RAZVOJ ANDROID APLIKACIJE ZA PRAĆENJE I KOMENTIRANJE TV SERIJA

Novak, Mirna

VISOKO UČILIŠTE ALGEBRA

ZAVRŠNI RAD

RAZVOJ ANDROID APLIKACIJE ZA PRAĆENJE I KOMENTIRANJE TV SERIJA

Mirna Novak

Zagreb, veljača 2018.

„Pod punom odgovornošću pismeno potvrđujem da je ovo moj autorski rad čiji niti jedan dio nije nastao kopiranjem ili plagiranjem tuđeg sadržaja. Prilikom izrade rada koristila sam tuđe materijale navedene u popisu literature, ali nisam kopirala niti jedan njihov dio, osim citata za koje sam navela autora i izvor, te ih jasno označila znakovima navodnika. U slučaju da se u bilo kojem trenutku dokaže suprotno, spremna sam snositi sve posljedice uključivo i poništenje javne isprave stečene dijelom i na temelju ovoga rada“.

U Zagrebu, 12.2.2018.

Mirna Novak

Predgovor

Srdačno se zahvaljujem se mentoru dipl. ing. Aleksanderu Radovanu koji mi je stručnim savjetima pomagao tijekom izrade završnog rada, te svojim vodstvom i potporom omogućio efikasno rješavanje svih nedoumica vezanih kako uz teorijski, tako i uz praktični dio rada.

Zahvaljujem svojim roditeljima za potporu i strpljenje iskazano tijekom cijelog školovanja.

Također se zahvaljujem svim kolegama i prijateljima koji su mi savjetima i preporukama pomogli u pisanju ovoga rada.

Prilikom uvezivanja rada, Umjesto ove stranice ne zaboravite umetnuti original potvrde o prihvaćanju teme završnog rada kojeg ste preuzeli u studentskoj referadi

Sažetak

U završnom radu izrađena je mobilna aplikacija za praćenje i komentiranje TV serija za Android mobilne uređaje. Koncept je razrađen u potpunosti, od početnog planiranja, pa sve do implementacije i testiranja. Izrađen je prototip aplikacije u Android Studio razvojnom okruženju sa svim pripadajućim funkcionalnostima. Aplikacija se sastoji od 5 ekrana, koji omogućuju korisnicima pristup raznim aktivnostima. Razgovor za određenu TV seriju pojavljuje se jedan dan prije početka emitiranja serije i čuva se u bazi sljedećih tjedan dana. Pripadajuća baza podataka Firebase sadrži podatke o korisnicima, uključujući i rating korisnika, te kontakte korisnika koji su pristali podijeliti kontakt. Nadalje, baza podataka sadrži razgovore razvrstane prema temi razgovora, odnosno seriji o čijoj se radnji razgovor vodi. Čuvaju se i pitanja s pripadajućim odgovorima priloženim u aktivnosti `Forum.java`. Datumu i vrijeme izlaska serija dohvaća se iz TMDb API baze podataka, koja je besplatna i otvorena za korištenje.

Pri registraciji korisnika u aplikaciju, sukladno njegovim odabirima, generira se lista serija. Korisnicima je omogućena međusobna komunikacija u vidu foruma i chat-ova, a zbog zanimljivije i sadržajnije komunikacije, korisnici se dijele u skupine po 5 osoba u svakom chat-u. Rad aplikacije je testiran na testnoj grupi korisnika.

Rad sadržava i opis komponenata koje nisu implementirane u ovom stadiju razvoja, već bi teoretski postojale kada bi slična aplikacija zadobila određeni broj korisnika. To su, primjerice, rješenja koja se tiču jako velikih baza podataka (engl. *Very Large Database*, skraćeno VLDBs), te razine pristupa korisnika u sklopu besplatnog (engl. *freemium*) poslovnog plana aplikacije, uz mogućnost nadogradnje aplikacije uz plaćanje.

Ključne riječi: Android, mobilna aplikacija, TV serije, baza podataka, forum, chat, testiranje aplikacije.

Sadržaj

1. Uvod

Aplikacije su u današnje doba sveprisutne, jednostavne za upotrebu, lako im se pristupa, te ih se svakim danom pojavljuje sve više na tržištu. Većini populacije teško je zamisliti moderan život bez mobilnih aplikacija, koje im omogućuju lak pristup prognozi vremena ili postavljaju automatsku budilicu. Također, puštaju muziku na temelju postavka korisnika, te korisnicima omogućuju pristup društvenim mrežama, gdje god se nalazili. Aplikacijama se nastoji pojednostaviti svakodnevni život te automatizirati svakodnevne aktivnosti.

Android je sustav otvorenog kôda, koji korisnike privlači upravo zbog otvorene i transparentne strukture i velike mogućnosti personalizacije. Također, daje programskim inženjerima potpunu slobodu prilikom pisanja kôda.

Rad je podijeljen u 5 cjelina. Prva cjelina, naziva „Android“, bavi se tematikom Android operacijskog sustava te mogućnostima grafičkih sučelja za razvoj Android aplikacija.

U drugoj cjelini, „Dizajn korisničkog sučelja“, razrađuju se razne mogućnosti dizajna Android aplikacija, uključujući i vodič za vizualni dizajn razvijen od strane Android inženjera.

Treća cjelina, „Funkcionalnosti aplikacije“, bavi se samom aplikacijom. Opisuje dohvaćanje podataka iz baze podataka, objašnjava pojedine aktivnosti aplikacije, te razrađuje kompleksne dijelove kôda, uz primjere.

Nadalje, četvrta cjelina, „Jako velike baze podataka (engl. *Very Large Databases*, skraćeno VLDBs)“, vezana je uz predmet funkcioniranja jako velikih baza podataka. Obrađene su pravilne mjere održavanja i administracije takvih baza, te neke od široko primijenjenih praksi i politika vezanih uz opisane baze. Također, navedeni su i primjeri načina pohrane podataka u tvrtkama koje broje milijune korisnika.

Peta cjelina, „Testiranje aplikacije“, bavi se načinima testiranja Android aplikacija, od odabira testne grupe korisnika, do testiranja intuitivnosti aplikacija kroz razne programe. Također, bavi se i načinima otkrivanja bugova i ranjivosti aplikacija.

2. Android

Android je naziv za mobilni operacijski sustav temeljen na Linux jezgri. U vlasništvu je Googlea, te je otvorenog kôda. Dostupan je u cijelosti pod Apache licencom, te je upravo zbog takve fleksibilnosti jedan od najraširenijih sustava.[1] Prvobitna verzija izašla je 2008. godine, dok je aktualna verzija, naziva „Oreo“, 15-ta po redu. Stariji Android mobilni telefoni koji su još uvijek u širokoj upotrebi ne podržavaju najnovije verzije, stoga su češće korištene starije verzije, koje podržava širi raspon mobilnih telefona. Neke od najkorištenijih verzija Android operacijskog sustava nazivaju se „Lollipop“, „Marshmallow“ i „Nougat“. Android operacijski sustav također podržava tablete i pametne televizore, te je uspješno implementiran i u Samsung kamere i frižidere. Aplikacija u praktičnom djelu ovog rada izrađena je u verziji Android 7.1.1, naziva „Nougat“, aplikacijskog programskog sučelja 25 (engl. *Application programming interface*, skraćeno API) za mobilne telefone. Trenutno je podržana na 1,5% Android mobilnih telefona.[2]

2.1. Značajke Androida

Android arhitektura prikazana je na slici (Slika 2.1.) i bazira se na Linux 2.6 jezgri koja se nalazi na dnu arhitekture i sastoji od drivera za međuprocesnu komunikaciju (engl. *Inter-process communication*, skraćeno IPC). (Božić et al., 2013:11)

Razlikuju se međuprocesne komunikacije između međusobno srodnih procesa i nepovezanih procesa. Najvažnije su lokalne domene za komunikaciju unutar lokalnog Linux sustava, te internet domene IPv4 i IPv6 za lokalnu i udaljenu komunikaciju preko interneta, odgovarajućim protokolima. Sloj za apstrakciju sklopovlja (engl. *Hardware Abstraction Layer*, skraćeno HAL) standardno je sučelje koje se sastoji od više modula, od kojih svaki implementira sučelje za određenu hardversku komponentu. Primjerice, modul za kameru i modul za Bluetooth. Od API-a 21, odnosno Android verzije 5.0, svaka aplikacija pokreće se kao zaseban proces s vlastitom instancom Android izvršnog okruženja (engl. *Android Runtime*, skraćeno ART). ART je zamjena za prethodan način pokretanja. naziva Dalvik,

[1] Lifewire. Dostupno na: https://www.lifewire.com/what-is-google-android-1616887 [18. veljače 2018.]
[2] Android Source. Dostupno na: https://source.android.com/compatibility/7.1/android-7.1-cdd.pdf [20. veljače 2018.]

koji je funkcionirao na način kompiliranja (engl. *compile*) kôda točno na vrijeme (engl. *Just-in-time*, skraćeno JIT). Koristi prijevremeni (engl. *Ahead-Of-Time*, skraćeno AOT) kompajler koji pridonosi bržem učitavanju stranica, zahvaljujući prijevodu *bytecode*-a u specifičan kôd prilikom preuzimanja aplikacije s Google Play-a. [3] Komponente Android sustava, uključujući i ART i HAL, stvorene su od izvornog kôda koji je napisan u C i C++ programskom jeziku. (Wang, 2011:340)

Iznad jezgre nalaze se biblioteke (eng. *libraries*). Neke od njih su:

- Surface Manager – za pravilno iscrtavanje korisničkog sučelja
- OpenGL – za manipuliranje 2D i 3D vektorskom grafikom
- SGL – za skalabilne grafičke biblioteke
- SSL – za sigurnu internet komunikaciju
- SQLite – za implementaciju baza podataka

Neke od osnovnih Android biblioteka za razvojne programere su:

- android.app – temelj Android aplikacija koji omogućuje pristup aplikaciji
- android.content – služi za komunikaciju između aplikacije i komponenti aplikacije
- android.os – za renderiranje i manipuliranje tekstom
- android.view – osnovne komponente za izradu grafičkog sučelja aplikacije
- android.widget – prethodno izrađene komponente kao što su gumbi, liste ili polje za unos teksta
- android.webkit – klase koje aplikaciji omogućuje kretanje po internetu[4]

Nakon biblioteka dolazi aplikacijski okvir koji omogućuje korisniku rad sa servisima višeg nivoa u obliku Java klasa. On uključuje iduće servise:

