Razvoj mobilne aplikacije za android "Kalorina"

Mesić, Karolina

Sveučilište u Rijeci - Odjel za informatiku

Dvopredmetni studij: njemački jezik i informatika

Karolina Mesić

Razvoj mobilne aplikacije za Android

"Kalorina"

Završni rad

Mentorica: doc.dr.sc. Marina Ivašić Kos

Rijeka, 2017.

IZJAVA

kojom ja, Karolina Mesić, JMBAG: 0009060147 student/ica Filozofskog fakulteta Sveučilišta u Rijeci, kao autor/ica završnog rada s naslovom: „Razvoj mobilne aplikacije za Android – Kalorina“.

Izjavljujem da sam završni rad izradio/la samostalno pod mentorstvom prof. dr. sc. Marine Ivašić Kos. U radu sam primijenio/la metodologiju znanstveno istraživačkog rada i koristio/la literaturu koja je navedena na kraju rada. Tuđe spoznaje, stavove, zaključke, teorije i zakonitosti koje sam izravno ili parafrazirajući naveo/la u radu citirao/la sam i povezao/la s korištenim bibliografskim jedinicama sukladno odredbama Pravilnika o završnom radu Filozofskog fakulteta u Rijeci. Rad je pisan u duhu hrvatskog jezika.

Student/ica

_____

Karolina Mesić

Sadržaj

1. Uvod

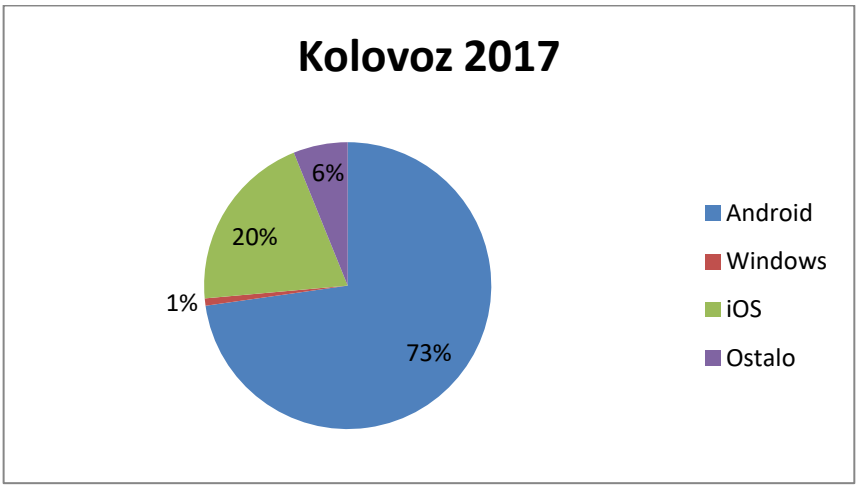
Živimo u 21.stoljeću, dobu u kojem Internet i tehnologija određuju ritam života. U taj okvir ulaze tableti, pametni telefoni i drugi mobilni uređaji. Mobilne uređaje pokreću različiti operativni sustavi, a prema analitičkoj tvrtki *StatCounter* trenutačno su najpoznatiji Googleov Android, Microsoftov Windows te Appleov iOS.[1] Ovaj rad bazirat će se isključivo na Android operativnom sustavu i Google Play Trgovini koja je u sustavu Androida.

Funkcija mobilnih uređaja je komunikacija između korisnika u obliku tekstualne ili glasovne poruke, pa sve do pokretanja raznih aplikacija. Mnoge mobilne aplikacije dobijemo već instalirane na samom uređaju, a sve ostale možemo preuzeti s raznih Web trgovina besplatno ili uz plaćanje. Aplikacije koje su napravljene za operativni sustav Android, odnosno za Google Play tržište, nisu kompatibilne za iOS ili Windows operativne sustave. Zanimljiv je podatak da se pojedine aplikacije proizvode samo za određeni operativni sustav, dok za drugi ne. Uspoređujući pojedine Android aplikacije s identičnima na iOS-u, vidljiva je značajna razlika u kvaliteti. Kao razlog se navodi postojanje velikog broja Android uređaja slabije kvalitete i lošijih performansi koji ne mogu pokrenuti zahtjevnije verzije aplikacije. Istraživanja su pokazala da se značajno razliku i navike korisnika Android uređaja od onih koji koriste iOS. Dokazano je da Android korisnici puno češće deinstaliraju aplikacije, točnije, 200% više.[2] Razvoj mobilne Aplikacije za Android tema ovog završnog rada. „Kalorina“ je aplikacija koja pamti i regulira dnevni unos kalorija s čime smanjuje rizik od pretjeranog ili premalog unosa hrane u organizam. Razvijena je pomoću programa Android Studio u programskom jeziku Java. Procese i razvitak aplikacije „Kalorina“ detaljnije će biti opisane u nastavku rada.

[1] http://gs.statcounter.com/os-market-share#monthly-201703-201703-map (6.9.2017)
[2] http://www.bug.hr/vijesti/korisnici-android-uredaja-cesce-deinstaliraju-apli/157070.aspx (8.9.2017.)

2. Operativni sustav Android

Mobilni uređaji koriste različite operativne sustave, a najpoznatija su IOS, Android i Windows. Prema sljedećem grafu možemo utvrditi da je Android operativni sustav trenutačno najkorišteniji operativni sustav projektiran za mobilne uređaje (Slika 1.)



Slika 1 Prikazuje tržišni udio operativnog sustava za mobitele diljem svijeta - kolovoz 2017. godine (http://gs.statcounter.com/os-market-share/mobile/worldwide#monthly-201703-201703-map, 6.9.2017)

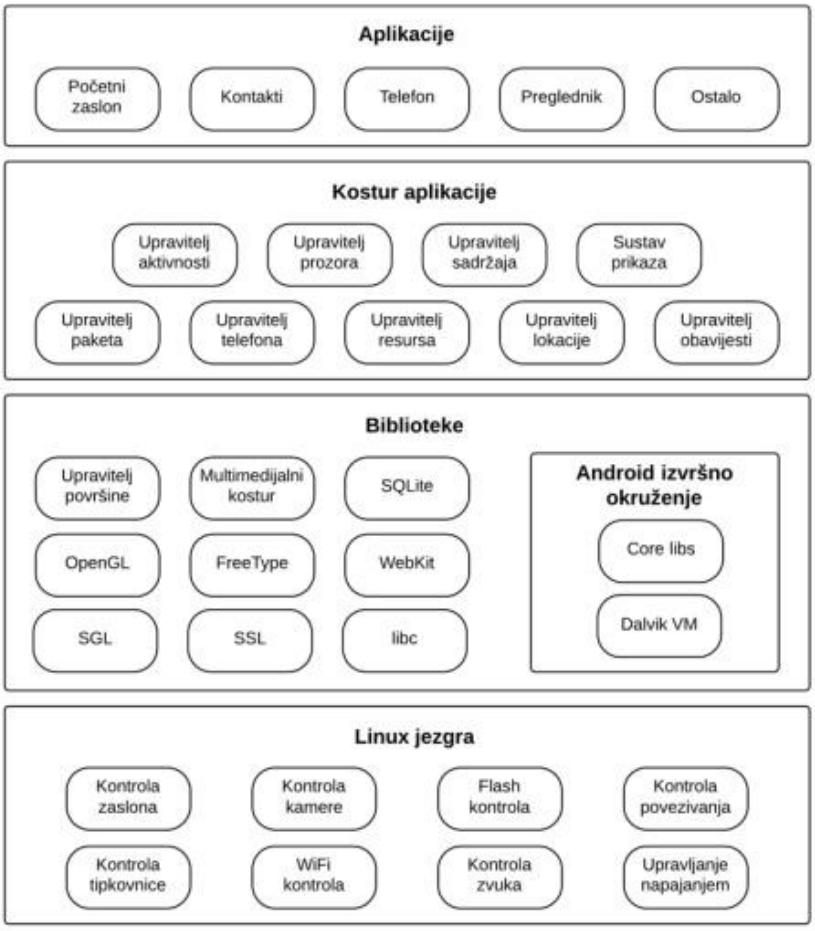
Tvrtka Android datira iz 2003. godine, a dvije godine kasnije kupuje je Google te se od tad zalaže za razvoj Android platforme.[3] Godine 2007. osnovan je savez Open Handset Alliance koji uključuje proizvođače mobilnih uređaja, operatere mobilnih mreža, softverske kompanije i brojne druge tvrtke. Godine 2008. savez predstavlja Android s SDK verzijom 1.0., a krajem godine na tržište dolazi Googleov prvi telefon tajvanskog proizvođača HTC. Tijekom godina su objavljene nove verzije SDK-a kao i novi pametni telefoni.

Android OS je izrađen na Linuxu, koji je primjer otvorenog izvornog koda (Gargenta, 2011.,str.7). Također pruža brz i jednostavan razvoj mobilnih aplikacija. Mogu se koristiti razni programi, poput alata otvorenog koda Eclipse, ali za izradu ove mobilne aplikacije korišten je Android Studio o kojem će biti riječi u nastavku rada.

[3] http://www.connect.de/ratgeber/android-geschichte-des-erfolgs-1491130.html (6.9.2017)

2.1. Arhitektura operativnog sustava Android

Već je spomenuto da je Android open source koji se temelji na Linxu. Sljedeća slika prikazuje glavne komponente Android platforme.(Slika 2.)



Slika 2 Dio arhitekture operativnog sustava Android
(Gargenta, 2011.,str.7 , 6.9.2017.)

