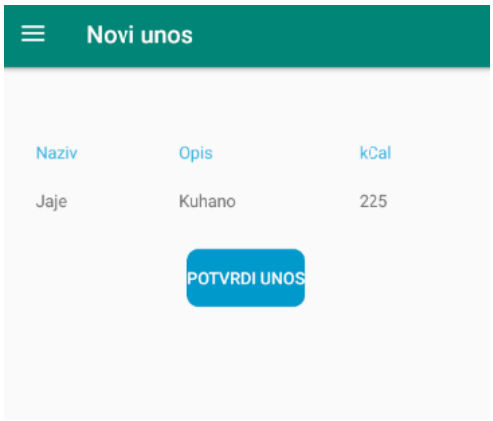
Na slici 27 prikazana je lista iz koje korisnik odabire hranu koju želi unijeti u dnevnik. Pritiskom na određeni element sa liste otvara se novi prozor u kojem korisnik potvrđuje svoj unos kao što je i prikazano na slici 28. Pritiskom na gumb “dodaj” poziva se metoda foodEntry() koja korisnika premješta na novi fragment, foodEntry klasu. Kod je prikazan na slici 26.

```
private void foodEntry() {
    Fragment fragment = null;
    Class fragClass = null;
    fragClass = FoodEntry.class;
    try {
        fragment = (Fragment) fragClass.newInstance();
    } catch (Exception e) {
        e.printStackTrace();
    }
    FragmentManager fragmentManager = getActivity().getSupportFragmentManager();
    fragmentManager.beginTransaction().replace(R.id.frameLayout1, fragment).commit();
}
```

SLIKA 26 – FOODENTRY



SLIKA 27 - NOVI UNOS



SLIKA 28 - POTVRDA UNOSA

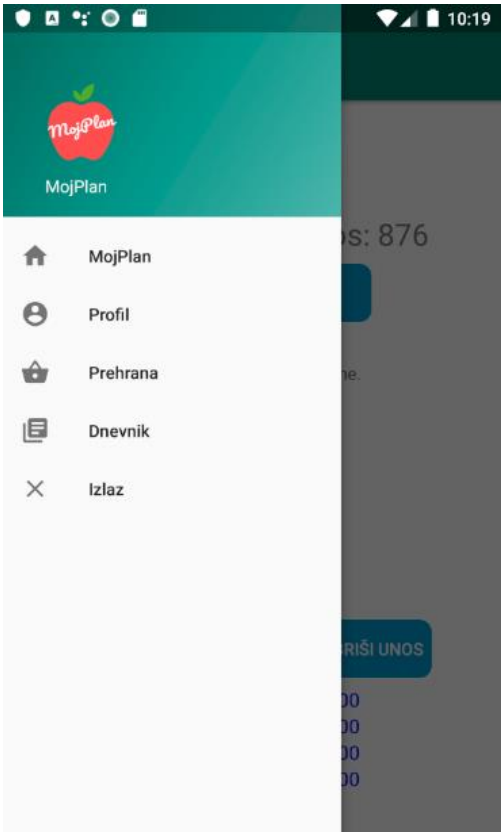
Potvrdom unosa podaci se spremaju u bazu podataka, te se prikazuju u obliku liste. U slučaju da korisnik želi unijeti hranu koja se ne nalazi na listi, u gornjem desnom krutu pritiskom na ikonu korisnik može dodati novu hranu. Hrana mora sadržavati sve tri informacije koje se traže od korisnika, a to su naziv, iznos kalorija i opis hrane. Pritiskom na gumb "izbriši unos" korisnik briše unos u dnevniku za trenutni dan. Na slici 29 prikazan je dizajn za unos novog elementa u bazu podataka.



SLIKA 29 - UNOS NOVOG ELEMENTA U BAZU PODATAKA

5.2 Aktivnost unosa podataka i početan zaslon

U bočnom izborniku koji je prikazan na slici 30 korisnik ima opciju izmjene svojih osobnih informacija, dodavanja i brisanja hrane iz baze podataka koja mu je prikazana u obliku liste, te ima pristup dnevniku koji prati unos kalorija po danima. Ako se korisnik nalazi unutar neke druge aktivnosti, dostupan je i povratak na početnu aktivnost, te izlaz iz aplikacije.