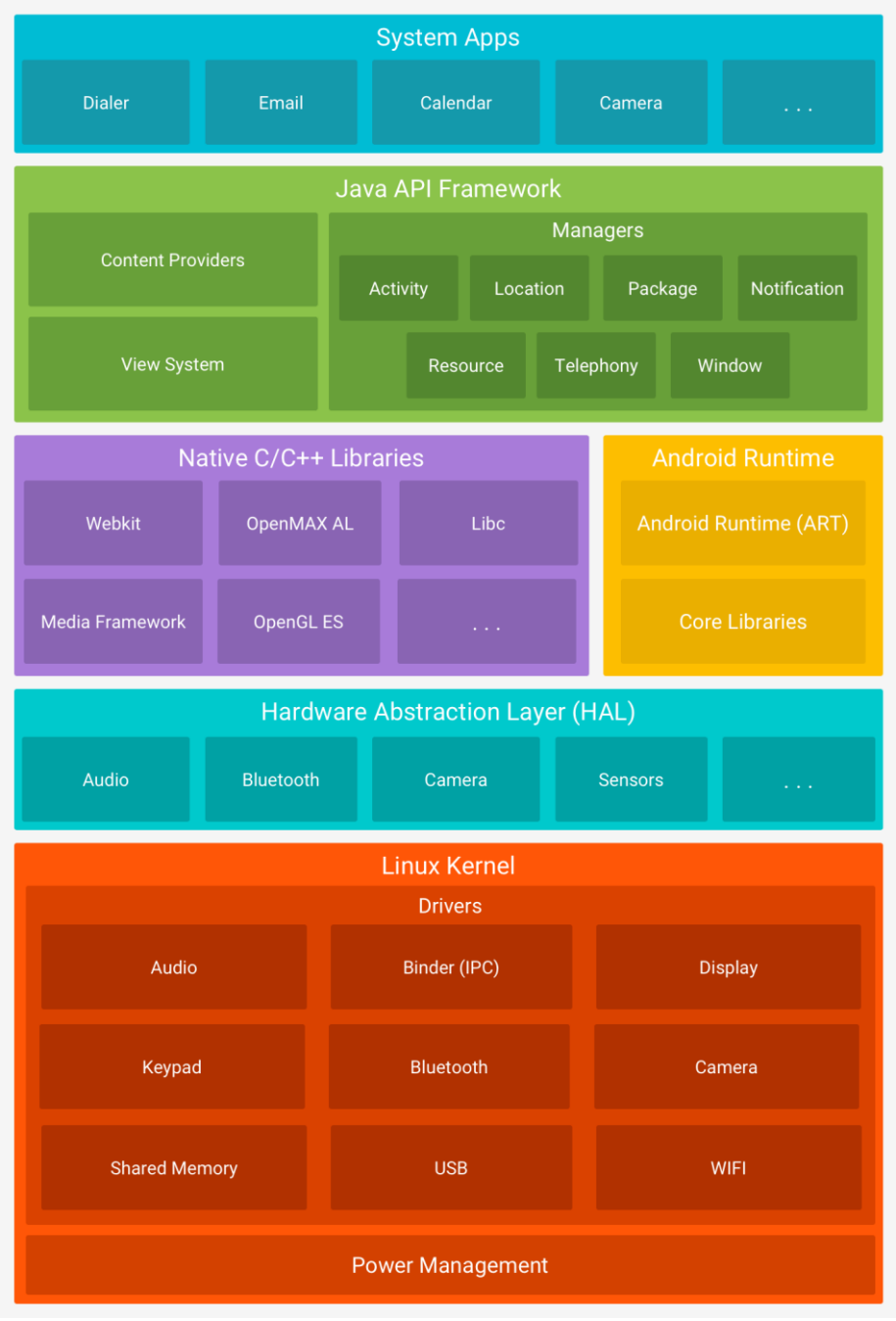
- Menadžer aktivnosti (engl. *Activity Manager*) – za kontrolu životnog ciklusa aplikacije
- Davatelj sadržaja (engl. *Content Provider*) – omogućuje dijeljenje podataka s ostalim aplikacijama
- Upravitelj resursima (engl. *Resource Manager*) – za pristup resursima, kao što su boje i izgled grafičkog sučelja

[3]ICT business. Dostupno na: http://www.ictbusiness.info/vijesti/novi-android-runtime-ce-zamijeniti-dosadasnji-dalvik [15. veljače 2018.]
[4] Tutorials point. Dostupno na: https://www.tutorialspoint.com/android/android_architecture.htm [15. veljače 2018.]

- Upravitelj obavijesti (engl. *Notifications Manager*) – za prikazivanje upozorenja i obavijesti
- Prikaz Sustava (engl. *View System*) – skup prikaza za stvaranje grafičkog korisničkog sučelja

Posljednji sloj Android arhitekture čine aplikacije. Neke od njih su birač brojeva, popis kontakata, internetski pretraživač, kalendar i kamera.[5]

[5] Android. Dostupno na: devhttps://developer.android.com/guide/platform/index.html [15. veljače 2018.]



Slika 2.1. Stog softvera Android operacijskog sustava

2.2. Android Studio i Eclipse

Android Studio i Eclipse dva su najpoznatija integrirana razvojna okruženja (engl. *Integrated Development Environment*, skraćeno IDE) za Android operacijski sustav. [6]

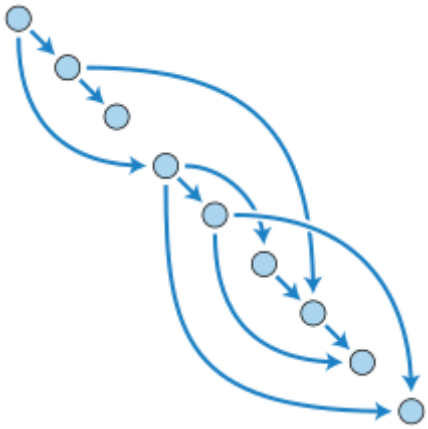
[6] Android. Dostupno na: https://developer.android.com/studio/intro/index.html [16. veljače 2018.]

Android Studio službeno je razvojno okruženje za Android aplikacije. Izrađen je na IntelliJ IDEA softveru razvijenom od strane JetBrains-a, 2013. godine. Nastao je kao alternativa za Eclipse-ove alate za razvoj Android aplikacija. [7]

Android Studio koristi Gradle sustav za izgradnju aplikacija (engl. *build*), pomoću Apache Ant i Apache Maven koncepata. Također koristi Groovy-based DSL domenski jezik za deklariranje konfiguracije projekta, odnosno skriptiranje *build*-a aplikacija. Android Studio dizajniran je za izgradnju velikog broja projekata, a to mu omogućuje korištenje usmjerenog acikličkog grafa (engl. *Directed acyclic graph*, skraćeno DAG). Usmjereni aciklički graf je graf u kojem se svi rubovi kreću prema naprijed, te ne postoje ciklusi, kao što je prikazano na slici (Slika 2.2.). Android Studio koristi module. Svaki modul može imati Grade datoteku, a moduli imaju mogućnost simultano funkcionirati.[8]

Eclipse razvojno okruženje pisano je u Javi. Ovo programsko okruženje nije razvijeno isključivo za izgradnju Android aplikacija, već se u njemu može razvijati i niz aplikacija u drugim programskim jezicima, kao što su Java, Ada, C, C++, PHP, Python, Ruby, COBOL i mnogi drugi. Upravo zbog toga, samo korištenje Eclipse razvojnog okruženja zahtjeva veliku količinu memorije s nasumičnim pristupom (engl. *Random Access Memory*, skraćeno RAM) i procesorsku snagu.[9] Projekti u radnom okruženju u Eclipse-u organiziraju se prilikom pokretanja, zbog čega automatsko korištenje projekata iz drugih radnih okruženja nije moguće.[10]

[7] Android. Dostupno na: https://android-developers.googleblog.com/2013/05/android-studio-ide-built-for-android.html [15. veljače 2018.]
[8] AndroidPub. Dostupno na: https://android.jlelse.eu/mastering-viewpager-with-directed-acyclic-graph-533ed5ee1e5e [15. veljače 2018.]
[9] Kinetic Growth. Dostupno na: http://www.kineticgrowth.com/make-eclipse-use-memory-ram-efficiently/ [15. veljače 2018.]
[10] Eclipse. Dostupno na: https://www.eclipse.org/ide/ [15. veljače 2018.]



Slika 2.2. Prikaz forme DAG acikličkog grafa korištenog u Android Studio-u

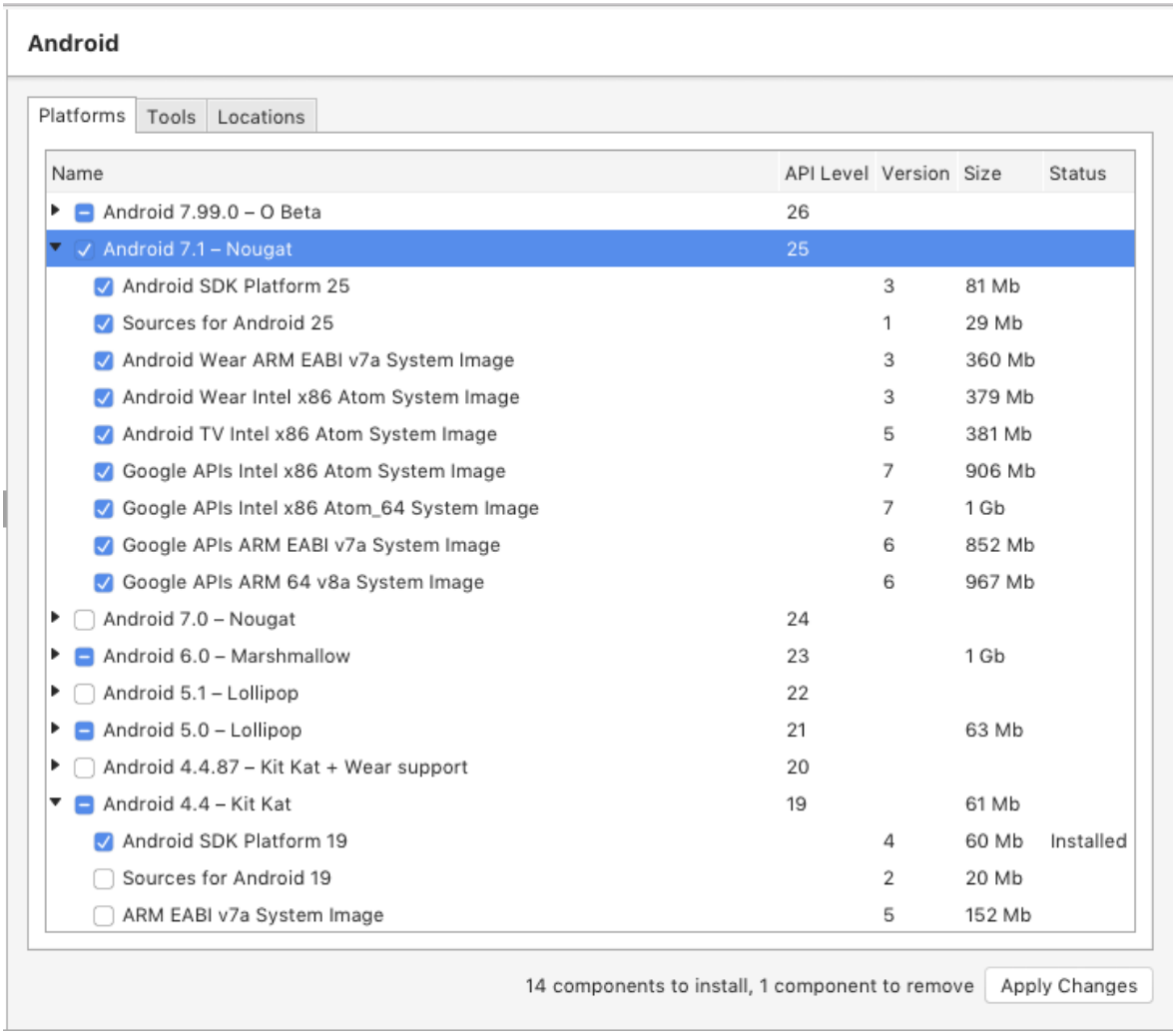
Oba razvojna okruženja koriste Java automatsko izvršavanje kôda. Aplikacija razvijena u sklopu ovog rada razvijena je u Android Studio razvojnom okruženju.

Kako bi se aplikacije mogle pravilno razvijati potrebno je instalirati i Android SDK razvojni paket.

2.2.1. Android SDK

Android SDK programski je razvojni paket koji sadrži podatke o aplikaciji, izvorni kôd, manifest datoteku, te metapodatke aplikacije. Podatke pohranjuje u datoteku koja završava s ekstenzijom „.apk“, generiranu na temelju značajki ZIP formata.[11] Mogućnost odabira verzije Androida, korištenjem SDK Manager-a, prikazano je na slici (Slika 2.3.). Preko Android SDK paketa moguće je preuzeti niz alata, kao što su Android emulator za realističnu simulaciju aplikacije na mobilnom ekranu, postavke Bluetootha ili postavke GPS-a. Također, sadrži i alate za uklanjanje grešaka u kôdu, primjere kôda i mnoge druge.[12]

[11] Macworld. Dostupno na: https://www.macworld.com/article/1061005/androidsdk.html [16. veljače 2018.]
[12] Android. Dostupno na: https://developer.android.com/studio/intro/index.html [16. veljače 2018.]