Korištenje Linux jezgre Androidu omogućuje iskorištavanje ključnih sigurnosnih značajki i omogućuje proizvođačima uređaja da razviju hardverske upravljačke programe za dobro poznatu jezgru. Postoji mnogo razloga zašto je baš Linux izabran kao temelj Android platformi, a neki od najvažnih su sigurnost, prenosivost, privatnost i činjenica da podržava gotove sve programske jezike.[4]

Za uređaje s Android verzijom 5.0 (API razina 21) ili novijom verzijom koristi se Android Runtime (ART). ART i njegov prethodnik Dalvik su kompatibilni virtualni strojevi

[4] https://itsfoss.com/linux-better-than-windows/ (7.9.2017.)

posebno razvijeni za Android platformu.[5] ART također izvršava dex. datoteke pa aplikacije razvijene za Dalvik trebaju raditi i prilikom pokretanja s ART-om.

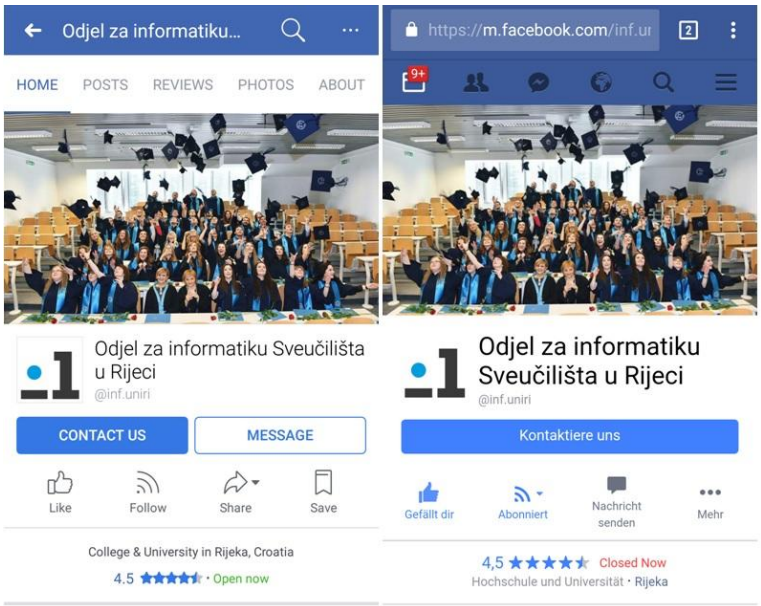
U Androidu Runtimeu nalaze se bibliotke koje programerima omogućuju programiranje aplikacija. Delvik i ART se nalaze na Linux jezgri, a Linux između ostalog upravlja zaslonom, tipkovnicom, kamerom, zvukom i dr. (Slika 2.). Izvorne biblioteke, kao što su Webkit, SQLite, Apache Harmony, Open Gl i druge, napisane su u C i C++ programskom jeziku i služe za nativne aplikacije o kojima će poslije biti riječ. U kosturu aplikacije se nalaze Java biblioteke i mnoge usluge upravljanja (upravitelj aktivnosti, paketa, sadržaja, prozora itd.) s kojima aplikacija ima mogućnost povezivanja. Na kraju su sve razvijene aplikacije koje se nalaze na samom uređaju ili koje se mogu preuzeti s raznih Web trgovina besplatno ili uz plaćanje. Najpoznatija trgovina je Googleov Android Market.

[5] https://source.android.com/devices/tech/dalvik/ (7.9.2017.)

3. Vrste Android aplikacija

Postoje tri osnovne vrste aplikacija, a to su nativne aplikacije, hibridne aplikacije i web aplikacije.[6] Nativne aplikacije su aplikacije koje su programirane za određeni mobilni uređaj, pametni telefon, tablet itd. Njihova instalacija se vrši preko Web trgovine, poput Google Playa za Android ili App Store za iOS uređaje.

Web aplikacija je web stranica prilagođena za mobilne uređaje kojoj se pristupa putem Internet preglednika na samom uređaju.[7] Koriste web preglednik koji je napisan pomoću HTML-a, CSS-a ili JavaScript. Hibridne mobilne aplikacije su kombinacija nativne aplikacije i online web aplikacije.[8] Baš kao što su i web stranice na Internetu, hibridne mobilne aplikacije su izrađene web tehnologijama kao što su HTML, CSS i Java Script. Ključna razlika između običnih aplikacija i hibridnih aplikacija je u tome što su hibridne aplikacije smještene unutar izvorne aplikacije koja koristi WebView mobilnu platformu.[9] WebView je prozor preglednika koji je obično konfiguriran za pokretanje cijelog zaslona (full screen). To omogućuje pristup opcijama uređaja kao što su kamera, kontakti i još mnogo toga.



Slika 3 Prikaz nativne aplikacije (lijevo) i mobilne web aplikacije(desno).

[6] https://thinkmobiles.com/blog/popular-types-of-apps/ (7.9.2017.)
[7] http://www.hdonweb.com/mobiteli/nativna-aplikacija-mobilna-web-stranica (7.9.2017.)
[8] http://www.evolva.hr/hr/razvoj_hibridnih_mobilnih_aplikacija.html (7.9.2017.)
[9] https://developer.telerik.com/featured/what-is-a-hybrid-mobile-app/ (7.9.2017.)

3.1. Aplikacije po kategorijama

Odabir aplikacije se temelji na korisničkim željama i potrebama. Između čak 33 ponuđene kategorije na Google Play Trgovini obuhvaćena su sva društvena i znanstvena područja.[10] Od Knjiga, Hrane, Obrazovanja, Umjetnosti, Glazbe pa sve do Financija, Navigacije i Kućanstva – za svakoga će se nešto pronaći.

Statistika koja je prikazana na slici 4. prikazuje kategoriju koja je najpopularnija na tržištu, a to je kategorija Mobilnih igara. Preuzimanje mobilnih igara u prvom kvartalu 2017.godine znatno se povećala u usporedbi na prvi kvartal u 2016.godini. U današnje vrijeme je to normalno jer su igre korisnicima izvor zabave i opuštanja. Veoma su „zarazne" što korisnike potiče da ih igraju dok ih ne prijeđu. Postoji velik broj podvrsta igara; akcijske, arkadne, edukativne, strategijske, simulacije te mnoge druge. Neke od poznatih mobilnih igara su Clash of Clans, Temple Run, Angry Birds, Candy Crush i Solitaire. Prateći trend rasta preuzimanja igara, možemo zaključiti da će se on u budućnosti samo povećavati. Potrebno je napomenuti da je razvijanje mobilnih igara trenutačno najprofitabilnije područje.

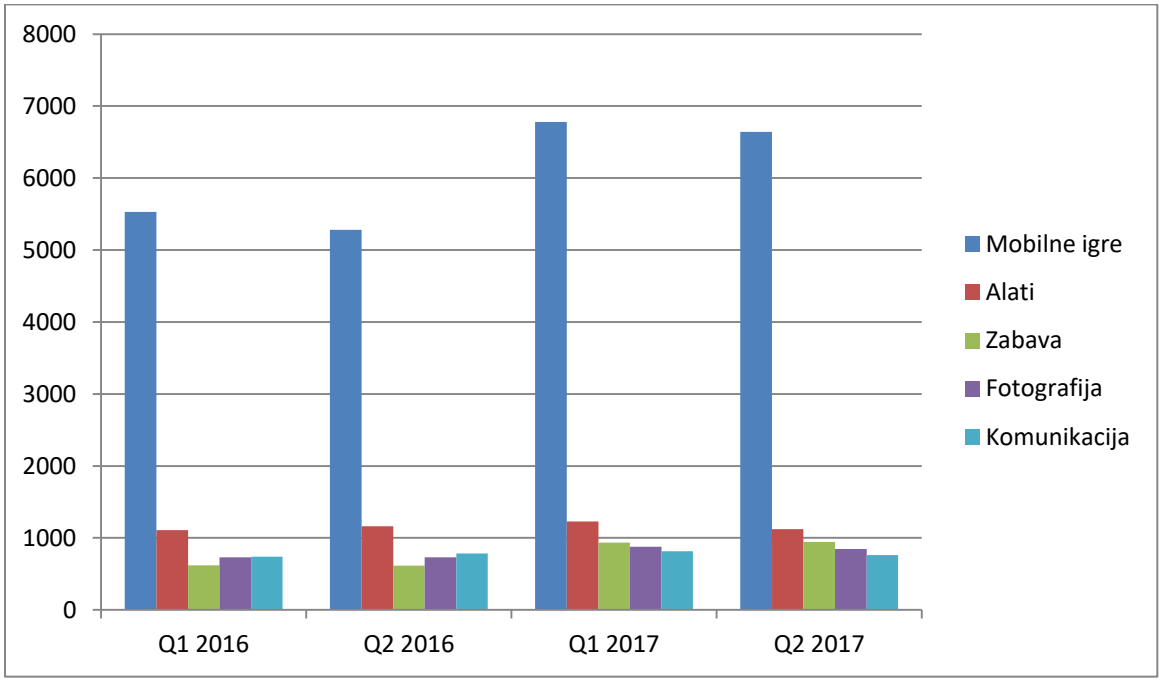
U kategoriju Alati, kao što i sama riječ govori, ulaze aplikacije s kojima se obavlja neka radnja. Dizajnirane su kao oruđe koje korisniku putem mobilnog telofona izvršava ili pomaže pri izvršavanju nekog zadatka. Takve aplikacije se najčešće koriste za optimizaciju mobitela, čišćenje od virusa i nepotrebnih datoteka, prevođenje jezika, mjerenje brzine interneta, praćenje vremenske prognoze i slično. Najviše se preuzimaju Clean Master, Google Translate, Speedtest by Ookla i Virus Cleaner.