SLIKA 30 - OPCIJE MENU-A

U nastavku na slici 31 je prikazan kod koji pokreće odabrani fragment ili zatvara aplikaciju unutar navigacije menu-a:

```
@SuppressWarnings("StatementWithEmptyBody")
@Override
public boolean onNavigationItemSelected(MenuItem item) {
    //spremi korsinikov odabir u fragmentClass
    int id = item.getItemId();

    Fragment fragment = null;
    Class fragmentClass = null;

    if (id == R.id.nav_home) {
        fragmentClass = HomeFragment.class;
    } else if (id == R.id.nav_profile) {
        fragmentClass = EditUserDataFragment.class;
    } else if (id == R.id.nav_food) {
        fragmentClass = FoodFragment.class;
    } else if (id == R.id.nav_diary) {
        fragmentClass = DiaryFragment.class;
    } else if (id == R.id.nav_exit) {
        finish();
        System.exit(0);
    }

    //item -> fragment
    try {
        fragment = (Fragment) fragmentClass.newInstance();
    } catch (Exception e) {
        e.printStackTrace();
    }

    //Mijenjanje prikaza na odabrani fragment
    FragmentManager fragmentManager = getSupportFragmentManager();
    try{
        fragmentManager.beginTransaction().replace(R.id.frameLayout1, fragment).commit();
    } catch (Exception e) {
        e.printStackTrace();
        Toast.makeText(this, "Error: " + e.toString(), Toast.LENGTH_LONG).show();
    }

    DrawerLayout drawer = findViewById(R.id.drawer_layout);
    drawer.closeDrawer(GravityCompat.START);
    return true;
}
```

SLIKA 31 - ONNAVIGATIONITEMSELECTED

5.3 Izmjena osobnih informacija

Korisnik ima mogućnost izmjene osobnih informacija. Odabirom druge opcije menu-a pod nazivom profil otvoriti će se fragment EditUserDataFragment vrlo sličan početnoj aktivnosti za unos podataka. Klikom na gumb „potvrdi“ ažurirati će se promjene u tablici user. Dizajn fragmenta prikazan je na slici 33. Slika 32 prikazuje kod za ispis poruke koja će se pojaviti ovisno o uspješnom ili neuspješnom ažuriranju polja iz tablice user:

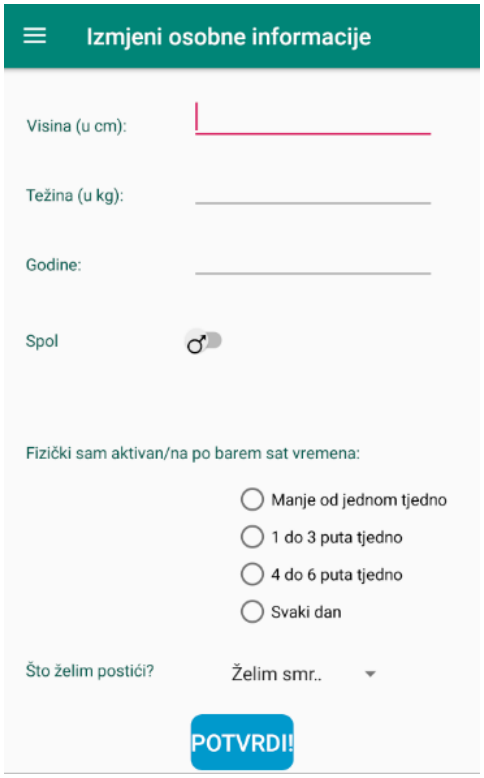
```
String stringInput[] = new String[] {heightIntSQLstring, weightIntSQLstring, ageIntSQLstring, genderIntSQLstring, activitySQLstring, goalIntSQLstring,
BMRSQLstring, BMISQLstring};

String fields[] = new String[] {
      "user_height",
      "user_weight",
      "user_age",
      "user_gender",
      "user_activity",
      "user_goal",
      "user_BMR",
      "user_BMI"
};

long id = 1;

try {
   sqLiteAdapter.updateFields("user", "_id", id, fields, stringInput);
   Toast.makeText(getActivity(), "Informacije USPJEŠNO ažurirane!", Toast.LENGTH_SHORT).show();
}catch (StringIndexOutOfBoundsException e) {
   e.printStackTrace();
   Toast.makeText(getActivity(), "informacije NISU uspješno ažurirane!", Toast.LENGTH_SHORT).show();
}
sqLiteAdapter.close();
```