Slika 2.3. Primjer izgleda SDK Manager-a

2.3. Dizajn korisničkog sučelja

Material Design, također poznat pod nazivom Quantum Paper, naziv je za sveobuhvatan vodič za vizualni dizajn i interaktivnost Android aplikacija. Uveden je 2014. godine. Primjenjuju ga Gmail, YouTube, te sve aplikacije Google Play-a. Material Design razvojnim programerima omogućuje korištenje alata za stvaranje novih tema, mijenjanje boje gotovih softverskih objekata za izradu grafičkog korisničkog sučelja (engl. *widget*)[13], te prilagođenih animacija.[14] Primjer dizajna Android aplikacije stvoren primjenom Material Design-a prikazan je na slici (Slika 2.4.). Također, Material Design nudi mogućnost personalizacije

[13] Prizvuk. Dostupno na: http://www.prizvuk.hr/old/hr_widgets.html [16. veljače 2018.]
[14] Android Authority. Dostupno na: https://www.androidauthority.com/best-material-design-apps-for-android-523420/ [16. veljače 2018.]

dijaloga upozorenja, opcija pojedinačnog ili višestrukog odabira, opcija za odabir vremena i datuma, dijaloških okvira, te načina prikaza dijaloga preko cijelog zaslona mobilnih uređaja.[15]

Slika 2.4. Primjer primjene Material Design dizajna na Android mobilnom telefonu

Uz ugrađenu Android opciju za dizajn mobilnih aplikacija, moguće je pronaći i niz gotovih rješenja za dizajn aplikacija, besplatnih ili uz jednokratno plaćanje. Ponuda stilova i tema velika je, te najčešće u obliku gotovih predložaka dizajna. Predlošci nude velik izbor jednostavnih i modernih tema, s uputama za laku implementaciju. Naglasak se stavlja na intuitivnost dizajna korisničkog sučelja, kako bi aplikacija krajnjim korisnicima bila što jasnija i jednostavnija za korištenje.[16]

2.4. Linux-libre operacijski sustav

Android arhitektura bazira se na Linux operacijskom sustavu. Linux operacijski sustav otvorenog je tipa, što znači da je njegova arhitektura i kôd dostupan svima.[17] Zahvaljujuću tome, razvojni programeri imaju mogućnost potpuno razumjeti sve dijelove kôda, te sukladno tome pronaći način na koji mogu modificirati vlastiti kôd, uz cilj postizanja

[15] Android. Dostupno na: https://developer.android.com/design/material/index.html [16. veljače 2018.]
[16] Material Design. Dostupno na: https://material.io/guidelines/ [16. veljače 2018.]
[17] LWN. Dostupno na: https://lwn.net/Articles/376566/ [18. veljače 2018.]

optimalnih performansi nekog programa ili aplikacije u Linux operacijskom sustavu. Linux privlači sve više korisnika upravo zbog slobode koju im omogućuje. Ovaj operacijski sustav, međutim, još uvijek sadržava dijelove kôda koji su u binarnom obliku, te su dostupne, ali nisu čitljive korisnicima. Također, jedan dio jezgre Linux sustava licenciran je, odnosno nije u potpunosti dostupan.

Linux-libre naziv je inicijative koju vodi organizacija za besplatne softvere u Južnoj Americi (engl. *Free Software Foundation Latin America*, skraćeno FSFLA). Cilj joj je učiniti Linux u potpunosti čitljivim, slobodnim i besplatnim. [18]

[18] FSF Linux-libre. Dostupno na: https://www.fsfla.org/ikiwiki/selibre/linux-libre/ [18. veljače 2018.]

3. Funkcionalnosti aplikacije

Aplikacija razvijena u sklopu ovog rada bavi se tematikom komentiranja TV serija, s naglaskom na mogućnost sudjelovanja u razgovorima. Razgovori vezani od pojedinu TV seriju otvaraju se jedan dan prije početka emitiranja serije, te se čuva u bazi tjedan dana.

Nakon što se korisnik registrira ili prijavi u aplikaciju s postojećim korisničkim podatcima, na temelju podataka o korisniku koji se čuvaju u Firebase bazi podataka, lista serija koje korisnik prati bit će generirana na početnom ekranu. Prilikom odabira izbornika, pojavljuje se mogućnost odabira jednog od pet ekrana. Redom, to su ekran s listom TV serija koje korisnik prati, ekran s prikazom serija, s opcijom pristupa chat-u vezanom uz pojedinu seriju. Nadalje, ekran s listom pitanja postavljenih od strane korisnika, sortiranim prema nazivu serije, ekran s popisom osoba koji su podijelili svoje osobne podatke s trenutnim korisnikom, te ekran s prikazom profila korisnika.

3.1. Volley

Android Volley API mrežna je biblioteka koja pomaže automatizaciji upravljanja mrežnim zahtjevima. Volley biblioteku karakteriziraju iduće značajke i mogućnosti:

- Automatiziran raspored mrežnih zahtjeva
- Mogućnost više simultanih mrežnih veza
- Volley predmemorijska (engl. *cache*) HTTP biblioteka
- Prioritizacija zahtjeva (engl. *request*)
- Otkazivanje već pokrenutih API zahtjeva[19]

Kako bi Volley bilo moguće koristiti u Android aplikaciji, potrebno je klonirati Volley repozitorij. U ovdje priloženoj aplikaciji koristi se repozitorij preuzet s GitHub razvojne platforme.[20] Nakon toga, u projekt se dodaje kloniran repozitorij. U „settings.gradle“ i

[19] SitePoint. Dostupno na: https://www.sitepoint.com/volley-a-networking-library-for-android/ [18. veljače 2018.]
[20] GitHub. Dostupno na: https://github.com/google/volley [18. veljače 2018.]

„build.gradle“ datoteke, uključuje se prethodno dodan repozitorij kako bi se povezao s projektom.[21]

Kôd u „build.gradle“ datoteci, sa podcrtanim dijelom u kojem se dodaje daje naredba za kompiliranje, odnosno povezivanje Volley repozitorija, prilikom pokretanja aplikacije:

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    androidTestCompile('com.android.support.test.espresso:espresso-
    core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-
        annotations'
    })
    compile 'com.android.support:appcompat-v7:25.3.1'
    compile 'com.android.support.constraint:constraint-layout:1.0.2'
    testCompile 'junit:junit:4.12'
    compile 'com.firebase:firebase-client-android:2.5.2+'
    testCompile 'junit:junit:4.12'
    compile project (':volley')
    compile 'com.squareup.picasso:picasso:2.5.2'
    compile 'com.android.support:recyclerview-v7:25.3.1'
    compile 'com.android.support:design:25.3.1'
    compile 'com.android.support:cardview-v7:25.3.1'
}
```

Kôd 1. Dio kôda iz „build.gradle“ datoteke pomoću koje je aplikacija povezana s Volley bibliotekom

Kôd u „settings.gradle“ datoteci:

```
include ':app', ':volley'
```

Nakon što se Volley biblioteka poveže s projektom, u klasi `ForumRoomActivity.java` definiramo varijablu *request*. Pomoću nje dohvaćaju se podatci s unaprijed definirane lokacije sadržaja (engl. *Uniform Resource Locator*, skraćeno URL). Dobivaju se podatci u formatu JavaScript objekta (engl. *JavaScript Object Notation*, skraćeno JSON). Ti podatci zatim se šalju u metodu *doOnSuccess (String s)*, gdje se iščitavaju. Ovaj proces događa se asinkrono, tako što se pomoću *Volley.newRequestQueue* naredbe, zahvaljujući Volley biblioteci, navedeni zahtjev (engl. *request*) automatski dodaje u procesni kanal (engl. *pipeline*). Kada se proces dovrši, Volley signalizira slušatelju odgovora (engl. *listener*) da započne *onResponse (String s)* metodu. Primjer korištenja Volley biblioteke u aplikaciji, za podnošenje GET zahtjeva u `ForumRoomActivity.java` klasi:

[21] Rich ard Rose Blog. Dostupno na: https://richardroseblog.wordpress.com/2016/05/06/jolly-volley-android-networking-library/ [18. veljače 2018.]

```
StringRequest request = new StringRequest(Request.Method.GET, url, new
Response.Listener<String>() {
    @Override
    public void onResponse(String s
    ) {
        doOnSuccess(s);
    }
}, new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError volleyError) {
        System.out.println("" + volleyError);
    }
});

RequestQueue rQueue = Volley.newRequestQueue(ForumRoomActivity.this);
rQueue.add(request);
```

Kôd 2. Primjer korištenja Volley-a u `ForumRoomActivity.java` datoteci za GET zahtjev (engl. *request*)

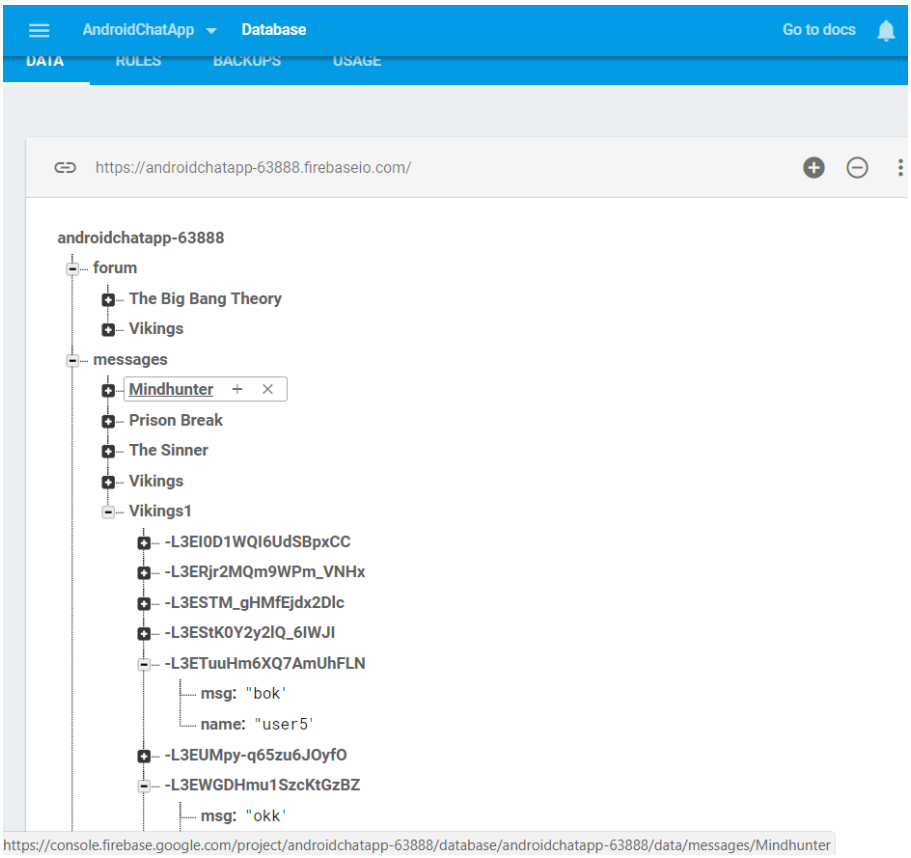
3.2. Firebase

Firebase je naziv platforme za razvoj aplikacija otvorenog kôda. Nudi velik broj servisa kao što su:

- Firebase Analytics – pruža uvid u iskustvo korisnika prilikom korištenja aplikacije
- Firebase Cloud Messaging – koristi se kao rješenje za slanje poruka i notifikacija preko različitih platformi (uključuje Android, iOS i web aplikacije)
- Firebase Remote Config – daje razvojnim programerima mogućnost promjene aplikacije bez nužnog zahtijevanja ažuriranje aplikacije od strane korisnika [22]

RealTime Database također je jedan od Firebase servisa. Izgled Firebase sučelja aplikacije razvijene u sklopu ovog rada prikazan je na slici (Slika 3.1.).

[22] Firebase. Dostupno na: https://firebase.google.com/docs/ [18. veljače 2018.]



Slika 3.1. Izgled podataka u Firebase bazi podataka povezanoj s aplikacijom

Firebase baza podataka može se dodati u aplikaciju direktno iz Android Studio korisničkog sučelja, a povezuje se dodavanjem zavisnosti (engl. *dependency*) u „build.gradle“ datoteku.[23]

U `Register.java` aktivnosti, prilikom klika na gumb *registerButton* pri registraciji novog korisnika, prvo se dohvaća tekst iz polja za unos teksta. Polja imaju naziv *username* i *password,* a sadržaj polja unosi se od strane korisnika. Također, u varijablu tekstualne liste (engl. *ArrayList <String>*) naziva *checkedIt* spremaju se TV serije koje je korisnik označio prilikom registracije. Navodi se putanja do željenog mjesta pohrane podataka u Firebase bazi podataka, te se pomoću referenci određuje naziv djeteta (engl. *child*). U ovom slučaju za naziv se koristi korisničko ime korisnika. Prilikom spremanja više podataka koji imaju jednog roditelja (engl. *parent*) u bazu, koristi se mapa (engl. *HashMap*), u koju se prethodno kao lista podataka pohranjuju nazivi odabranih TV serija. Nakon toga, cijela se mapa upisuje kao lista u roditelja sa proizvoljnim nazivom. Nadalje, provjerava se postoji li već korisnik s istim korisničkim imenom. Ako ne postoji, korisnik se upisuje u Firebase bazu podataka. Naziv roditelja svih korisnika je *users*. Zatim dolaze imena svih korisnika, a kao *child*

[23] Firebase. Dostupno na: https://firebase.google.com/docs/database/android/start/ [18. veljače 2018.]

svakome od njih pohranjeni su lista serija i lozinka, pod nazivima *myItems* i *password*. Primjer upisivanja podataka u Firebase bazu podataka u `Register.java` aktivnosti:

```
registerButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        user = username.getText().toString();
        pass = password.getText().toString();
        checkedIt=adapter.getCheckedItems();
String url = "https://androidchatapp-63888.firebaseio.com/users.json";
        StringRequest request = new StringRequest(Request.Method.GET,
        url, new Response.Listener<String>(){
                @Override
                public void onResponse(String s) {
                    Firebase reference = new
                    Firebase("https://androidchatapp-
                    63888.firebaseio.com/users");
                       if(s.equals("null")) {
      reference.child(user).child("rating").setValue(UserDetails.rating);
      reference.child(user).child("password").setValue(pass);
      HashMap<String, String> names =new HashMap<String, String>();
          for (String ch:checkedIt) {
             names.put(ch, ch);
           }
       reference.child(user).child("myItems").setValue(names);
       Toast.makeText(Register.this, "registration successful",
       Toast.LENGTH_LONG).show();
       startActivity(new Intent(Register.this, MovieListActivity.class));
                       }
                       else {
                          try {
                    JSONObject obj = new JSONObject(s);
                    if (!obj.has(user)) {
      reference.child(user).child("rating").setValue(UserDetails.rating);
      reference.child(user).child("password").setValue(pass);
      HashMap<String, String> names =new HashMap<String, String>();
          for (String ch:checkedIt) {
             names.put(ch, ch);
          }
      reference.child(user).child("myItems").setValue(names);

      Toast.makeText(Register.this, "registration successful",
      Toast.LENGTH_LONG).show();
                    }
                    else {
      Toast.makeText(Register.this, "username already exists",
      Toast.LENGTH_LONG).show();
                     }

          } catch (JSONException e) {
              e.printStackTrace();
           }
      }
   },new Response.ErrorListener(){
       @Override
       public void onErrorResponse(VolleyError volleyError) {
            System.out.println("" + volleyError );
            pd.dismiss();
        }
  });
```

```
  RequestQueue rQueue = Volley.newRequestQueue(Register.this);
  rQueue.add(request);
}
```

Kôd 3. Upisivanje podataka u Firebase bazu podataka u aktivnosti `Register.java`

Prilikom dohvaćanja podataka iz Firebase baze podataka, prvo se definira varijabla naziva *url*, odnosno putanja do mjesta u kojem su podatci spremljeni. U ovom primjeru iz `MovieListActivity.java` aktivnosti, dohvaćaju se nazivi svih TV serija koje je trenutni korisnik odabrao prilikom registracije. Nakon što, pomoću *Volley.newRequestQueue* naredbe, Volley da signal za početak *onResponse (String s)* metode, započinje metoda *doOnSuccess (String s)*, gdje je varijabla *s* dobiveni dogovor. U metodi *doOnSuccess (String s)*, proslijeđeni JSON objekt *s* sadrži podatke od sve djece čiji je naziv roditelja *myItems* definiran u varijabli *url*. Pomoću iteratora *i*, dobivaju se imena svakog djeteta. Dokle god postoji idući naziv, *while (i.hasNext()){}* petljom dobiva se naziv svake od serija, te ih se sprema se u varijablu *chosenShows*.

Primjer dohvaćanja liste odabranih serija trenutnog korisnika iz Firebase baze podataka:

```
String url = "https://androidchatapp-63888.firebaseio.com/users/" +
UserDetails.username + "/myItems.json";

StringRequest request = new StringRequest(Request.Method.GET, url, new
Response.Listener<String>(){
    @Override
    public void onResponse(String s
    ) {
        doOnSuccess(s);
    }
    },new Response.ErrorListener(){
    @Override
    public void onErrorResponse(VolleyError volleyError) {
        System.out.println("" + volleyError);
    }
});
RequestQueue rQueue = Volley.newRequestQueue(MovieListActivity.this);
rQueue.add(request);

public void doOnSuccess(String s) {
    try {
        JSONObject obj = new JSONObject(s);
        Iterator i = obj.keys();
        String key = "";

        while (i.hasNext()) {
            key = i.next().toString();
            chosenShows.add(key);
        }
    } catch (JSONException e) {
        e.printStackTrace();
    }
}
```

Kôd 4. Dohvaćanje podataka iz Firebase baze podataka u aktivnosti `MovieListActivity.java`

3.3. TMDb API

Nazive serija, kratki sadržaj, te vrijeme emitiranja TV serija, aplikacija dohvaća iz baze podataka filmova (engl. *The Movie Database*, skraćeno TMDb).

Sučelje za programiranje aplikacija (engl. *Application Programming Interface*, skraćeno API) skup je funkcija i protokola koje programeri koriste za stvaranja aplikacija koje pristupaju podatcima operacijskog sustava, aplikacije ili druge usluge.[24] Uz uvjet registracije, TMDd dodjeljuje API ključ, te su pomoću njega programeri u mogućnosti pristupiti TMDb bazi podataka i implementirati ju u svoje projekte, te koristiti dobivene podatke.[25]

Nakon dodjeljivanja API ključa, potrebno je isti dodati u „gradle.properties“ datoteku. Izgled API ključa korištenog u priloženoj aplikaciji:

TMDB_API_KEY="9f989f7fd0f7316e89f32263adeadabe"

Također je potrebno dozvoliti aplikaciji pristup internetu:

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

Kôd 5. Dio kôda koji dozvoljava pristup aplikacije internetu u „AndroidManifest.xml“ datoteci

Potrebni podatci o TV serijama dohvaćaju se iz TMDb baze podataka. Nakon što se proslijedi prikladna putanja, u ovom slučaju, putanja do liste svih TV serija pohranjenih u bazi, nasumično se sprema prvih 40 stranica dobivenih odgovora. Ovo je lista koja se prikazuje prilikom registracije korisnika, a zbog veće brzine aplikacije prikazuje se samo prvih 40 rezultata. Ostali rezultati također su vidljivi, ali tek kada se koristi tražilica integrirana u `Register.java` klasi. Dobiveni rezultati pohranjuju se u prethodno definirane varijable. Slika koja prikazuje seriju također je među tim podatcima. Na početku, definira se varijabla *MOVIE_IMAGE_URI* koja je početak *url* adrese slika svih serija. Da bi se dobila puna putanja, na nju se zatim dodaje podatak o veličini ekrana mobilnog telefona *mScreenDensity* korisnika koji trenutno koristi aplikaciju kako bi slika bila prikladne

[24]IGI Global. Dostupno na: https://www.igi-global.com/dictionary/api/1298 [18. veljače 2018.]
[25]TheMovieDatabase. Dostupno na: https://www.themoviedb.org/documentation/api [18. veljače 2018.]

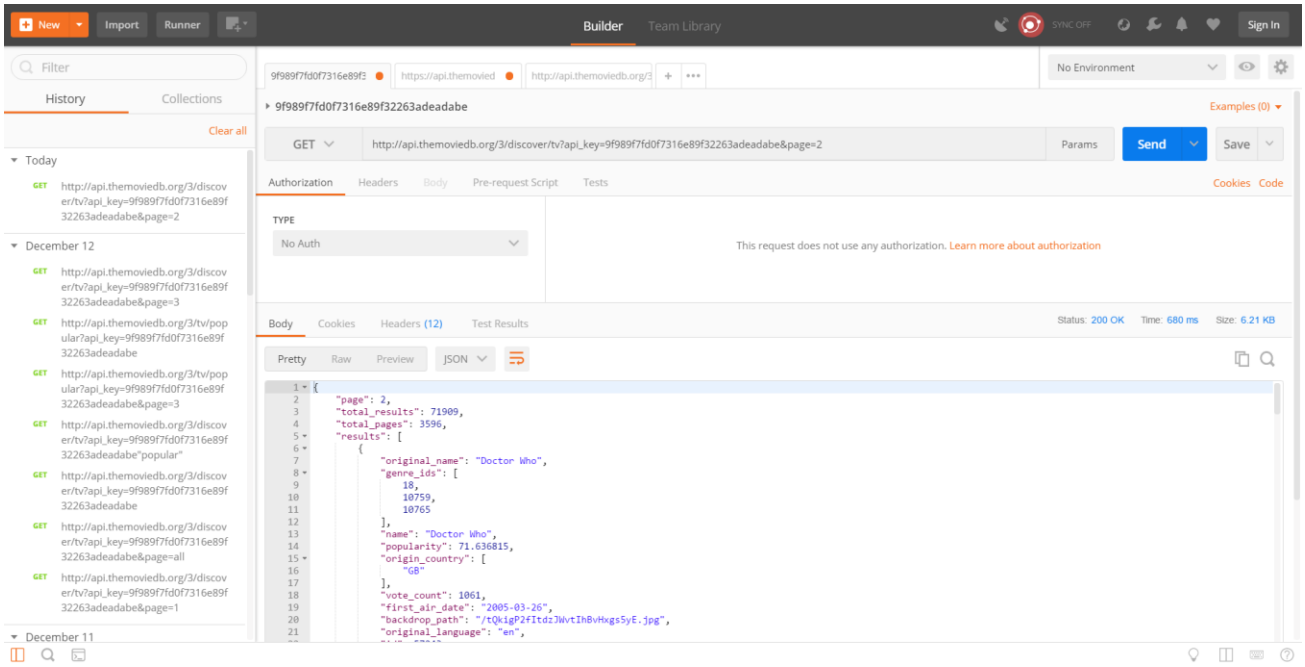
veličine. Nadalje, dodaje se posljednji dio *url* putanje koja vodi do slike određene serije dohvaćen iz TMDb-a, pohranjen u varijablu *poster_uri*.

Primjer dohvaćanja sadržaja iz TMDb baze podataka:

```
private void onRequestMovieAPI() {
    final String MOVIE_API_URI =
"http://api.themoviedb.org/3/discover/tv" ;

    final String MOVIE_API_KEY = "?api_key=" + API_KEY;
    final String MOVIE_IMAGE_URI = "http://image.tmdb.org/t/p/";

    Integer i=1;
    do {
    final JsonObjectRequest mJsonObjectRequest = new JsonObjectRequest
       (Request.Method.GET, MOVIE_API_URI+MOVIE_API_KEY+ "&page="+i,
        null, new Response.Listener<JSONObject>() {
          @Override
          public void onResponse(JSONObject response) {
            try {
              JSONArray jsonArray = response.getJSONArray("results");
              String id;
              String title;
              double  rating;
              String poster_uri;

              for (int i=0; i < jsonArray.length(); i++) {
                   JSONObject movie = jsonArray.getJSONObject(i);
                   id = movie.getString("id");
                   title = movie.getString("name");
                   rating = movie.getDouble("vote_average");
                   poster_uri = movie.getString("poster_path");
                   for(int j=0; j< chosenShows.size(); j++){
                      if(chosenShows.get(j).equals(title)){
                          mMediaInformation.add(new Media(id,
                          MOVIE_IMAGE_URI+mScreenDensity+poster_uri,
                          title, rating));
                      }
                    }
                }
                 mMovieAdapter.notifyDataSetChanged();
              } catch (JSONException ex) {
                  ex.printStackTrace();
              }
      }
  }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            error.printStackTrace();
            Log.i("JSON", error.getMessage());
        }
  });

 Volley.newRequestQueue(getApplicationContext()).add(mJsonObjectRequest);
 i++;
 }while(i<40);
}
```

Kôd 6. Dohvaćanje podataka iz TMDb baze podataka

Postman je skupina alata jednostavnih za korištenje, koji omogućuje brzo dohvaćanje i prikaz podataka. Većinom ga koriste razvojni programeri API-a, međutim koristan je zbog jasnog grafičkog pregleda strukture podataka određenog API ključa.



Slika 3.2. Primjer prikaza podataka u Postman-u, s *url* putanjom do serija u TMDb bazi podataka korištenoj prilikom izrade aplikacije

3.4. Aktivnosti

Aplikacija se sastoji od 5 glavnih ekrana kojima se pristupa iz glavnog menija dostupnog u svakoj aktivnosti, te od dodatnih sadržaja ekrana kojima se pristupa iz pojedinih aktivnosti.

3.4.1. Aktivnost s popisom TV serija

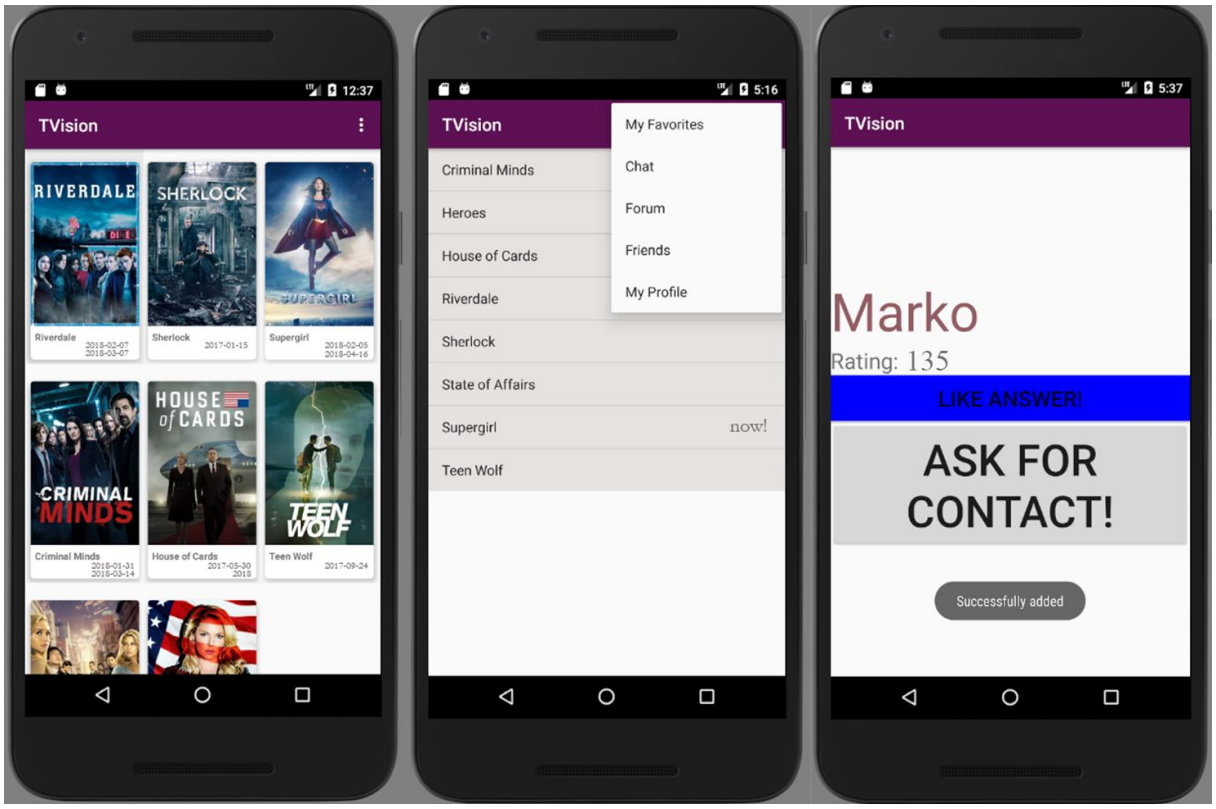
Aktivnost s listom TV serija prva je aktivnost koje se pojavljuje na ekranu nakon prijava korisnika u aplikaciju. Ovdje se aplikacija povezuje s TMDb bazom podataka koja sadržava podatke o filmovima, te pomoću API-a dohvaća listu TV serija. Filtriranjem se prikazuju podatci o onim serijama koje je korisnik odabrao prilikom registracije u aplikaciju. Podatci o korisniku i o serijama koje je korisnik odabrao čuvaju se u Android Firebase bazi podataka. Prikazuju se slike TV serija pomoću linkova dohvaćenim iz baze podataka, naziv serije, te vrijeme emitiranja zadnje i prve iduće epizode TV serije. Klikom na bilo koju od njih, otvara se nova aktivnost u kojoj je naveden kratki sadržaj, ocjena serije temeljena na mišljenju gledatelja, te osnovni podatci o odabranoj seriji.

3.4.2. Chat aktivnost

Prilikom klika na meni, otvara se mogućnost odabira aktivnosti naziva `ChatSeries.java`, koja prikazuje listu naziva TV serija koje korisnik prati, a emitiraju se u okviru vremena od jednog dan prije izlaska nove epizode, do tjedan dana nakon emitiranje te epizode. To je okvir vremena u kojem se razgovor vezan uz tu epizodu TV serije čuva u Firebase bazi podataka. Nakon što korisnik klikne na ime serije čijem chat-u želi pristupiti otvara se aktivnost `ChatRoomActivity.java`, gdje se prikazuje dotadašnji razgovor o temi. Tekst razgovora i korisnici koji sudjeluju u njemu čuvaju se u Firebase bazi podataka.

3.4.3. Aktivnost s forumom

U aktivnosti `Forum.java`, prikazana je lista serija koje korisnik prati. Nakon klika na željenu seriju otvara se aktivnost `ForumRoomQuestions.java` gdje se nalazi lista već postavljenih pitanja o odabranoj seriji od strane drugih korisnika. Korisnik ima mogućnost postaviti novi pitanje ili pristupiti razgovoru vezanom uz jedno od ponuđenih pitanja i pružiti odgovor na njega ili se pridružiti diskusiji. Prilikom klika na neki od već podnesenih odgovora, otvara se profil korisnika koji je napisao odabrani odgovor. Ovdje se otvara mogućnost pozitivnog ocjenjivanja tog odgovora, te se može zatražiti kontakt osobe koja je odgovor napisala (Slika 3.3.).



Slika 3.3. Prikaz ekrana aktivnosti s popisom TV serija, ekrana za odabir teme TV serije prilikom pristupa u chat, te ekrana koji se otvara prilikom klika na neku od osoba koja sudjeluje u forumu

3.4.4. Aktivnost s popisom prijatelja

Kada se iz aktivnosti `Forum.java` klikne na neku od osoba u razgovoru, korisnik može zatražiti kontakt te osobe. Prilikom registracije, svaki korisnik unosi o sebi neki oblik kontakta po želji (Facebook ime, broj telefona, e-mail adresa). Ako netko zatraži njihov kontakt, bit će o tome obaviješteni. Ako odluče podijeliti kontakt s drugim korisnikom, tom korisniku pojavit će se u aktivnosti s popisom prijatelja. Ova aktivnost služi kako bi prilikom eventualne nadogradnje aplikacije (engl. *premium plan*), trenutni korisnik imao mogućnost pozvati nekog od prijatelja u željeni chat ili diskusiju. U priloženoj aplikaciji, moguće je prisustvovati razgovoru ili diskusiji samo s osobama odabranim nasumičnim odabirom.

3.4.5. Aktivnost s prikazom profila korisnika

Kada korisnik odabere prikaz svog profila, prikazuju se njegove informacije prikupljene prilikom registracije. Ovdje korisnik može promijeniti svoje podatke te dodati ili ukloniti TV serije koje prati. Korisniku su ovdje također prikazane notifikacije. Primjerice, notifikacija koja korisnika obavještava kada drugi korisnik zatraži njegov kontakt.

3.5. Razrada dijelova kôda

Dijeljenje korisnika u skupine prilikom razgovora postignuto je dijelovima kôda u `ChatSeries.java` i `ChatRoomActivity.java` aktivnostima. Prilikom pristupa u Chat, korisnici se dijele u skupine po pet osoba. U bazi podataka Firebase, tada se stvaraju zapisi. Primjerice, ako chat s temom serije naziva „Veep“[26] još ne postoji, ili ako postoji i ima manje od 5 korisnika, ime korisnika koji mu želi pristupiti upisuje se pod sudionike chata s nazivom „Veep“. Međutim, ako u njemu već postoji pet sudionika, stvara se nova tablica naziva „Veep1“. Kada se ona popuni, stvara se tablica „Veep2“ i tako dalje.

U aktivnosti `ChatSeries.java`, prilikom klika na naziv određene serije, prije pokretanja aktivnosti `ChatRoomActivity.java`, provjerava se postoji li ime serije odabranog chata u bazi podataka, koliko korisnika broji, te sadržava li baza naziv tog chata s brojem kraj njegovog imena (npr. „Veep1“).

Prikaz poziva metode *howMany (String s, int i)* nakon pritiska na neku od serija u listi:

```
usersList.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int
    position, long id) {
        UserDetails.chatWith = al.get(position);
        howMany(UserDetails.chatWith, 0);
    }
});
```

Kôd 7. Poziv metode *howMany (String s, int i)*, prilikom klika na željeni chat u `ChatSeries.java` klasi

Metoda naziva *howMany (String s, int i)* zatim provjerava podatke u bazi za zadanu seriju. Inicijalno, broj chata u varijabli *chatNumber* iznosi 0, što će pozvati metodu *doOnSuccessHowMany (String sa)*. Ako ime chata te serije još ne postoji u bazi, u tom chatu još nema korisnika i metoda *doOnSuccessHowMany (String sa)* pozvat će metodu *howMany (String s, int i)* s parametrom *chatNumber* koji iznosi -1, te će se otvoriti aktivnost `ChatRoomActivity.java`. Ako pak ime postoji, dakle kada metoda *doOnSuccessHowMany (String sa)* pokaže da već postoji određeni broj korisnika u chatu s temom odabrane serije, *chatNumber* se povećava. Ako *chatNumber* završi na broju 0, znači da naziv tog chata postoji i broj ima manje od 5 korisnika. Tada se otvara aktivnost `ChatRoomActivity.java`. Ako postoji više od 5 korisnika, metode se izmjenjuju dok

[26] Naziv američke TV serije

se ne dobije ukupni broj korisnika, odnosno broj naziva koji sadrže naziv te serije(„Veep“, „Veep1“, „Veep2“, „Veep3“…). Na ovaj način dobiva se ukupan broj korisnika u svim chat-ovima vezanim uz odabranu seriju, te broj koji stoji uz ime zadnjeg chata vezanog uz seriju, kako bi korisnici bili pravilno raspoređeni.

Prikaz kôda *howMany (String s, int i)* metode*:*

```
public void howMany(String st, int chatNumber){

    if(chatNumber ==-1) {
        startActivity(new Intent(Users.this, ChatRoomActivity.class));
    }
    else if(chatNumber ==0) {
        String url = "https://androidchatapp-
        63888.firebaseio.com/messages/" + st + "/participants.json";
        StringRequest requesta = new StringRequest(Request.Method.GET,
        url, new Response.Listener<String>() {
            @Override
            public void onResponse(String sa) {
                doOnSuccessHowMany(sa);
            }
        }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError volleyError) {
                System.out.println("" + volleyError);
            }
        });
        RequestQueue rQueue = Volley.newRequestQueue(Users.this);
        rQueue.add(requesta);
    }
    else{
        String url = "https://androidchatapp-
        63888.firebaseio.com/messages/" + st+ chatNumber +
        "/participants.json";
        StringRequest requesta = new StringRequest(Request.Method.GET,
        url, new Response.Listener<String>() {
            @Override
            public void onResponse(String sa) {
                if(sa.isEmpty()){
                    startActivity(new Intent(Users.this,
                    ChatRoomActivity.class));
                }
                else{
                    doOnSuccessHowMany(sa);
                }
            }
        }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError volleyError) {
                System.out.println("" + volleyError);
            }
        });
        RequestQueue rQueue = Volley.newRequestQueue(Users.this);
        rQueue.add(requesta);
    }
}
```

Kôd 8. *howMany (String s, int i)* metoda

Slijedi prikaz kôda metode *doOnSuccessHowMany (String sa)*:

```
int chatNumber =0;
public  void doOnSuccessHowMany(String sa){
    try {
        totalUsersa=0;
        if(String.valueOf(sa).equals("null")){
            Log.i("je prazan nijee chosen", String.valueOf(sa));
            howMany(UserDetails.chatWith, -1);
            chatNumber =-1;
        }

        JSONObject obj = new JSONObject(sa);

        Iterator i = obj.keys();
        String key = "";

        while (i.hasNext()) {
            key = i.next().toString();
            ala.add(key);
            totalUsersa++;
            ChatDetails.totalChatUsers ++;
            if(key.equals(UserDetails.username)){
                ChatDetails.currentChatNumber = chatNumber;
            }
        }

        ChatDetails.totalUsr=totalUsersa;
        if(totalUsersa==6){
            chatNumber ++;
            howMany(UserDetails.chatWith, chatNumber);
        }
        else{
            ChatDetails. lastNumber = chatNumber;
            startActivity(new Intent(Users.this,
            ChatRoomActivity.class));
        }

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

Kôd 9. *doOnSuccessHowMany (String sa)* metoda

Kada se otvori aktivnost `ChatRoomActivity.java`, korisnike se pridružuje pripadajućim chat-ovima na osnovu informacija dobivenih u `ChatSeries.java` aktivnosti. Neke od njih su: ukupan broj korisnika koji sudjeluju u svim razgovorima vezanim uz odabranu seriju, broj chata u kojem se potencijalno već od prije nalazi trenutni korisnik, te zadnji broj chata koji trenutno postoji vezan uz odabranu TV seriju. Informacije su pohranjene u klasi `ChatDetails.java`.

Dio kôda iz `ChatRoomActivity.java` klase, koji na temelju dobivenih podataka određuje sadržaj reference *reference1*, te se pomoću dobivene putanje otvara chat prigodan za trenutnog korisnika:

```
if(ChatDetails.totalChatUsers>6 && ChatDetails.currentChatNumber==-1){
    Integer chatBroj=ChatDetails.totalChatUsers/6;
    reference1 = new Firebase("https://androidchatapp-
    63888.firebaseio.com/messages/" + UserDetails.chatWith+chatBroj);
}

else if(ChatDetails.totalChatUsers <=6 && ChatDetails.currentChatNumber
==-1){
    reference1 = new Firebase("https://androidchatapp-
    63888.firebaseio.com/messages/" + UserDetails.chatWith);
}

else if(ChatDetails.totalChatUsers >6 && ChatDetails.currentChatNumber
==-1){
    Integer lastNumber=ChatDetails. lastNumber +1;
    reference1 = new Firebase("https://androidchatapp-
    63888.firebaseio.com/messages/" + UserDetails.chatWith+ lastNumber);
}

else if(ChatDetails.currentChatNumber >-1) {
    Integer a=ChatDetails.currentChatNumber;
    Integer b=a-1;
    if(a==0){
        reference1 = new Firebase("https://androidchatapp-
        63888.firebaseio.com/messages/" + UserDetails.chatWith);
    }
    else {
        reference1 = new Firebase("https://androidchatapp-
        63888.firebaseio.com/messages/" + UserDetails.chatWith + a);
    }
}
else {
    reference1 = new Firebase("https://androidchatapp-
    63888.firebaseio.com/messages/" + UserDetails.chatWith);
}
```

Kôd 10. „if“ funkcija iz `ChatRoomActivity.java` klase koja pomaže razvrstavanju korisnika u odgovarajuće razgovore

4. Jako velike baze podataka (engl. *Very Large Databases*, skraćeno VLDBs)

Jako velike baze podataka (skraćeno VLDBs) baze su koje sadržavaju izuzetno puno podataka. Prema važećoj definiciji, to su baze koje zaokupljaju više od jednog terabajta (engl. *terabyte*, skraćeno TB, ekvivalentno 1024 gigabajta) prostora. Međutim, u skoroj budućnosti taj termin vjerojatno će koristiti se za baze veće od nekoliko desetaka TB, zahvaljujući sve većem kapacitetu pohrane podataka i sve manjim troškovima. (Elmasri et al., 2015:615)

Potrebna je značajna količina fizičkog prostora za spremanje VLDBs-a. To su često baze koje se koriste u sustavima za potporu odlučivanju ili aplikacijama za provedbu transakcija.[27]

The Winter Corporation naziv je za organizaciju koja provodi istraživanja o najvećim i najčešće korištenim bazama podataka. U jednoj od procjena pružatelje usluga VLDBs-a, dotične je rangirao po veličini prostora koji je potreban za održavanje jako velikih baza podataka.[28] Prvih 10 kompanija, u istraživanja provedenih 2005. godine, prikazano je na slici (**Pogreška! Izvor reference nije pronađen.**.).[29] U ovu kategoriju spadaju sustavi skladištenja podataka temeljeni na bilo kojem operacijskom sustavu.

[27]Tech target. Dostupno na: http://searchoracle.techtarget.com/definition/Very-Large-Database [10. veljače 2018.]
[28] ITProToday. Dostupno na: http://www.itprotoday.com/microsoft-sql-server/sql-server-and-vldb-playing-big-boys [10. veljače 2018.]
[29] CISION. Dostupno na: http://www.prweb.com/releases/2005/03/prweb216653.htm [10. veljače 2018.]

Tablica 4.1. Najveći sustavi skladištenja podataka prema istraživanju provedenom 2005. godine od strane The Winter Corporation-a[30]

2005 Winter TopTen Award Winners
14 September 2005

Database Size, All Environments, DW

Company/Organization	Database Size (GB)	DBMS	Platform	Architecture	DBMS Vendor	System Vendor	Storage Vendor
Yahoo!	100,386	Oracle	UNIX	Centralized/SMP	Oracle	Fujitsu Siemens	EMC
AT&T	93,876	Daytona	UNIX	Federated/SMP	AT&T	HP	HP
KT IT-Group	49,397	DB2	UNIX	Centralized/Cluster	IBM	IBM	Hitachi
AT&T	26,713	Daytona	UNIX	Federated/SMP	AT&T	Sun	Sun
LGR - Cingular Wireless	25,203	Oracle	UNIX	Centralized/SMP	Oracle	HP	HP
Amazon.com	24,773	Oracle RAC	Linux	Centralized/Cluster	Oracle	HP	HP
Anonymous	19,654	DB2	UNIX	Centralized/MPP	IBM	IBM	EMC
UPSS	19,467	SQL Server	Windows	Centralized/SMP	Microsoft	Unisys	EMC
Amazon.com	18,558	Oracle RAC	Linux	Centralized/Cluster	Oracle	HP	HP
Nielsen Media Research	17,685	Sybase IQ	UNIX	Centralized/SMP	Sybase	Sun	EMC

Veličine baza nekih od korporacija koje koriste jako velike baze podataka iz 2010. godine:

- NERSC (engl. National Energy Research Scientific Computing Center) – 2.8 petabajta (engl. *petabyte*, skraćeno PB), ekvivalentno 1000 TB podataka
- World Dana Centre for Climate – 220 TB uz 6 PB dodatnih podataka
- AT&T – 323 TB podataka
- Google – približno 33 trilijuna redaka pohranjenih i bazi podataka
- ChoicePoint – 250 TB podataka

VLDBs baze podataka zahtjevne su kako za administriranje, tako i za dizajniranje arhitekture baze. Odluke o strukturiranju baza trebaju biti promišljene, uz pridavanje pažnje mogućim promjenama. Također, postoji mogućnost za potrebu za proširenjem u budućnosti, s obzirom na to da tehnologija svakim danom rapidno napreduje. (Coronel et al., 2016:601)

Podjela ukupne količine podataka u dijelove (engl. *partition*), praksa je koja se provodi kako bi se tablice podataka mogle pohranjivati na različitim setovima upravljačkih programa (engl. *drivers*). Dijelovi na koje su tablice podijeljene indeksiraju se, te im se dodjeljuju primarni ključevi. Vrijeme oporavka za ovako strukturirane baze podataka znatno se

[30]WinterCorp, Web Archive. Dostupno na: https://web.archive.org/web/20120312175519/http://www.wintercorp.com/VLDB/2005_TopTen_Survey/2005TopTenWinners.pdf [20. veljače 2018.]

skraćuje. Da bi baza pravilno funkcionirala, potrebno je što efikasnije i u što kraćem roku eliminirati sve indekse koji se više ne koriste.

Kako bi se minimizirala mogućnost gubitka podataka, VLDBs baze podataka potrebno je pravilno održavati. Ako dođe do potrebe za vraćanjem izgubljenih podataka iz baze, sukladno veličini baze, vrijeme oporavka (engl. *recovery*) se također produžuje.

Prilikom skladištenja podataka, mogućnosti pohrane podataka jako velikih baza podataka ovise o broju zahtjeva ulaza i izlaza (engl. *input/output*, skraćeno I/O), koje je moguće provesti u sekundi vremena (engl. *I/O Requests Per Second,* skraćeno IOPS).