Često se kategorije Igrice i Zabava poistovjećuju, no to nije ista kategorija unatoč tome što što im je svrha slična. Pod Zabavu pripadaju aplikacije koje se koriste u slobodno vrijeme i uglavnom su šaljive prirode. Uključuju gledanje filmova, korištenje Emoji-ja, slušanje radija, čitanje viceva, dodavanje raznih efekata glasovima i slikama. Neke od poznatijih su Netflix, Talking Tom Cat, Make Me Bald, Twitch, Bitmoji. Danas, kada svaki mobilni uređaj ima kameru i sve fotografije završavaju na društvenim mrežama, dobro je pogledati kategoriju Fotografija. Ona uglavnom sadrži aplikacije za instantno uređivanje fotografija i pravljenje kolaža. Najviše se koriste filteri koji fotografiji daju profesionalniji

[10] https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/(8.9.2017.)

izgled bez dodatnih uređivanja. Tu spadaju Photo Editor, Selfie Camera, Boomerang for Instagram, Retrica, Collage Maker Pic Grid i mnoštvo ostalih sličnog sadržaja.

Aplikacije iz kategorije Komunikacija su, slobodno možemo reći, promijenile način komuniciranja ljudi diljem svijeta. Zbog njih su nekad najkorištenije SMS poruke gotovo zanemarene. Aplikacije poput Vibera, Messengera, WhatsApp-a i Skypa zahtjevaju pristup internetu, no daljnje korištenje je posve besplatno. Upravo na tome se zasniva njihov uspjeh. Zbog privlačnog dizajna i jednostavnosti sve češće ih koriste i starije generacije. U istu kategoriju ulaze i preglednici (browseri) Mozilla, Chrome, Opera te aplikacije za pristup e-mailu kao što je Gmail i Yahoo Mail.



Slika 4 Prikaz statistike najpopularnijih kategorija na Google Play trgovini (https://www.statista.com/statistics/256772/most-popular-app-categories-in-the-google-play-store/, 7.9.2017)

U prikazanom grafikonu nema kategorije Društvene aplikacije iz razloga što on prikazuje aplikacije koje su najviše preuzimane u posljednje dvije godine (Slika 4.) No, pogledamo li sveukupan broj preuzimanja od začetka Google Play Trgovine (2008.)[11], uz Komunikacije, glavnu ulogu vode Društvene aplikacije. Spomenemo li aplikacije kao što su Facebook, Instagram, Snapchat, LinkedIn, možemo uvidjeti njihovu veličinu i utjecaj u modernom svijetu.

[11] https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/

Aplikacije se u Google Play Trgovini, osim po kategorijama, dijele po cijeni. Postoje aplikacije koje su besplatne i one koje se plaćaju. Besplatne aplikacije su u većini slučajeva nešto jednostavnije građe i funkcije, dok su plative aplikacije u pravilu zahtjevnijih specifikacija, boljih performansi i sadrže mnogo više dodatnih opcija koje se u besplatnima ne nalaze. Ipak, i besplatne aplikacije imaju svoje kvalitete i svrhu što dokazuje njihov puno veći broj preuzimanja od onih koje se plaćaju.

3.2. Top pet aplikacija u kategoriji "Zdravlje i kondicija"

Kategoriju „Zdravlje i kondicija" čini mnoštvo aplikacija kojima je svrha obrazovati i usmjeriti korisnika prema zdravom stilu života. Koristeći te aplikacije, korisnici spoznaju važnost zdrave prehrane i redovite tjelovježbe. Mogućnosti su neograničene, od aplikacija koje pružaju kompletne programe treninga do dijetnih koje organiziraju mjesto i vrijeme sljedećeg obroka. Kod većine aplikacija je radi jednostavnije upotrebe, princip rada i praćenja prilagođen kućnim uvjetima. Trenutno najpopularnije aplikacije iz navedene kategorije opisati ću u nastavku.

Menstrualni Kalendar je aplikacija za praćenje menstrualnog ciklusa i ovulacije. Osim što prati menstrualni ciklus, žene mogu bilježiti simptome, raspoloženja, spolne odnose, tjelesnu težinu, temperaturu, bilješke o kontracepcijskim tabletama koje koriste itd. Korisna opcija je da svaki dan može izračunati vjerojatnost nastanka trudnoće. Aplikacija je napravljena u obliku osobnog dnevnika.

Runtastic Running & Fitness Tracker predstavlja odličan alat za mjerenje kilometraže i vremena koje nam je bilo potrebno za preći tu udaljenost. Prvenstveno je namijenjen sportašima koji prate promjene svojih rezultata no često ga koriste i amateri u svrhu bilježenja osobnih vremena koje su prohodali ili istrčali. Napredne opcije aplikacije čine ju vrlo korisnom. Koristi GPS koji nakon trčanja detaljno prikazuje rutu koju smo prošli na mapi, a tijekom trčanja glasovnom signalizacijom najavljuje pređene kilometre. Ukoliko aplikaciju nadoplatimo, prelazimo na Premium verziju. U njoj su dostupne opcije kao što je plan treninga, brojni edukativni filmovi, tjedno izvješće odrađenog treninga te dodatne statistike.

Peludna prognoza omogućuje svakodnevno praćenje prisutnosti pojedinih alergena. Veoma je korisna za alergičare koji praćenjem uputa i mogu izravno utjecati na svoje

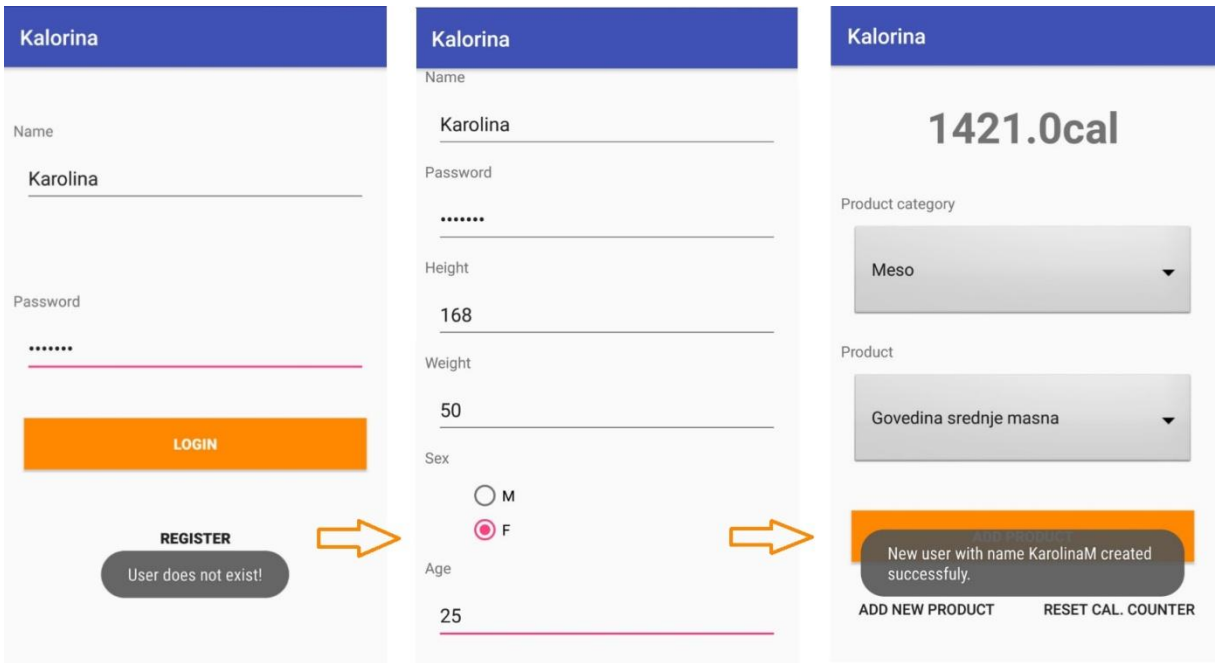
zdravstveno stanje. Sadrži podatke o najčešćim alergenim biljkama te putem interneta prihvaća mjerenja većine gradova u Hrvatskoj. Na taj način prognozira peludnu prognozu za sljedeća dva dana.

Sleep Cycle Alarm Clock je inteligentna vrsta budilice. Koristi se tako da se mobilni uređaj postavi na krevet, pored tijela, koji svojim osjetljivim senzorom akcelerometrom mjeri svaki naš pokret tijekom sna. Po intenzitetu pokreta prepoznaje našu trenutnu fazu sna. Aplikacija tako analizira naš san te nas budi u najlakšoj fazi sna – opuštene i odmorne. Po želji se mogu zatražiti izvješća i grafovi za svaku prospavanu noć, za detaljnu usporedbu sna u određenom razdoblju.