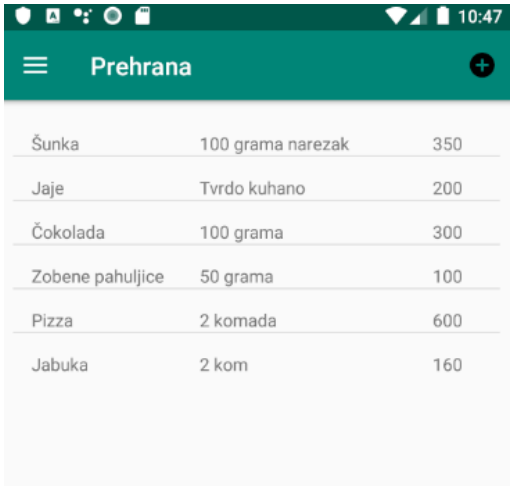
SLIKA 32 - KOD ZA ISPIS PORUKE KOD AŽURIRANJA KORISNIKA



SLIKA 33 - IZMJENA OSOBNIH INFORMACIJA

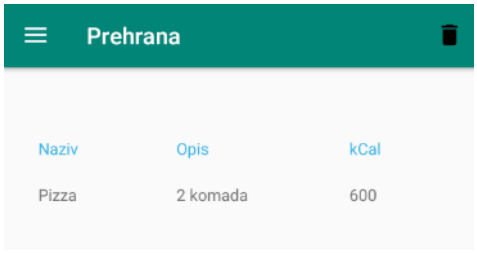
5.4 Prikaz prehrane

Odabirom treće opcije menu-a pod nazivom prehrana otvoriti će se FoodFragment fragment koji prikazuje svu hranu koja se nalazi u tablici food u obliku TextView liste elemenata. Taj fragment korisniku daje uvid u prehranu, omogućuje dodavanje novih elemenata u tablicu te sva hrana koja se nalazi na ovoj listi se može dodati u dnevnik prehrane u HomeFragment fragmentu. Dizajn je prikazan na slici 34.



SLIKA 34 - PRIKAZ LISTE PREHRANE

Odabirom nekog elementa otvoriti će se novi pogled sa zasebnim prikazom hrane, te mogućnosti brisanja tog elementa iz tablice kao što je i prikazano na slici 35.



SLIKA 35 - ZASEBAN PRIKAZ

Na slici 36 je prikazan kod FoodCursorAdapter klase koji služi za dinamičko dodavanje novih TextView elemenata unutar fragmenta, te na slici 37 prikazana je metoda populateList() koja poziva FoodCursorAdapter:

```
package com.example.easyfit;

import android.content.Context;
import android.database.Cursor;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.CursorAdapter;
import android.widget.TextView;

public class FoodCursorAdapter extends CursorAdapter {
    public FoodCursorAdapter(Context context, Cursor cursor) {
        super(context, cursor, 0);
    }

    @Override
    public View newView(Context context, Cursor cursor, ViewGroup parent) {
        return LayoutInflater.from(context).inflate(R.layout.fragment_food_list_item, parent, false);
    }

    @Override
    public void bindView(View view, Context context, Cursor cursor) {
        // Polja za ispuniti
        TextView textViewListName = view.findViewById(R.id.textView_list_name);
        TextView textViewListNumber = view.findViewById(R.id.textView_list_Number);
        TextView textViewListDescription = view.findViewById(R.id.textView_sub_name);

        // Izvlačenje vrijednosti iz kursora
        int getID = cursor.getInt(cursor.getColumnIndexOrThrow("food_calories"));
        String getName = cursor.getString(cursor.getColumnIndexOrThrow("food_name"));
        String getDescription = cursor.getString(cursor.getColumnIndexOrThrow("food_description"));

        // Ispuna polja sa vrijednostima
        textViewListName.setText(getName);
        textViewListNumber.setText(String.valueOf(getID));
        textViewListDescription.setText(String.valueOf(getDescription));
    }
}
```