Razina dostupnosti VLDB-a postiže se tehnikama zrcaljenja (engl. *mirroring*). Neke od tehnika zrcaljenja su:

- Zrcaljenje pomoću hardvera – pomoću RAID 1 i RAID 5 tehnika zrcaljenja
- Zrcaljenje pomoću ASM-a – uz menadžer automatske pohrane (engl. *Automatic Storage Manager*, skraćeno ASM), zrcaljenje se postiže korištenjem servera baza podataka
- Zrcaljenje uz pomoć softvera, bez ASM-a[31]

Kako bi jako velike baze podataka pružale optimalne performanse, diskovi moraju paralelno raditi. Koristi se takozvana tehnika dijeljenja podataka (engl. *striping*). Blokovi podataka pohranjuju se u nekoliko uređaja za pohranu podataka, u dijelovima jednakih veličina. Oracle se zalaže za zrcaljenje i pohranu podataka u dijelovima jednakih veličina (engl. *Stripe and Mirror Everything*, skraćeno S.A.M.E.). Takva metodologija implementira AMS, te ju Oracle preporučuju za sve baze podataka veće od 1 MB. [32]

Zbog veličina VLDB-a, kada se dodaju novi podatci, veličina novih podataka relativno je mala u odnosu na veličinu same baze. To može dovesti do previda stvarne veličine novih podataka. Međutim, kada se administracija vrši u skladu s propisanim praksama, performanse ovakvih baza podataka ne variraju mnogo od performansi običnih baza podataka, mnogo manjih po veličini.

[31] Toad World. Dostupno na: https://www.toadworld.com/platforms/sql-server/w/wiki/9453.very-large-databases-vldbs [10. veljače 2018.]
[32] Oracle. Dostupno na; https://docs.oracle.com/cd/E11882_01/server.112/e25523/intro.htm [10. veljače 2018.]

4.1. Implementirana rješenja i načini pohranjivanja jako velikih baza podataka

Bryce Canyon naziv je Facebook-ove platforme za pohranu podataka nove generacije, uvedene početkom prošle godine. Namijenjena je optimizaciji rukovanja podatcima vizualnog sadržaja. Dizajn je otvorenog tipa (engl. *Open Computer Project*, skraćeno OCP). Danas, Facebook pohranjuje više od nekoliko milijuna TB podataka, a ta brojka svakodnevno raste, trenutno dostigavši 2 milijarde korisnika.[33]

Bryce Canyon sustav je za pohranu podataka velike gustoće. U prostoru u kojem se ranije moglo pohraniti 16 tvrdih diskova (engl. *Hard Disc Drive*, skraćeno HDD), sada se može pohraniti 72 HDD-a. Koriste se HDD tvrdi diskovi koji funkcioniraju na bazi helija, odnosno svjetlosti karakteristične za helij. Time se povećava brzina diska i povećava gustoća pohranjenih podataka. Takvo povećanje gustoće podataka moguće je zbog gustoća helija, koja je manja od gustoće zraka. Usporedba HDD-a koji radi na bazi helija s HDD-om na bazi zraka prikazana je na slici (Slika 4.1.).[34]

Ova platforma također koristi Mono Lake mikro servere, s kojima je moguća prenamjena i ponovna upotreba takvih servera. Mono Lake serveri podnose opterećenja prilikom obavljanja mnogo paralelnih zadataka. Kako bi se memorija optimalno iskoristila, koristi se Tioga Pass (nova inačica Leopard-a). Tioga Pass koristi matičnu ploču s dva utora koja podržava jednostrane i dvostrane dizajne s dualnim memorijskim modulima (engl. *Dual in-line memory module*, skraćeno DIMM) s obje strane sklopne ploče (engl. *Printed Circuit Bord*, skraćeno PCB) koja spaja električne komponente matične ploče.[35]

Facebook također ulaže milijarde dolara godišnje za izgradnju podatkovnih centara diljem Sjedinjenih Američkih Država.

---

[33] Digital Trends. Dostupno na: https://www.digitaltrends.com/computing/facebook-server-hardware/ [13. veljače 2018.]
[34] Tech target. Dostupno na: http://whatis.techtarget.com/definition/helium-hard-drive [13. veljače 2018.]
[35]Cisco. Dostupno na: https://blogs.cisco.com/datacenter/is-facebooks-new-storage-platform-performance-hungry [13. veljače 2018.]

Slika 4.1. Prikaz usporedbe tvrdog diska na bazi zraka i na bazi helija, koji se koristi u Facebook-ovoj platformi za pohranu podataka Bryce Canyon.[36]

4.2. Gorilla

Gorilla je naziv za sistem baze podataka koji je automatiziran, brz, skalabilan i prati događaje u stvarnom vremenu. Facebook kompanija koristi ga za praćenje događaja u svojoj infrastrukturi. Cilj mu je trenutni identifikacija problema, kako bi se oni mogli riješiti u minimalnom roku s minimalnom štetom. [37]

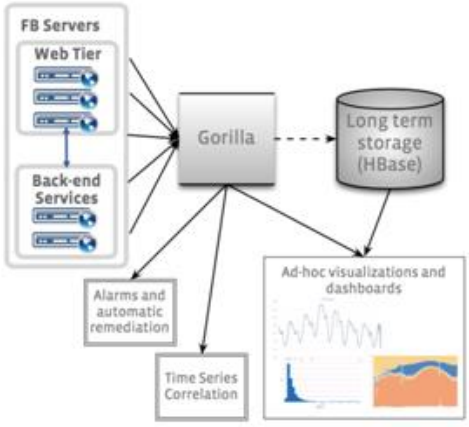
Zbog količine Facebook-ovih elemenata i servisa, te količine korisnika koji ga simultano koriste, stvorena je potreba za Gorilla sistemom. On konstantno prati performanse i zdravlje svih komponenata u realnom vremenu, te ako je potrebno, u najkraćem roku obavještava o nastalim greškama i anomalijama prilikom korištenja.[38]

Ovakav sustav praćenja rada, u jednom danu bilježi i obradi više od jednog trilijuna podataka u točki vremena, prilikom čega analizira iste. Gorilla funkcionira na temelju vremensko serijskih softvera baza podataka (engl. *Time Series Database*, skraćeno TSDB). *Time Series Database* softver služi za optimizaciju prikupljanja podataka, koji se spremaju u obliku

[36] Tom's IT PRO. Dostupno na: http://www.tomsitpro.com/articles/hgst-helium-hdd-helioseal,1-2522.html [13. veljače 2018.]
[37] Aleksey Charapko. Dostupno na: http://charap.co/gorilla-facebooks-cache-for-monitoring-data/ [13. veljače 2018.]
[38]Storage Switzerland. Dostupno na: https://storageswiss.com/2016/02/02/why-low-latency-matters/ [13. veljače 2018.]

nizova indeksiranih točaka vremena prikupljenih tijekom korištenja programa. Arhitektura sustava za praćenje podataka Gorilla prikazana je na slici (Slika 4.2.).[39]

Kôd biblioteka koje koristi Gorilla sustav pisan je u C++ programskom jeziku. Svi dobivene podatci pohranjuju se u GlusterFS. GlusterFS arhitektura je koja na optimalan način pohranjuje podatke i izvršava potrebne računske operacije nad njima. Primjerice, ako program odredi da je dobivena vrijednost ista prethodnoj, ne sprema se njegova puna vrijednost i ne stvara kopija, već se ta informacija sprema se u obliku samo jednog bita. Tako se smanjuje veličina iskorištenog prostora za spremanje podataka.[40]



Slika 4.2. Arhitektura Gorilla sustava za praćenje podataka[41]

Ukupnim podatcima dodjeljuju se prioriteti sukladno vremenu u kojem su dobiveni. Prednost imaju podatci najbliži trenutnom vremenu. U ovakvom tipu baze podataka, kojoj je zadaća praćenje podataka u stvarnom vremenu, važnu ulogu ima i latentnost. Latentnost je vrijeme koje je potrebno da se ostvari potpuni proces dobivanja odgovora, od automatskog unosa podataka koji su prikupljeni, do izlaza istih. Također, sustav treba uvijek biti raspoloživ i u mogućnosti pratiti događaje, čak i kada je dio sustava srušen. Uz rizik gubitka manjih količina podataka, potrebno je pratiti koliko je grešaka u određenom trenutku identificirano, paralelno uz sve popratne aktivnosti sustava.[42]

Nova inačica ovog sustava naziva se Beringei, a uvedena je 2017. godine. Međutim, ona nije u potpunosti zamijenila sve komponente sustava Gorilla. Beringei je sustav otvorenog kôda,

[39]The morning paper. Dostupno na: https://blog.acolyer.org/2016/05/03/gorilla-a-fast-scalable-in-memory-time-series-database/ [14. veljače 2018.]
[40]Seagate blog. Dostupno na: https://blog.seagate.com/intelligent/facebooks-new-storage-platform-design-enables-capacity-to-scale-exponentially-easily-and-affordably/ [14. veljače 2018.]
[41] Facebook vldb 2015. Dostupno na: http://slideplayer.com/slide/10784539/, slide 11 [14. veljače 2018.]
[42] Facebook vldb 2015. Dostupno na: http://slideplayer.com/slide/10784539/ [14. veljače 2018.]

a razlikuje se od drugih sustava za praćenje događaja jer je prilikom njene izrade akcent stavljen na praćenje zdravlja cjelokupnog sustava, a ne na manje događaje unutar njega. Tako su mogući propusti koji utječu na male skupine ljudi, koje su beznačajne u odnosu na ukupnu količinu korisnika.[43]

[43] Architecht. Dostupno na: https://architecht.io/facebook-open-sourced-its-time-series-database-for-systems-monitoring-aedf3c603f9f [14. veljače 2018.]

5. Testiranje aplikacije

Testiranje je izvršavanje programa aplikacije u svrhu pronalaska mogućih grešaka u programu aplikacije. Testiranja se provode kako bi se maksimalan broj grešaka mogao ispraviti prije no što aplikacija izađe na tržište kao potpuni proizvod.

Prvi korak testiranja aplikacije naziva se Alpha testiranje. Da bi testna skupina aplikaciju mogla preuzeti s Google Play platforme, vlasnik aplikacije prvo ju mora objaviti. Testnu skupinu korisnika potrebno je odabrati prema određenim parametrima. Primjerice, koriste li adekvatne mobilne uređaje na kojima će pokretati aplikaciju. Poželjno imati što veću raznolikost uređaja, kako bi se testiralo pokretanja aplikacije na raznim verzijama mobilnih telefona. Također, poželjno je da testna grupa korisnika koristi aplikacije slične aplikaciji koju će testirati. Tako je programskim inženjerima omogućen uvid u razinu jednostavnosti i intuitivnosti u korištenju aplikacije, ovisno o snalaženju korisnika u navigaciji kroz funkcionalnosti aplikacije. Posljednje, šalje se link svim Alpha testnim korisnicima kako bi mogli pristupiti aplikaciji. [44]

Ovaj način testiranja provodi se kada aplikacija još ne sadrži sve predviđene funkcionalnosti, međutim stabilna je i funkcionalna. Cilj ovakvog testiranja pronalazak je bugova u aplikaciji, te događaja koji prethode padu aplikacije kako bi se iste moglo eliminirati.

Nakon Alpha testiranja, dolazi testiranje Beta verzije aplikacije. Ovdje se također prati broj bugova i pada aplikacije, te se prati napredak u usporedbi s Alpha testiranjem. Nakon ovakvog testiranja uvode se manje promjene u aplikaciji prije lansiranja finalnog proizvoda na tržište.[45]

5.1. Mogućnosti testiranja aplikacija

U sklopu Firebase platforme za razvoj mobilnih aplikacija, nalazi se laboratorij za testiranje Android aplikacija (engl. *Firebase Test Lab for Android*). Pomoću njega, moguće se testirati aplikaciju na nizu simulacija različitih verzija mobilnih uređaja s različitim konfiguracijama.