4. Korištenje aplikacije „Kalorina“

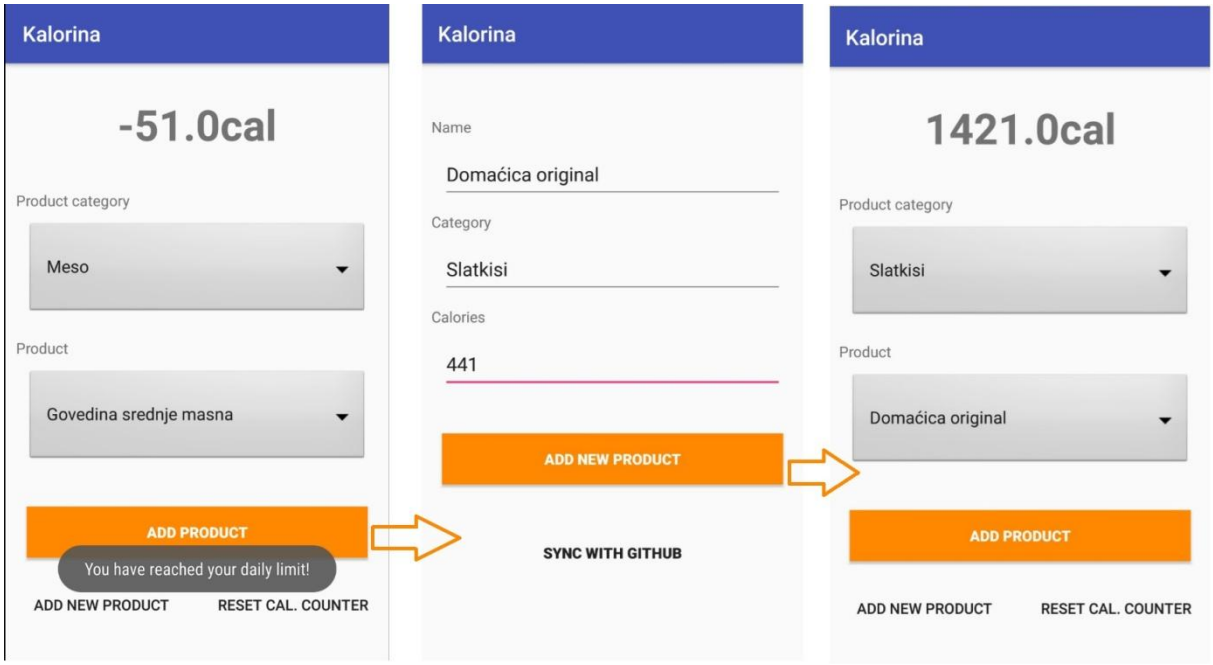
Aplikacija „Kalorina“ koju sam razvila u okviru završnog rada pripada kategoriji aplikacija za Zdravlje i kodniciju. Aplikacija usmjerava korisnike s povećanom ili smanjenom tjelesnom težinom prema postizanju boljeg fizičkog i psihičkog stanja po uzoru na već provjerene, poznatije aplikacije. Koristi se na način da se na dnevnoj bazi poslije svakog obroka u aplikaciju upiše količina unesene hrane. Aplikacija kalorije automatski zbraja i na temelju dopuštenog dnevnog unosa pokazuje koliko još hrane smijemo konzumirati želimo li održati ravnotežu. Dopušteni dnevni unos hrane razlikuje se od osobe do osobe te ovisi o spolu, visini i težini.

Nakon instalacije aplikacije na Android uređaju, u popisu aplikacija pojavit će nam se ikona „Kalorina“. Dodirom na ikonu otvara se prozor za Login. Ukoliko smo novi korisnik, potrebno je obaviti registraciju koja je uvjet za daljnje korištenje aplikacije. Upisujemo ime, lozinku, visinu, težinu, spol te godine. Uspješnom registracijom (pojavljuje se obavijest o kreiranju novog korisnika) otvara se glavni prozor s brojem koji označava dopušteni dnevni unos kalorija i kategorije proizvoda. Dopušteni dnevni unos kalorija ovisi o podacima koje smo upisali pri registraciji i razlikuje se za muškarce i žene (Slika 5).



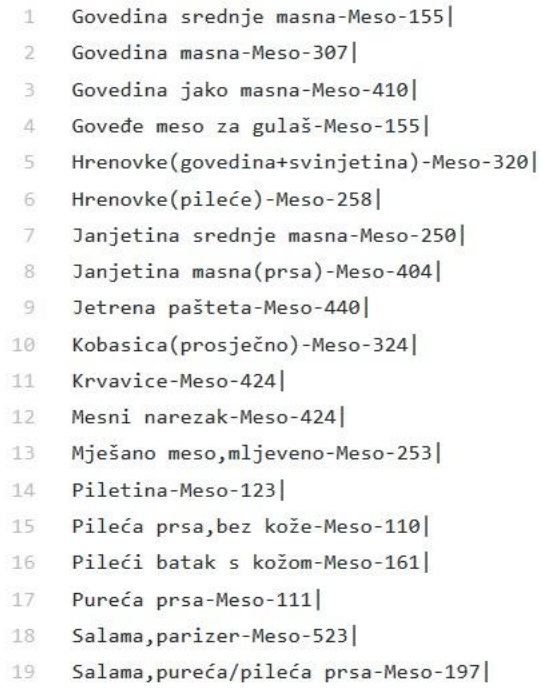
Slika 5 Proces prijave i registracije korisnika

Odabirom konzumiranog proizvoda, aplikacija nam automatski umanjuje dnevni unos kalorija za iznos hranjive vrijednosti odabranog proizvoda to jest količinu kalorija koje proizvod sadrži na 100 grama. Ukoliko u bazi proizvoda neki proizvod nedostaje, odabiremo tipku „Add Product“ koja nam omogućuje dodavanje novog proizvoda i njegove vrijednosti. Novi proizvod ostaje trajno upisan u bazi i ne treba ga svaki put ponovno dodavati (Slika 6).



Slika 6 Proces dodavanje novog proizvoda

U nastavku je prikaz primjera baze proizvoda s platforme GitHub. Aplikacija „Kalorina“ se prilikom pokretanja sinkronizira s bazom GitHub-a te odabirom proizvoda direktno povlači njezine podatke. U bazi su proizvodi označeni rednim brojem, imenom proizvoda, kategorijom u kojoj se nalaze i brojem kalorija koje sadržavaju.



Slika 7 Primjer baze proizvoda s platforme GitHub

5. Implementacija aplikacije "Kalorina"

Cilj aplikacija „Kalorina“ je olakšati korisniku da uravnoteži prehranu i da postigne zdrav način života. Razvijena je pomoću programa Android Studio u programskom jeziku Java.

5.1. Android Studio

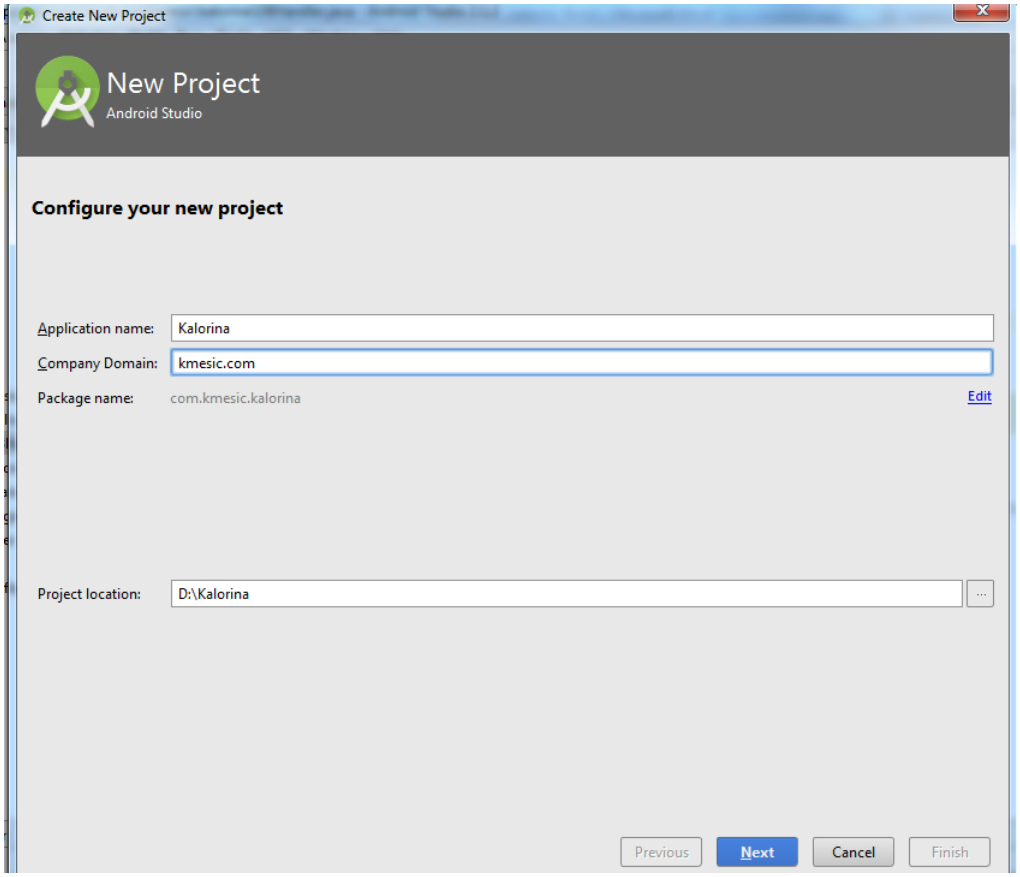
Android Studio je najavljen na Googleovoj I/O konferenciji 2003. godine, a godinu dana kasnije je objavljen Android Studio 1.0 koji od tad postaje službeni razvojni alat za programere Androida.[12] Program Android Studio je baziran na Java razvojnom alatu i to u IntelliJ IDEA Community Edition tvrtke JetBrains koji je dizajniran isključivo za razvoj Android aplikacija.[13]

Prije instalacije ovog programa preporučuje se instalirati najnoviju verziju JDK-a (Java Development Kit) koji je nužan za uspješno funkcioniranje Android Studia. Kako bismo počeli programirati potreban je i Android SDK(Software Development Kit) koji sadrži pomoćne alate koji omogućuju razvoj softwera. Kao i svaki drugi softver tako i Android Studio s vremenom poboljšava svoje performanse te uklanja greške. Svaka nova verzija Androida donosi promjenu u sučelju za programiranje aplikacija API(Application Programming Interface)[14]. Kako bi aplikacija radila na što većem broju mobilnih uređaja potrebna je najniža moguća verzija API-ja.