SLIKA 36 - FOODCURSORADAPTER

```
public void populateList() {

    //baza podataka
    SQLiteAdapter sqLiteAdapter = new SQLiteAdapter(getActivity());
    sqLiteAdapter.open();

    //hrana (food tablica)
    String fields[] = new String[]{
            "_id",
            "food_name",
            "food_description",
            "food_calories"
    };
    listCursor = sqLiteAdapter.select("food", fields);

    //Odabir ListView-a za popuniti
    ListView items = getActivity().findViewById(R.id.list_view_food);

    //Postavljanje CursorAdaptera
    FoodCursorAdapter foodCursorAdapter = new FoodCursorAdapter(getActivity(), listCursor);

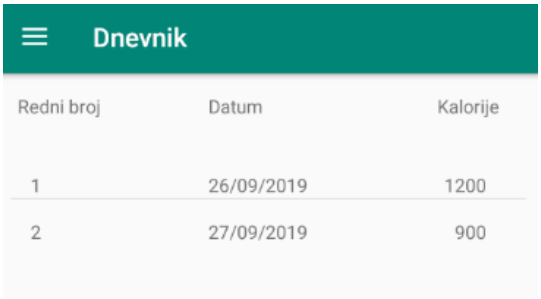
    //Spajanje cursor adaptera na ListView
    items.setAdapter(foodCursorAdapter);

    //onclicklistener
    items.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
            listItemClicked(i);
        }
    });
    sqLiteAdapter.close();
}
```

SLIKA 37 - POPULATELIST

5.5 Dnevnik prehrane

Četvrta opcija menu-a je prikaz iznosa ukupnog broja kalorija po danima. Lista se dinamički ažurira sa trenutnim vrijednostima iz dnevnika prehrane iz HomeFragment fragmenta. Dizajn je prikazan na slici 38. Korištena je metoda za dinamičko dodavanje redaka populateList() i DiaryCursorAdapter klasa vrlo slični metodi iz FoodFragment fragmenta te FoodCursorAdapter klasi.

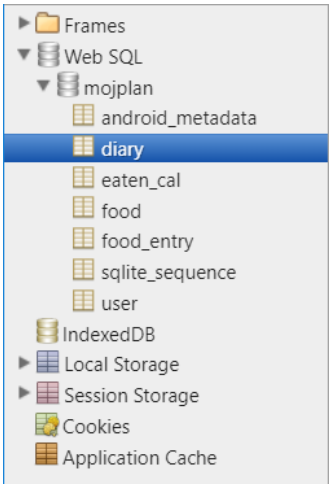


SLIKA 38 - PRIKAZ UNOSA KALORIJA PO DANIMA

Zadnja opcija menu-a pod nazivom izlaz jednostavno prekida sve procese te uredno zatvara aplikaciju koristeći ugrađene metode finish() i System.exit(0).

5.6 Baza podataka

Za izradu baze podataka koristio sam SQLite programsku knjižnicu unutar Android Studio razvojnog okruženja. Kao što je prikazano na slici 39, aplikacija sadrži šest kreiranih tablica u kojima se pohranjuju podaci bitni za pravilan rad aplikacije, uz tablice android_metadata i sqlite_sequence.



SLIKA 39 - TABLICE BAZE PODATAKA

Tablice unutar baze podataka su:

- tablica user - pohranjuje sve podatke koje korisnik unosi unutar aktivnosti UserDataEntry tijekom prvog pokretanja aplikacije, datum kada su podaci uneseni ili ažurirani te rezultate izračuna za BMR i BMI. Atributi tablice su primarni ključ _id, zatim user_name, user_height, user_weight, user_age, user_gender, user_activity, user_goal, user_BMR, user_BMI i user_date. Svi podaci osim primarnog ključa i imena mogu se ažurirati od strane korisnika unutar fragmenta EditUserDataFragment pomoću metode updateFields().

- tablica food - pohranjuje svu hranu koju korisnik dodaje unutar fragmenata FoodFragment i FoodEntry. Atributi tablice su primarni ključ _id, food_name, food_description i food_calories. Tablica se koristi za prikaz liste unutar FoodFragment fragmenta.

- tablica food_entry - pohranjuje svaki zasebni unos, odnosno koristi se unutar metode za potvrdu dnevnog unosa hrane addFoodToDB() metode unutar FoodEntry fragmenta tako da dodaje datum kada je unos kreiran. Atributi tablice su primarni ključ _id, fe_name, fe_date, fe_food_id i fe_calories. Tablica se koristi

i unutar HomeFragment fragmenta u updateDiary() metodi koja ubacuje ili ažurira podatke po datumima u tablicu diary.

- tablica eaten_cal - tablica je nastala iz potrebe da se svakim novim unosom ažurira ukupan iznos kalorija, odnosno zbroj kalorija po danima u dnevniku prehrane. Tablica se koristi unutar updateCalDisplay() metode u HomeFragment fragmentu. Argumenti tablice su primarni ključ _id, eaten_cal_id, eaten_cal_date i eaten_cal.

- tablica diary - tablica se ažurira posljednjim unosom za svaki dan iz tablice eaten_cal unutar updateDiary() metode u HomeFragment fragmentu, te se koristi za prikaz elemenata u dnevniku prehrane unutar DiaryFragment fragmenta.

Na slici 40 prikazana je metoda onCreate() klase SQLiteAdapter koja kreira pet tablica, dok su na slici 41 prikazane neke od najvažnijih metoda. U nastavku je prikazana relacijska shema baze podataka sa relacijama diary, eaten_cal, food_entry, food i user te njihovim pripadajućim atributima:

diary (id, diary_cal_date, diary_cal)
eaten_cal (id, eaten_cal_id, eaten_cal_date, eaten_cal)
food_entry (id, fe_name, fe_date, fe_food_id, fe_calories)
food (id, food_id, food_name, food_description, food_calories)
user (id, user_name, user_height, user_weight, user_age, user_gender, user_activity, user_goal, user_BMR, user_BMI, user_date)

```
@Override
public void onCreate(SQLiteDatabase sqLiteDatabase) {
  try {
    //Kreiranje tablica
    sqLiteDatabase.execSQL("CREATE TABLE IF NOT EXISTS diary (" +
        " _id INTEGER PRIMARY KEY AUTOINCREMENT," +
        " diary_cal_date DATE," +
        " diary_cal INT);");

    sqLiteDatabase.execSQL("CREATE TABLE IF NOT EXISTS eaten_cal (" +
        " _id INTEGER PRIMARY KEY AUTOINCREMENT," +
        " eaten_cal_id INT," +
        " eaten_cal_date DATE," +
        " eaten_cal INT);");

    sqLiteDatabase.execSQL("CREATE TABLE IF NOT EXISTS food_entry (" +
        " _id INTEGER PRIMARY KEY AUTOINCREMENT," +
        " fe_name VARCHAR," +
        " fe_date DATE," +
        " fe_food_id INTEGER," +
        " fe_calories DOUBLE);");

    sqLiteDatabase.execSQL("CREATE TABLE IF NOT EXISTS food (" +
        " _id INTEGER PRIMARY KEY AUTOINCREMENT, " +
        " food_id INT, " +
        " food_name VARCHAR," +
        " food_description VARCHAR," +
        " food_calories DOUBLE);");

    sqLiteDatabase.execSQL("CREATE TABLE IF NOT EXISTS user (" +
        " _id INTEGER PRIMARY KEY AUTOINCREMENT, " +
        " user_name VARCHAR," +
        " user_height INT," +
        " user_weight INT," +
        " user_age INT," +
        " user_gender INT," +
        " user_activity INT," +
        " user_goal INT, " +
        " user_BMR DATE, " +
        " user_BMI DOUBLE, " +
        " user_date DATE);");

  }
  catch (SQLException e) {
    e.printStackTrace();
  }
}
```

SLIKA 40 - ONCREATE

```
    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {
        sqLiteDatabase.execSQL("DROP TABLE IF EXISTS food");
        sqLiteDatabase.execSQL("DROP TABLE IF EXISTS food_entry");
        sqLiteDatabase.execSQL("DROP TABLE IF EXISTS eaten_cal");
        sqLiteDatabase.execSQL("DROP TABLE IF EXISTS user");
        sqLiteDatabase.execSQL("DROP TABLE IF EXISTS diary");
        onCreate(sqLiteDatabase);
    }
}
public SQLiteAdapter open() throws SQLException {
    db = dbHelper.getWritableDatabase();
    return  this;
}
public void close() {
    dbHelper.close();
}
//unos podataka u tablicu
public void add(String table, String fields, String values) {
    db.execSQL("INSERT INTO " + table + "(" + fields + ") VALUES (" + values + ")");
}
//broj unosa
public int countAllRecordsNotes(String table) {
    Cursor mCount = db.rawQuery("SELECT COUNT(*) FROM " + table + "", null);
    mCount.moveToFirst();
    int count = mCount.getInt(0);
    mCount.close();
    return count;
}
//metoda za provjeru unosa (osiguravamo da podaci koje unosimo nisu opasni za bazu)
public String checkEntry(String value) {
    boolean isNumeric = false;
    try{
        Double myDouble = Double.parseDouble(value);
        isNumeric = true;
    }
    catch (NumberFormatException e) {
        System.out.println("Could not parse " + e);
    }
    if (!isNumeric)
        if (value != null && value.length() > 0){
            value = value.replace("\\", "\\\\");
            value = value.replace("'", "\\'");
            value = value.replace("\0", "\\0");
            value = value.replace("\n", "\\n");
            value = value.replace("\r", "\\r");
            value = value.replace("\"", "\\\"");
            value = value.replace("\\x1a", "\\Z");
        }
    }
    value = "'" + value + "'";
    return value;
}
```