Pri izradi aplikacije u Android Studiu potrebno je izraditi novi projekt i to na način da kliknemo na File →New →New Project. Otvara se prozor za izradu novog projekta gdje se definiraju početne značajke (Slika 2.). Application name je naziv aplikacije koja se prikazuje korisnicima, što je u ovom slučaju "Kalorina". Company Domain predstavlja domenu, a to je "kmesic.com". Domena će biti dodana na Package name samo obrnutim redoslijedom; "com.kmesic.kalorina". Naziv paketa (Package name) je Javina konstrukcija i mora biti jedinstvena za sve pakete bazirane na Android sustavu. U javi je cijeli izvorni kod organiziran u pakete. Paketi su važni jer između ostalog, zadaju vidljivost objekata između različitih Java

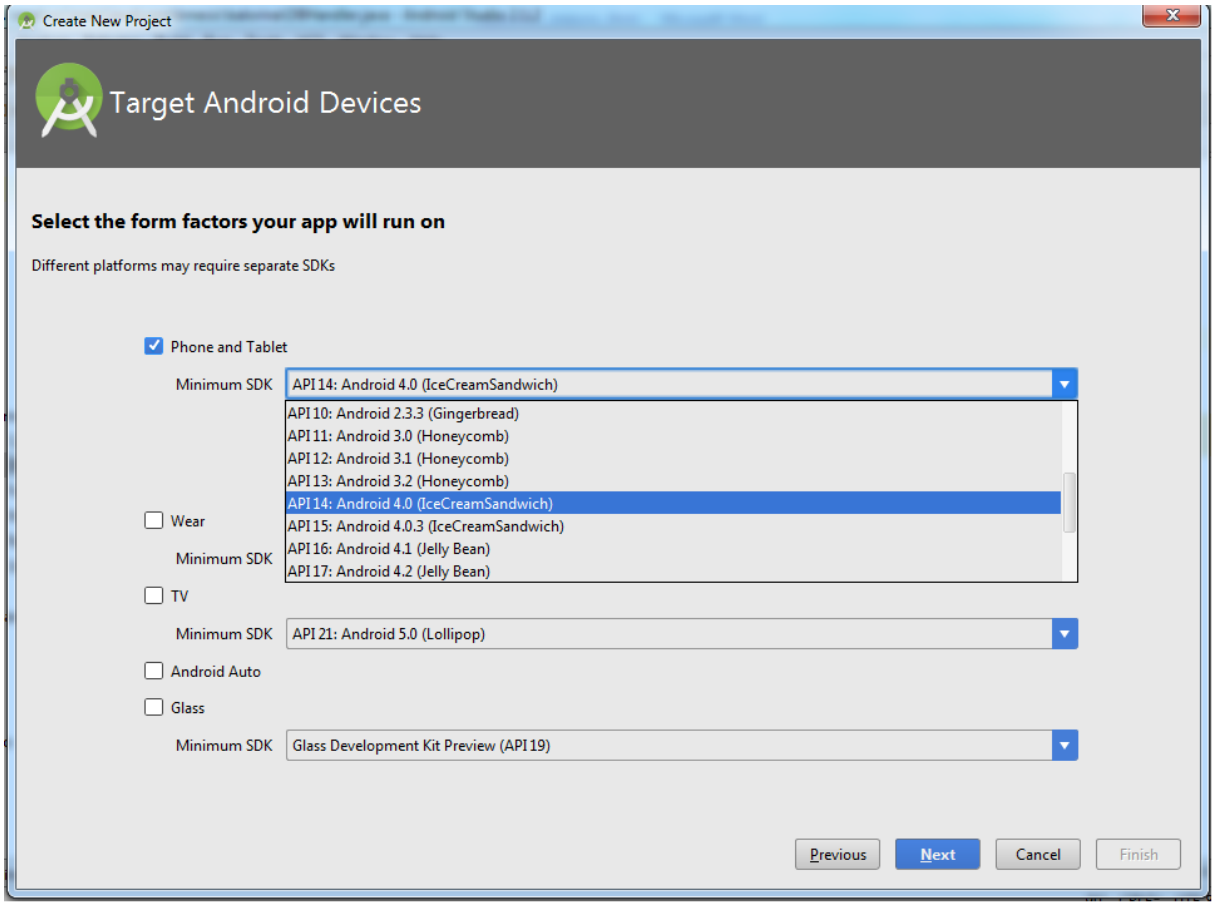
[12] https://venturebeat.com/2014/12/08/google-releases-android-studio-1-0-the-first-stable-version-of-its-ide/(7.9.2017.)
[13] https://android-developers.googleblog.com/2013/05/android-studio-ide-built-for-android.html (20.07.2017.)
[14] https://en.wikipedia.org/wiki/Application_programming_interface (6.9.2017.)

klasa u projektu(Gargenta, 2011.,str.19). Project location predstavlja direktorij u kojem se nalaze projektne datoteke.



Slika 8 Prikazuje zaslon za New Project

Slika 5. prikazuje prozor Target Android Devices koja služi za izgradnju (build target) aplikacije tj. izgradnju za koju Android platformu želimo napraviti projek. Trebamo odabrati verziju SDK (Minimus SDK). Već je spomenuto da je najniža verzija SDK predstavljena razinom API-ja koja je potrebna za izvršavanje aplikacije.



Slika 9 Prikazuje prozor Target Android Devices

Pod Add an Activity to Mobile izabere se Empty Activity to jest praznu aktivnost koju treba imenovati. Aktivnost će biti predstavljena Java klasom pa njeno ime mora započeti velikim slovom, a ukoliko se naziv sastoji od dvije ili više riječi, druga riječ će uvijek početi velikim slovom npr. MainActivity.

Odabirom konfiguracije dobili smo različite datoteke koje su nastale u procesu izrade projekta. Među njima je i xml datoteka manifesta koja povezuje sve komponente aplikacije. (Primjer 1.)

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.kmesic.jonsnow.kalorina">

<application
android:allowBackup="true"
android:icon="@mipmap/ic_launcher"
android:label="@string/app_name"
android:supportsRtl="true"
android:theme="@style/AppTheme">
<activity
android:name="com.kmesic.jonsnow.kalorina.MainActivity"
android:label="Kalorina">
<intent-filter>
<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<activity
android:name="com.kmesic.jonsnow.kalorina.RegistrationActivity"
android:label="Kalorina" />
<activity
android:name="com.kmesic.jonsnow.kalorina.userPageActivity"
android:label="Kalorina" />
<activity android:name="com.kmesic.jonsnow.kalorina.Vegetable_panel" />

</application>

</manifest>
```

Primjer 1. Prikaz AndoridManifest.xml datoteke

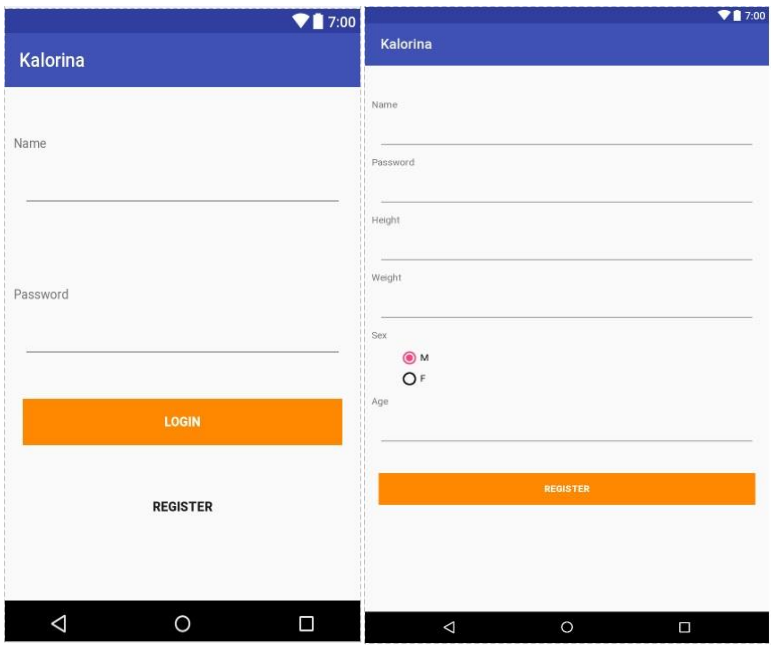
Kako bi pristupili bazi podataka koja se nalazi na hosting servisu GitHub (objašnjenje u natavku rada), potrebno je dopuštenje za pristup Internetu na način da u datoteku AndoridManifest.xml definiramo element za dopuštenje INTERNET.(Primjer 2.)