SLIKA 41 - METODE KLASE SQLITEADAPTER

5.7 Budući razvoj aplikacije

Tijekom svog rada često sam dolazio do različitih ideja kako poboljšati izgled i funkcionalnosti aplikacije. Osim redizajna korisničkog sučelja eksperimentiranjem sa temama, bojama i fontovima te korištenjem različitih vanjskih programskih knjižnica koje bi poboljšale vizualni prikaz te korisničko iskustvo, razmatrao bih implementaciju slijedećih funkcionalnosti u svoju aplikaciju:

- Korištenje pogleda dobrodošlice pri pokretanju aplikacije.
- Kod zasebnog prikaza hrane, ubacivanje detaljnog prikaza nutritivnih vrijednosti.
- Implementiranje detaljnih nutritivnih informacija o hrani kao što su vitamini, minerali, proteini, ugljikohidrati, masnoće i šećeri u aplikaciju.
- Ubacivanje poveznica na informativna web-sjedišta o važnosti pravilne prehrane i vježbanja.
- Implementiranje obavijesti svakih 4-8 sati kako bi podsjetili korisnika da unese svoju prehranu u dnevnik.
- Izmjena dnevnika na način da se prikazuju datumi sa mini ikonama koje bi označavale uspješan ili prekoračen dnevni unos umjesto obične liste.
- Osim mogućnosti unosa hrane, korisnik bi mogao i unositi dane kada je vježbao.
- Brisanje elemenata iz liste prehrane pokretom prsta ulijevo ili udesno, umjesto pritiska na gumb.
- Prevođenje aplikacije na strane jezike.

6. Zaključak

Od svog predstavljanja Android je postao najpopularniji mobilni operativni sustav. Android pruža puno veći izbor uređaja u odnosu na svoje konkurente. Velik izbor uređaja različitih dizajna i specifikacija omogućuje kupcima da prilagode uređaje svom budžetu što rezultira da si gotovo svatko može priuštiti Android telefon. Ekskluzivna priroda Appleovih proizvoda u jasnoj je suprotnosti s inkluzivnom prirodom Androida. Još jedna od prednosti je ta da se Android uređaji neprimjetno integriraju s Googleovim uslugama. Velika baza korisnika i širok spektar proizvođača koji proizvode Android uređaje mogu samo ubrzati daljnje inovacije.

U ovom diplomskom radu razvijena je android aplikacija namijenjena odraslim osobama. Aplikacija je uspješno izrađena te testirana i spremna za objavu na mrežnoj trgovini. Aplikacija daje uvid u količinu energije koju unosimo u sebe te na taj način daje nam mogućnost da reguliramo svoju težinu na zdrav način. Cijeli proces razvoja odvijao se u nekoliko faza kao što su razvijanje teorijske podloge Java programskog jezika i Android Studio razvojnog okruženja, osmišljavanje funkcionalnosti aplikacije, kreiranje testnih verzija te otkrivanje pogreški u kodu i ispravljanje. Prikazana je arhitektura i povijest Androida, te su pojašnjene srodne tehnologije koje su korištene prilikom izrade. Funkcionalnosti aplikacije su omogućene unutar deset klasa: dvije aktivnosti, dva adaptera za ispunu ListView-a, klasa za SQLite bazu podataka i pet fragmenata. Ovaj rad ne bi bio moguć bez pomoći programske zajednice koja je vrlo aktivna na forumima kao što su „Stackoverflow“, „CodeRanch“, „Reddit“ i ostali. Kao ključnu vrijednost ovog rada naveo bih stečena znanja o razvoju mobilnih aplikacija koja ću moći primijeniti u budućem radu.