```
<uses-permission-sdk-23 android:name="android.permission.INTERNET" />
```

Primjer 2. Prikaz elementa za dopuštenje INTERNET

5.2. Registracija i prijava

Prije nego počnemo brojati kalorije koje smo tijekom dana pojeli, moramo proći kroz registraciju, a zatim i prijavu korisnika. Za registraciju i prijavu korisnika se koristi XML koji opisuje izgled korisničkog sučelja, a kroz programiranje se povezujemo s Javom pri čemu deklariramo proces interakcije korisnika s dijelovima korisničkog sučelja (Gargenta, 2011,str.48).



Slika 10 Prikaz prijavu korisnika(lijevo) i registracije korisnika(desno)

Android organizira elemente korisničkog sučelja u rasporede (layouts) i prikaze (views) (Gargenta,2011,str.48). Prikazi predstavljaju sve što možemo vidjeti, a rasporedi organiziraju prikaze. Vrste rasporeda su: LinearLayout, TableLayout, FrameLayout, RelativeLayout, AbsoluteLayout itd. U aplikaciji je korišten LinearLayout jer predstavlja osnovni primjer rasporeda (Primjer 2.). Layout raspoređuje svoje potomke okomito ili vodoravno.[15]

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity"
android:orientation="vertical">
```

[15] 15https://developer.android.com/guide/topics/ui/declaring-layout.html (21.07.2017)

```
<TextView
android:id="@+id/totalcount"
android:layout_marginTop="50dp"
android:layout_marginBottom="10dp"
android:layout_marginLeft="10dp"
android:layout_marginRight="10dp"
android:text="Name"
android:layout_width="wrap_content"
android:layout_height="wrap_content" />
<EditText
android:id="@+id/uname"
android:layout_width="fill_parent"
android:layout_marginLeft="20dp"
android:layout_marginRight="20dp"
android:layout_height="50dp" />

[…]

</LinearLayout>
```

Primjer 3. DioLinearLayout koda za prijavu korisnika korisničkog sučelja.

Svojstva *layout_height* i *layout_width* definiraju koliko prostora element zahtjeva od roditeljskog rasporeda način prikazivanja. Koristili smo *match_parent* i *wrap_content: match_parent* znači da element želi sav raspoloživ prostor koji nudi roditeljski raspored, a *wrap_content*znači da mu je potrebno onoliko prostora koliko zauzima sadržaj koji treba prikazati (Gargenta, 2011,str.54).

Metoda *onCreate()* poziva se kada je aktivnost prvi put napravljena. Slijedi stvaranje grafičkog prikaza i pozivanje XML datoteke *activity_main.* U mapi *res* se nalazi podmapa *layout* u koju se spremaju sve XML datoteke. Metoda *setContentView()* učitava XML datoteku, parsira je te stvara sve potrebne Java objekte koji će odgovarati XML elementima i svukupno generirati cijeli prikaz (Gargenta,2011,str.57). Kada pritisnemo na Login button izvršava se metoda *setOnClickListener().* Vrijednosti koje upisujemo u polja dohvaćamo metodom *getText()* te se dobivena vrijednost pretvara u string. Unesene podatke u polja i*me* i *lozinka* uspoređujemo s bazom da ustanovimo postoji li korisnik s tim podatcima. Ukoliko postoji, vraća se njegov ID. Svaki element u bazi ima svoj ID. To je cijeli broj od 1 pa do beskonačno koji se automatski dodjeljuje svakoj vrijednosti kada se unese u bazu podataka.

Objekt *Toast* pomoću metode *makeText()* prikazuje obavijesti u malom prozorčiću koji se prikazuje na dnu ekrana prilikom logina ili registracije.

Ukoliko pokušajem Logina korisnik već postoji, otvara se idući Layout, a to je *activity_user_page.xml.* Za kretanje između Layouta koristi se klasa *Intent*. Instancira se tako

da joj se pošalju dvije vrijednosti. Prva vrijednost je klasa koja predstavlja Layout na kojem se trenutno nalazimo - klasa *MainActivity*, a druga klasa predstavlja Layout koji želimo otvoriti. Ujedno se šalje ID od korisnika jer kada otvorimo *activity_user_page.xml* moramo znati podatke o korisniku koji se prijavio.

Metoda *registrationListener()* se poziva prilikom pritiska gumba za registraciju. Ona također koristi klasu *Intent* kako bi se otvorio idući Layout, a u ovom slučaju je to *activity_registration.xml.*

```
publicvoidregistrationListener() {
but1 = (Button) findViewById(R.id.but1);
but1.setOnClickListener(newView.OnClickListener() {

@Override
publicvoidonClick(Viewview) {
Intentintent = newIntent(MainActivity.this, RegistrationActivity.class);
startActivity(intent);
        }
    });
}
```

Primjer 4. Prikaz metode *registrationListener().*

Za svaki gumb ili polje za unos podataka postoji već napisana klasa koja definira njihovo ponašanje. Stoga, na početku koda potrebno je definirati varijable. Varijabla tipa EditText, predstavlja varijablu tipa Integer. Potrebno je napraviti i varijablu tipa DBHandler koja je implementirana klasa pomoću koje komuniciramo s bazom podataka.

Nakon što se unesu podatci za registraciju korisnika potrebna je provjera samog korisnika. Ako postoji isti takav korisnik, klasa Toast nas obavijesti s porukom „User Alredy Exist“ („Korisnik već postoji“) ili „New user with name XY created successfully“ („Novi korisnik s imenom XY uspješno kreiran“) ukoliko je upis podataka uspješno obavljeno. Na kraju koda se ponovno koristi *klasa Intent* koja nas vodi do idućeg Layouta, a to je *activity_user_page.xml.*(Primjer 5.)

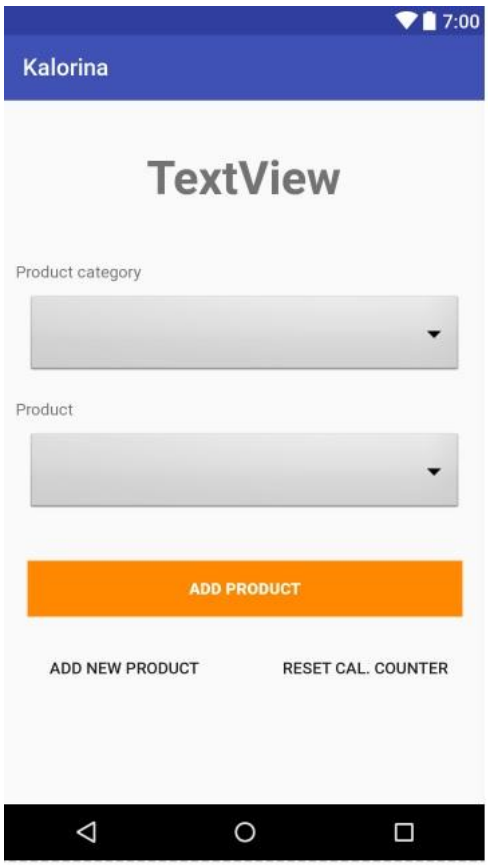
```
db=new DbHandler(RegistrationActivity.this);
int id = db.checkUser(name, password);

if(id != -1)
                {
                    Toast.makeText(RegistrationActivity.this,"User Already
Exist",Toast.LENGTH_SHORT).show();
                }
```

```
else
{
db.addUser(new User(name, password, height, weight, age, sex));
int id_ = db.checkUser(name, password);


                    Toast.makeText(RegistrationActivity.this,"New user with
name " +
                                name + " created
successfully.",Toast.LENGTH_LONG).show();

                    Intent intent = new Intent(RegistrationActivity.this,
userPageActivity.class);

                    Bundle b = new Bundle();
                    b.putString("user_ID", String.valueOf(id_));
                    intent.putExtras(b);
                    startActivity(intent);

                }
            }
        });
    }
```

Primjer 5. Prikaz koda iz klase *RegistrationActivity.java*.

5.2. Klasa User Page

Nakon uspješne registracije ili prijave, klasa *Intent* preusmjerava prema *activity_user_page.xml* tj. prema klasi *userPageActivity.java*.



Slika 11 Prikaz datoteke *activity_user_page.xml.*

Prilikom odabira kategorije („Product Category“) i proizvoda („Product“) koriste se drop downovi koji se nazivaju *Spinner-i.* Polja u Javi imaju duljinu koju nije moguće mijenjati nakon što je polje alocirano, pa ako se dinamički želi mijenjati duljinu polja treba koristiti klasu ArrayList.[16] No, pošto nam treba polje varijabilne dimenzije s parametrom String, koristit ćemo *ArrayList<String>*.

U datoteci *activity_user_page.xml* postoji tri gumba. Jedan za dodavanje proizvoda tj. oduzimanje kalorija od ukupnog BMR-a („Add Product“), drugi za dodavanje novog proizvoda u bazu podataka(„Add new product“) i treći gumb za resetiranje („Reset

[16] https://web.math.pmf.unizg.hr/nastava/rp2/pred4/pred4.html (22.07.2017.)

Cal.Counter“). Iz tog razloga moramo imati i tri *Listenera.* Prije nego se svakom Listeneru pošalje ID korisnika, treba se dohvatiti varijabla koja će se poslati kao parametar. U ovom slučaju je to ID korisnika. Slijedi instanciranje baze i prikazivanje kalorija te se traži BMRtmp kojeg ćemo u klasi *DbHandler.java* detaljnije objasniti.

Prvi listener za prvi dropdown pri odabiru kategorije je *firstSpinnerListener().* Slijedi prikaz dropdownova te dohvaćanje kategorija iz baze podataka. Nakon što smo dohvatili podatke imamo *for* petlju u kojoj punimo prvi dropdown s dohvaćenim kategorijama. Drugi dropdown prikazuje proizvode koji pripadaju odabranoj kategoriji u prvom spinneru.

Metoda *onItemSelected()* dohvaća kategorije odabrene u prvom dropdownu i drugom dropdownu, a zbog estetike je punjenje nove liste napravljeno u zasebnoj funkciji *fillarray().*

Metoda *fillarray()* sadrži metodu *addListenerOnButton()* koja oduzima kalorije i metodu *onClick()* koja dohvaća potrebene vrijednosti iz dropdownova. Odabirom kategorije proizvoda i samog proizvoda u bazi podataka traži se broj kalorija koje ima odabrani proizvod.Iz tog slijedi dohvaćanje kalorija koje su preostale korisniku. Na kraju je dodan string na koji je nadopisan „cal“. Njega je potrebno obrisati kako bi se dobio preostali iznos kalorija.
Ako je dnevni unos hrane 1200cal, moramo dohvatiti taj string, obrisati „cal“ iz njega na način da zamijenimo taj izraz s prazninom, a ostatak pretvorimo u *double*.

U if-else petlji se prikazuje obavijest o tome koliko je kalorija preostalo. To postižemo otvaranjem istog Layouta jer se na taj način osvježava prikaz preostalih kalorija. Dostigne li dnevni unos kalorija nulu ili manje od nule, prikazuje se poruka "You have reached your daily limit!" („Već ste dostignuli svoj dnevni limit!“). (Primjer 8.)

```
StringonlyCals = currentCals.replace("cal", "");
DoublecallsCasted = Double.valueOf(onlyCals);

if(callsCasted<= 0){
Toast.makeText(userPageActivity.this,
"You have reached your daily limit!",
Toast.LENGTH_LONG).show();
                }
else{


Doublenew_cals = Double.valueOf(onlyCals) - Double.valueOf(calories);
Toast.makeText(userPageActivity.this,"Choosen -> " +"\nCategory: "+
String.valueOf(spinner1.getSelectedItem()) +"\nProduct: "+
```

```
String.valueOf(spinner2.getSelectedItem()) +"\nRemcalories: "+
String.valueOf(new_cals),
Toast.LENGTH_LONG).show();

db.updateUserBMRtmp(user_id, new_cals);

Intentintent = newIntent(userPageActivity.this,userPageActivity.class);

Bundleb = newBundle();
b.putString("user_ID", String.valueOf(user_id));
intent.putExtras(b);

startActivity(intent);
                }
```

Primjer 6. Prikaz *if-else* petlje u klasi *userPageActivity.java*.

Ostatak koda prikazuje metode za ostala dva gumba. Jedan listener služi za dodavanje novog proizvoda u bazu podataka koristeći metodu *addListenerOnVegeButton()* kao i metodu *onClick()* koja šalje ID korisnika koji je potreban pri vraćanju iz drugog Layouta na prethodni. Na sličan način funkcionira i drugi gumb za resetiranje kalkulatora koji koristi metode *addListenerOnResetButton()* za resetiranje dnevnog unosa kalorija i *onClick()* koja šalje ID korisnika.

5.3. User/Korisnik

Svaki Java program mora sadržavati najmanje jednu klasu koja zna kako će obaviti određeni posao. Klase sadrže polja koja definiraju stanje objekta, metode koje sadrže izvorni kod i koje definiraju ponašanje objekta. Kada program napravi instancu klase, Java poziva konstruktor klase koji se korisit za pridruživanje vrijednosti varijablama klase. Prilikom deklariranja možemo odrediti i kontrolu pristupa. Java nudi četiri razine pristupa ali mi ćemo navesti samo tri, a to su *Public (*javno*), Protected (*zaštićeno*)* i *Private (*privatno*).*

Ključna riječ *this* se koristi kada se treba referencirati na instance klase iz njene metode. *Public User* je konstruktor klase koji prima parametar za BMR, a to su *name, password, height, weight, age i sex*. Vrijednost BMR-a se ne šalje iz registracijske forme već se računa u klasi *public User* na temelju gore spomenutih parametara.

Za vrijednost BMR-a su nam potrebne dvije varijabe, *bmrtmp* i *bmr*. *Bmrtmp* prima istu vrijednost kao izračunati *bmr*, jer u isto vrijeme treba ostati pohranjena zadnja vrijednost kalorija i ona početna vrijednost koja se računa na početku i koja je također fiksna. Prilikom

prijave u aplikaciju, korisnik smanji svoj dnevni unos kalorija za 200 i zatim se odjavi. Nakon par sati korisnik se vraća i upravo ta zadnja vrijednost treba ostati zapamćena i to u *bmrtmp*. Svaka varijabla u klasi treba imati svoj *getter i setter*. Getter je metoda koja vraća vrijednost varijable, a setter je metoda koja postavlja novu vrijednost u željenu varijablu.

U metodi *calculateBMR()* se računa BMR, a *BMR – basal metabolic rate* je najmanja potrošnja energije koju osoba troši u stanju mirovanja, a koja se koristi za održavanje vitalnih tjelesnih procesa organizma.[17] (Primjer 9.) BMR uzima u obzir godine, tjelesnu visinu i tjelesnu težinu. Kada u registraciji označimo spol, u metodi *calculateBMR()* izvršava se jedna od ovih formula:

„Za muškarce:

BMR = 66 + (13.7 x tjelesna masa u kg) + (5 x tjelesna visina u cm) - (6.8 x godine)

Za žene:

BMR = 655 + (9.6 x tjelesna masa u kg) + (1.8 x tjelesna visina u cm) - (4.7 x godine)"[18]

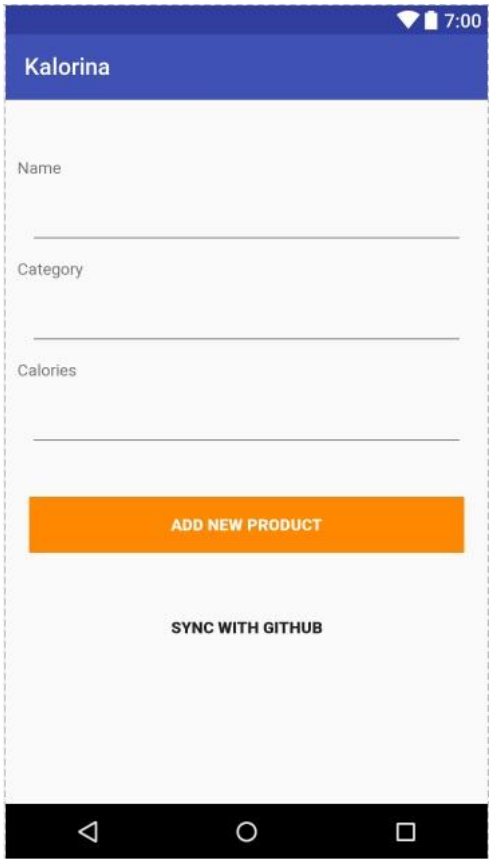
```
public Double calculateBMR(Integer Height, Integer Weight, Integer Age,
String Sex){
if (Sex == "F"){
return this.round((655 + (9.6 * Weight) + (1.8 * Height) - (4.7 * Age)),
1);
        }
else {
return this.round((66 + (13.7 * Weight) + (5 * Height) - (6.8 * Age)), 1);
        }
    }

public static double round(double value, int places) {
if (places <0) throw new IllegalArgumentException();

        BigDecimal bd = new BigDecimal(value);
        bd = bd.setScale(places, RoundingMode.HALF_UP);
return bd.doubleValue();
    }

public Double getBMRtmp() {
return bmrtmp;
    }
public void setBMRtmp(Double bmrtmp) {
this.bmrtmp = bmrtmp;
    }
```

Primjer 7. Prikaz metode *calculateBMR()*

[17] http://nadijeti.com/2013/02/11/bazalni-metabolizam-bmr/ (24.07.2017)
[18] https://www.fitness.com.hr/prehrana/nutricionizam/Kalorije.aspx (24.07.2017)

5.4. Dodavanje novih proizvoda

U klasi *Vegatable_pane* korisnik dodaje nove proizvode u bazu podataka koja se nalazi na GitHubu. GitHub je ogromna i visoko cijenjena platforma za kolaboraciju koja nam omogućuje korištenje svih funkcionalnosti Gita u kombinaciji sa setom drugih mogućnosti specifičnih za GitHub.[19] To je uvjedno i mjesto gdje se svoj kod može sa svima podijeliti.



Slika 12 Prikazuje izgled *activity_vegatable_panel.xml.*

Korisnik je pojeo svoju omiljenu čokoladu i želi je dodati u popis namirnica. Klikom na gumb „Add new product“ u *activity_user_page.xml* dolazimo do *activity_vegatable_panel.xml* gdje upisujemo tražene podatke, zatim kliknemo na “Add new product”. Prilikom klika pokrećemo metodu *setOnClickListener()* koja služi upravo za dodavanje vlastitih namirnica u bazu podatakam. Procedura je kao i kod ostalih Listenera koje smo spominjali slična. Nakon toga nas *Toast* obavijesti o uspješnom dodavanju nove namirnice.

[19] http://www.linuxzasve.com/uvod-u-github (24.07.2017)

Možemo protumačiti da je aplikacija "Kalorina" jedan proces, a dohvat podataka s Interneta zahtjeva jedan odvojeni drugi proces gdje u pozadini odrađuje svoje stvari. Za sinkronizaciju s GitHubom korisnimo klasu *AsyncTask* koja mora biti podklasa, a u ovom slučaju je podklasa klase *GetWeatherTask.* Podklasa će nadjačati barem jednu metodu (*doItBackground()*), a najčešće će nadjačati drugu metodu (*onPostExecute()*).[20]

U metodi *doItBackground()* imamo *Stream. Stream* je tok podataka koji se ostvaruje između nekog procesa na Internet mreži i aplikacije „Kalorina". *While* petljom se iterira redak po redak dohvaćenog skupa podataka, u ovom slučaju je to data file. Zatim se podatci iz tipa *StringBuilder* pretvara u obični *String*, a nakon toga se gasi konekcija. Ulazni parametar u metodi *onPostExecute()* je točno traženi dokument.