Literatura

[1] L. G. J. W. Li Ma, Research and Development of Mobile Application for Android Platform , https://bit.ly/33sEpX2, datum pristupa dokumentu: 20.08.2019.

[2] Developer.android: Platform Arhitecture, https://developer.android.com/guide/platform, datum pristupa dokumentu: 20.08.2019.

[3] R. S. Nisarg Gandhewar, Google Android: An Emerging Software Platform For Mobile Devices, https://bit.ly/2Mcw7gE, datum pristupa dokumentu: 20.08.2019.

[4] A. Thakur, Difference between Dalvik and ART runtimes in Android, https://bit.ly/2noCouk https://bit.ly/2noCouk, datum pristupa dokumentu: 20.08.2019.

[5] D. H. R. Nimodia C., Android Operating System, https://bit.ly/31nXyYi, datum pristupa dokumentu: 21.08.2019.

[6] Wikipedia: Android version history, https://en.wikipedia.org/wiki/Android_version_history, datum pristupa dokumentu: 22.08.2019.

[7] M. L. Murphy, The Busy Coder's Guide to Android Development.

[8] Codecademy: what is an ide, https://www.codecademy.com/articles/what-is-an-ide, datum pristupa dokumentu: 23.08.2019.

[9] M. v. Drongelen, Android Studio Cookbook.

[10] Developer.android: Meet Android Studio.

https://developer.android.com/studio/intro,

datum pristupa dokumentu: 23.08.2019.

[11] M. Butler, Android: Changing The Mobile Landscape,

https://bit.ly/2YW5a6D,

datum pristupa dokumentu: 23.08.2019.

[12] A. S. Palak Khanna, Google Android Operating System: A Review,

https://bit.ly/2yRYyru,

datum pristupa dokumentu: 23.08.2019.

[13] S. Brahler, »Analysis of the Android Architecture,

https://bit.ly/1WkhoOF,

datum pristupa dokumentu: 23.08.2019.

Tablice

Tablica 1 – Usporedba razvojnih okruženja za razvoj Android aplikacija 12

Slike

Slika 1 - tržišni udio mobilnih operativnih sistema 3
Slika 2 - broj aplikacija na najpoznatijim mrežnim trgovinama 4
Slika 3 - arhitektura androida 5
Slika 4 - relativan broj uređaja koji pokreću određenu verziju platforme android 9
Slika 5 - primjer označavanja sintakse 11
Slika 6 - prozor dobrodošlice 14
Slika 7 - prvi korak, konfiguracija projekta 14
Slika 8 - drugi korak, konfiguracija projekta 15
Slika 9 - prozor android projekta 16
Slika 10 - klient - server arhitektura i sqlite arhiterktura 18
Slika 11 - gradle dependencies 19
Slika 12 - Integracija stetho alata u aplikaciju 19
Slika 13 - chrome devtools 20
Slika 14 - web sql 21
Slika 15 - životni ciklus aktivnosti 23
Slika 16 - notifikacija za dodjelu dozvole aplikaciji 24
Slika 17 - životni ciklus fragmenta 25
Slika 18 - flowchart aplikacije 27
Slika 19 - pokretanje aktivnosti za unos informacija 28
Slika 20 - userDataSubmit 1/2 28
Slika 21 - userDataSubmit 2/2 29
Slika 22 - dizajn za aktivnost unos podataka 30
Slika 23 - init 31
Slika 24 – dizajn početnog zaslona 31
Slika 25 - onActivityCreated 32
Slika 26 – foodEntry 33
Slika 27 - novi unos 33
Slika 28 - potvrda unosa 33
Slika 29 - unos novog elementa u bazu podataka 34
Slika 30 - opcije menu-a 35
Slika 31 - onNavigationItemSelected 36
Slika 32 - kod za ispis poruke kod ažuriranja korisnika 37
Slika 33 - izmjena osobnih informacija 38
Slika 34 - prikaz liste prehrane 39
Slika 35 - zaseban prikaz 39
Slika 36 - FoodCursorAdapter 40
Slika 37 - populateList 41
Slika 38 - prikaz unosa kalorija po danima 42
Slika 39 - tablice baze podataka 43
Slika 40 - onCreate 45
Slika 41 - metode klase SQLiteAdapter 46