Kada smo uspješno dodali novu namirnicu tj.proizvod moramo sinkronizirati podatke s GitHubom. Klikom na gumb "Sync with Github" pozivamo metodu *syncListener()* u kojoj se nalazi link od baze podataka koja se nalazi na GitHubu. U nastavku malo više o tome.

5.5. SQLite baza podataka

„Android pruža elegantno sučelje preko kojeg aplikacija može raditi sa SQLite bazom podataka. Kako bismo pristupili bazi podataka potrebna nam je klasa koja pruža „vezu" s bazom podataka ako već ne postoji, a ta klasa se zove SQLite Open Helper (Gargenta, 2011,str.120).

U klasi DbHalndler imamo podklasu SQLite Open Helper. Na početku klase imamo definirane varijable tipa string kako bi kasnije kroz pisanje potrebnih SQL-ova bilo sve preglednije. Generalno imamo jednu bazu koja se zove *kalorina_db* i dvije tablice u njoj („user" i „vege"). Jedna za korisnika, a druga za pohranu namirnica koje se povlače s GitHuba. *Db_Version=1* je verzija baze podataka. Broj verzije je važan kako bismo kasnije, kada se shema promijeni, postojećim korisnicima mogli ponuditi način da ažuriraju bazu podataka na najnoviju shemu (Gargenta, 2011,str.124).

Ključna riječ *super* omogućava da izravno pozovemo metodu ili konstruktor iz roditeljske klase, u ovom slučaju DbHandler (Fain,2011,str.34).

[20] https://developer.android.com/reference/android/os/AsyncTask.html#doInBackground(Params...) (24.07.2017)

Na početku je potrebno kreirati željene tablice i to metodom *OnCreate( )*u DBHelperu (Primjer 10.). ID se postavlja kao primarni ključ jer svaka tablica treba imati definirati primarni ključ. Primarni ključ je varijabla koja će biti jedinstvena za svaki zapis. BMR je postavljen da bude DOUBLE jer je decimalni broj u pitanju, a ostale vrijednosti se unose kao cijeli broj, npr. za tjelesnu težinu upisujemo 50, a ne 50,5. Operaciju *db.execSQL*() koristimo za izvršavanje koda Create_Table.

```
publicvoidonCreate(SQLiteDatabasedb) {

StringCreate_Table = "CREATETABLE " + User_table_name+ "(" + User_id+ "
INTEGERPRIMARYKEY,"
+ User_name+ " TEXT,"
+ User_password+ " TEXT,"
+ User_height+ " INTEGER,"
+ User_weight+ " INTEGER,"
+ User_age+ " INTEGER,"
+ User_BMR+ " DOUBLE,"
+ User_BMR_tmp+ " DOUBLE,"
+ User_sex+ " TEXT" + ")";

db.execSQL(Create_Table);

StringCreate_Vege_Table = "CREATETABLE " + Vege_table_name+ "(" + Vege_id+
" INTEGERPRIMARYKEY,"
+ Vege_name+ " TEXT,"
+ Vege_calories+ " INTEGER,"
+ Vege_category+ " TEXT" + ")";

db.execSQL(Create_Vege_Table);
    }
```

Primjer 8. Prikaz metode *onCreate()*

Za korištenje SQl-a koristimo ne samo metodu *onCreate()* već i *onUpgrade()* (Primjer 11). Metoda *onUpgrade()* se poziva kad god je newVersion != oldVersion, drugim riječima, poziva se kad je verzija korisnikove baze podataka drugačija od verzije aplikacije.

```
@Override
publicvoidonUpgrade(SQLiteDatabasedb, intoldVersion, intnewVersion) {
db.execSQL("DROPTABLEIFEXISTS " + User_table_name);
onCreate(db);
    }

publicvoidaddUser(Userusr)
    {
SQLiteDatabasedb=this.getWritableDatabase();
ContentValuescv=newContentValues();

cv.put(User_name,usr.getName());
cv.put(User_password,usr.getPassword());
```

```
cv.put(User_height,usr.getHeight());
cv.put(User_weight,usr.getWeight());
cv.put(User_age,usr.getAge());
cv.put(User_BMR,usr.getBMR());
cv.put(User_BMR_tmp,usr.getBMRtmp());
cv.put(User_sex,usr.getSex());

db.insert(User_table_name, null, cv);

db.close();
    }
```

Primjer 9. Prikaz metode *OnUpgrade()*

Metoda *addUser()* je za dodavanje novog korisnika, a kao parametar prima objekt tipa User. Prvo dohvaća db instance za dodavanje novog korisnika. Dohvaća sve željene vrijednosti od korisnika koje smo dobili preko parametra metode. *Cv* je objekt čije vrijednosti zna čitati db instanca te se u njega prvo pohrane sve vrijednosti, zatim se pozove metoda *db.insert* kojoj šaljemo *cv* kao parametar i onda se sve spremi u bazu padataka. Na kraju je potrebno zatvoriti bazu podataka. Metoda *addVegetable()* funkcionira na način kao i *metoda addUser().* Razlika je u tome što dodaje nove proizvode, a kao parametar prima objekt tipa Vegetables.

Metoda *getCategories()* vraća listu stringova, a ne samo string, jer može biti više od jedne kategorije u bazi podataka. Metoda *getVegeByCategory()* dohvaća namirnice za neku željenu kategoriju čiji se naziv šalje putem parametra string category, a metoda *updateVege()* se poziva kada se unese željeni proizvod i njegove vrijednosti te se onda ti podatci samo dodaju na već postojeće podatke u bazu podataka.

6. Zaključak

Mobilna aplikacija nije nešto što se može dodirnuti, ona ne postoji u fizičkom svijetu – samo u virtualnom. No opet, toliko je utjecala na život ljudi. Sveukupno njeno virtualno djelovanje i postojanje direktno se implicira na stvaran, fizički svijet. Dovoljno je nekoliko klikova odnosno dodira ekrana mobitela a da posljedicu toga osjetimo u stvarnosti. Stoga, slobodno možemo reći da su aplikacije i Internet današnja stvarnost, a ne samo virtualni svijet. Kao primjer navodim moju aplikaciju. Trenutnu verziju karakterizira jednostavnost. Dizajnirana je na način da je mogu koristiti sve dobne skupine, bez poteškoća. Upravo ta jednostavnost funkcioniranja omogućuje neograničene nadogradnje u budućnosti; dodavanje fotografija hrane, grafikona koji pokazuju tijek unosa hrane, dijeljenje postignuća na društvene mreže, uspoređivanje s drugim korisnicima i drugo. Sve su to potencijalne nadogradnje koje ovise o korisničkim željama i potrebama tj. njihovoj povratnoj informaciji (feedback). Ograničavanjem dnevnog unosa kalorija u organizam na temelju izračuna koji su proizašli iz aplikacije, rezultirat će zdravijim stilom života. Upravo to pokazuje snagu i veličinu aplikacije i način na koji ona djeluje na čovjeka. Razvoj aplikacija i Interneta sadrži mnogo pozitivnih i negativnih strana. No, njihovim korištenjem u svrhu općeg napretka i blagostanja čovječanstva sa sigurnošću možemo reći da pozitivne strane prevladavaju. Jer ako pogledamo povijest, vidimo da svijet nikad nije brže napredovao nego danas, a životni standard nikad nije bio veći. Mobilne aplikacije dostupne su svima, u bilo koje vrijeme i na bilo kojem mjestu. One povezuju ljude, stvari i mjesta na način koji je do nedavno bio nezamisliv.

Popis slika

Literatura

Knjige:

1. Fain Y, 2011., *Programiranje Java*, Dobar Plan, Zagreb

2. Gargenta M, 2011., *Naučite Android*, Dobar Plan, Zagreb

Izvori s Interneta:

http://www.bug.hr/vijesti/korisnici-android-uredaja-cesce-deinstaliraju-apli/157070.aspx (8.9.2017.)
http://gs.statcounter.com/os-market-share#monthly-201703-201703-map (6.9.2017)
http://www.connect.de/ratgeber/android-geschichte-des-erfolgs-1491130.html (6.9.2017)
https://itsfoss.com/linux-better-than-windows/ (7.9.2017.)
https://source.android.com/devices/tech/dalvik/ (7.9.2017.)
https://thinkmobiles.com/blog/popular-types-of-apps/ (7.9.2017.)
http://www.hdonweb.com/mobiteli/nativna-aplikacija-mobilna-web-stranica (7.9.2017.)
http://www.evolva.hr/hr/razvoj_hibridnih_mobilnih_aplikacija.html (7.9.2017.)
https://developer.telerik.com/featured/what-is-a-hybrid-mobile-app/ (7.9.2017.)
https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/(8.9.2017.)
https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/(7.9.2017.)
https://venturebeat.com/2014/12/08/google-releases-android-studio-1-0-the-first-stable-version-of-its-ide/(7.9.2017.)
https://android-developers.googleblog.com/2013/05/android-studio-ide-built-for-android.html (20.07.2017.)
https://en.wikipedia.org/wiki/Application_programming_interface (6.9.2017.)
https://developer.android.com/guide/topics/ui/declaring-layout.html (21.07.2017)
https://web.math.pmf.unizg.hr/nastava/rp2/pred4/pred4.html (22.07.2017.)
http://nadijeti.com/2013/02/11/bazalni-metabolizam-bmr/ (24.07.2017)
https://www.fitness.com.hr/prehrana/nutricionizam/Kalorije.aspx (24.07.2017)
http://www.linuxzasve.com/uvod-u-github (24.07.2017)
https://developer.android.com/reference/android/os/AsyncTask.html#doInBackground(Params...) (24.07.2017)