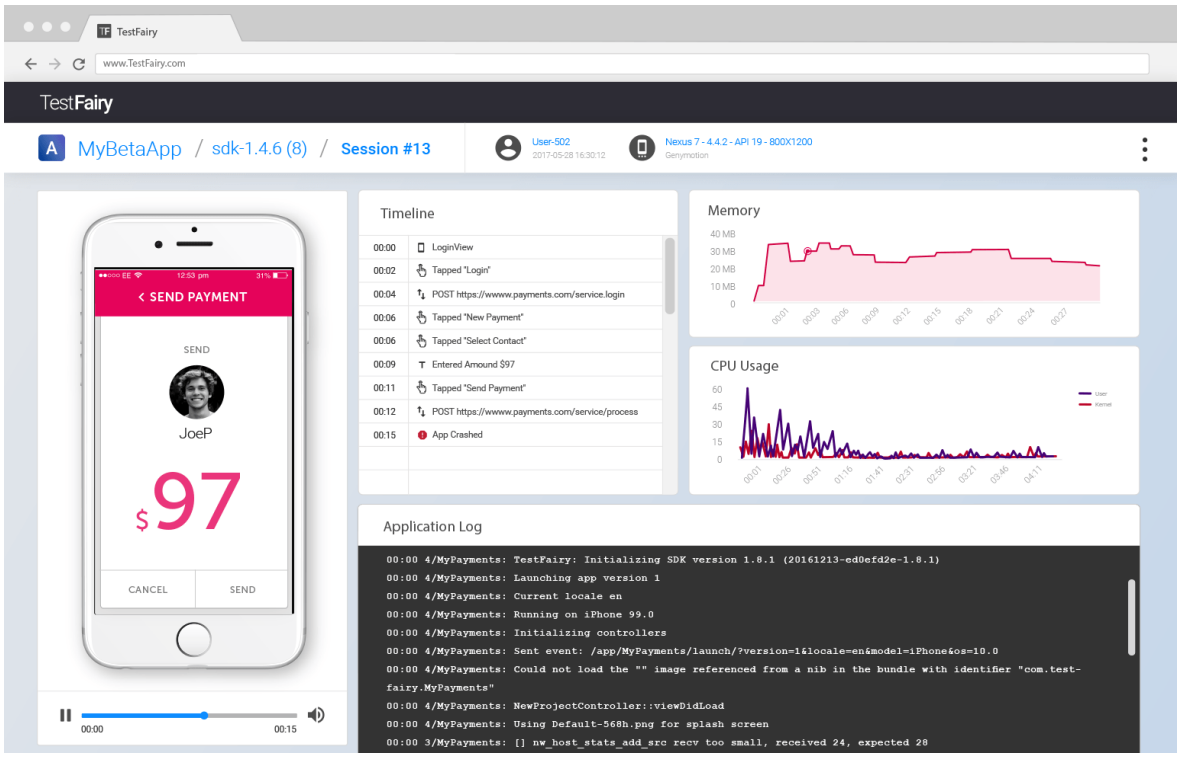
[44] Android developer. Dostupno na: https://support.google.com/googleplay/android-developer/answer/3131213?hl=en- [20. veljače 2018.]
[45] Centercode. Dostupno na: https://www.centercode.com/blog/2011/01/alpha-vs-beta-testing/ [20. veljače 2018.]

Prilikom ovakvog testiranja, Firebase-ov program automatski pronalazi i ispisuje bugove, te padove aplikacije u određenim okolnostima.[46] Firebase-ova saznanju o rušenju aplikacije (engl. *Firebase Crash Report*) zapisuju se u izvještaj koji sadržava događaje koji su prethodili padu aplikacije, te moguće razloge pada aplikacije u određenim verzijama mobilnih uređaja.[47]

Platforme kao što su TestFairy, HockeyApp i Ubertesters služe za provođenje Beta testiranja Android aplikacija.[48] Razvojnim programerima aplikacija daju uvid u snimke ekrana mobilnih telefona prilikom korištenja aplikacije i ponašanje korisnika pri njezinom korištenju. Također, dostupni su im i zapisnici o ponašanju aplikacije tijekom njezina korištenja zbog lakšeg shvaćanja i rješavanja problema s klijentske strane. Također, u slučaju pada aplikacije, dostupni su detaljni izvještaji o tijeku događaja koji su doveli do rušenja aplikacije. Primjer izgleda aplikacije TestFairy tijekom provedbe testova prikazan je na slici (Slika 5.1.).[49]

UserTesting platforma je koja plaća svojim korisnicima testiranje mobilnih aplikacija i web stranica. Korisnici koji žele testirati svoju aplikaciju mogu odabrati neke od ponuđenih profila korisnika. Neki od dostupnih podataka članova skupine koja će testirati aplikaciju su dob, mjesto prebivališta, aplikacije koje su do sada imali priliku koristiti, te područje posla kojim se bave. Naručitelji testiranja odabiru niz pitanja koja su testnim korisnicima postavljena prilikom korištenja njihove aplikacije. Tijekom testa, ekran testnih korisnika se snima, uz audio snimku njihovih odgovora na postavljena pitanja prilikom korištenja aplikacije. Također, testni korisnici mogu dobiti i zadatke kao što su pronalazak određenih informacija na aplikaciji. Na temelju njihovih odgovora dobivaju se podatci o razini intuitivnosti i jednostavnosti korištenja aplikacije. [50]

[46] Firebase. Dostupno na: https://firebase.google.com/docs/test-lab/ [20. veljače 2018.]
[47] Firebase. Dostupno na: https://firebase.google.com/docs/crash/ [20. veljače 2018.]
[48] SitePoint. Dostupno na: https://www.sitepoint.com/5-mobile-app-testing-tools/ [20. veljače 2018.]
[49] TestFairy. Dostupno na: https://testfairy.com/ [20. veljače 2018.]
[50] UserTesting. Dostupno na: https://www.usertesting.com/ [20. veljače 2018.]



Slika 5.1. Primjer izgleda TestFairy sučelja prilikom testiranja aplikacije

Zaključak

U današnje vrijeme, zbog same količine dostupnih mobilnih aplikacije, sve je više onih koje liče jedna na drugu po sadržaju, dizajnu ili namjeni. Nakon nekoliko poznatih primjera mobilnih aplikacija koje su postigle uspjeh preko noći, naglasak se više ne stavlja na kvalitetu sadržaja, već na što bržu provedbu i brz, kratkoročan uspjeh. Većina aplikacija uspijeva postići uspjeh zbog velikih investicija u reklamiranje namijenjeno velikim količinama ljudi. Također, sve je više investitora koji ulažu isključivo u mobilne aplikacije.

Iako je stvarna potreba i korist većine aplikacija diskutabilna, to ne utječe na njihovu popularnost. Ne smijemo zaboraviti da aplikacije koje su nekada počele s isključivo sadržajem namijenjenim za zabavu, kao što su Facebook, Instagram ili YouTube, sada uključuju i koristan sadržaj. Primjerice, Facebook-ova opcija označavanja osobe kao „siguran“ prilikom vremenskih nepogoda. Nadalje, aplikacije Instagram i YouTube osiguravaju stabilnost i plaću osobama koje izrađuju popularan sadržaj, većinom mladim ljudima. Bilo da je takva vrsta sadržaja napravljena iz osobne koristi tvrtke, s ciljem privlačenja više korisnika, to ne osporava dobrobit i korist društvu koju su one donijele.

TV serije dio su moderne kulture. Inspiriraju gledatelje, podučavaju ih životnim mogućnostima, pomažu razumijevanju načina razmišljanja različitih ljudi i kultura, te imaju moć utjecati na stvarni život i razmišljanja gledaoca. Ovdje priložena aplikacija namijenjena je za zabavu, međutim, u budućnosti može poslužiti kao osnova za analizu dubljih društvenih trendova i načina razmišljanja mladih osoba. Određeni okidači u radnji TV serije mogu potaknuti dublje razgovore, te postati temelj za analizu međusobnih odnosa ljudi, kao i aspekte njihova pogleda na svijet oko sebe ili na scenarije koji su mogući samo na malim ekranima.

Popis kratica

AOT	*Ahead-of-time*	prijevremeni
API	*Application Programming Interface*	sučelje za programiranje aplikacija
ART	*Android Runtime*	Android dužina vremena za pokretanje
ASM	*Automatic Storage Manager*	menadžer automatske pohrane
DAG	*Directed acyclic graph*	usmjereni aciklički graf
DIMM	*Dual in-line memory module*	dualni linijski memorijski modul
FSFLA	*Free Software Foundation Latin America*	organizacija za besplatne softvere
HAL	*Hardware Abstraction Layer*	sloj za apstrakciju sklopovlja
HDD	*Hard Disc Drive*	tvrdi disk
I/O	*Input/Output*	ulaz/izlaz
IDE	*Integrated Development Environment*	integrirano razvojno okruženje
IOPS	*Input/Output requests Per Second*	zahtjevi ulaza i izlaza koje je moguće provesti u sekundi vremena
IPC	*Inter-process communication*	međuprocesna komunikacija
JSON	*JavaScript Object Notation*	JavaScript objekt
MB	*Megabyte*	megabajt
NERSC	*National Energy Research Scientific Computing Center*	nacionalni centar koji se bavi znanstvenim istraživanjima energije
OCP	*Open Computer Project*	projekt otvorenog tipa
PB	*Petabyte*	petabajt
PCB	*Printed Circuit Board*	sklopna ploča
RAM	*Random Access Memory*	memorija s nasumičnim pristupom
S.A.M.E.	*Stripe and Mirror Everything*	primjena zrcaljenja i pohrane podataka u dijelove jednakih veličina na sve
TB	*Terabyte*	terabajt
TMDb	*The Movie Database*	baza podataka s podacima o filmovima
TSDB	*Time Series Database*	vremensko serijska baza podataka
URL	*Uniform Resource Locator*	lokator resursa
VLDB	*Very Large Database*	jako velika baza podataka

Popis slika

Popis tablica

Popis kôdova

Literatura

[1] Lifewire. Dostupno na: https://www.lifewire.com/what-is-google-android-1616887 [18. veljače 2018.]

[2] Android Source. Dostupno na: https://source.android.com/compatibility/7.1/android-7.1-cdd.pdf [20. veljače 2018.]

[3] BOŽIĆ, J., KADEŽABEK, T., TOMIĆ, F., KAŠTELAN, T., *Izrada aplikacija za mobilne uređaje*. Zagreb: Algebra, 2013.

[4] WANG, P. S., *Mastering Linux*. New York: A Chapman & Hall Book, 2011.

[5] ICT business. Dostupno na: http://www.ictbusiness.info/vijesti/novi-android-runtime-ce-zamijeniti-dosadasnji-dalvik [15. veljače 2018.]

[6] Tutorials point. Dostupno na: https://www.tutorialspoint.com/android/android_architecture.htm [15. veljače 2018.]

[7] Android. Dostupno na: devhttps://developer.android.com/guide/platform/index.html [15. veljače 2018.]

[8] Android. Dostupno na: https://developer.android.com/studio/intro/index.html [16. veljače 2018.]

[9] Android. Dostupno na: https://android-developers.googleblog.com/2013/05/android-studio-ide-built-for-android.html [15. veljače 2018.]

[10] AndroidPub. Dostupno na: https://android.jlelse.eu/mastering-viewpager-with-directed-acyclic-graph-533ed5ee1e5e [15. veljače 2018.]

[11] Kinetic Growth. Dostupno na: http://www.kineticgrowth.com/make-eclipse-use-memory-ram-efficiently/ [15. veljače 2018.]

[12] Eclipse. Dostupno na: https://www.eclipse.org/ide/ [15. veljače 2018.]

[13] Macworld. Dostupno na: https://www.macworld.com/article/1061005/androidsdk.html [16. veljače 2018.]

[14] Android. Dostupno na: https://developer.android.com/studio/intro/index.html [16. veljače 2018.]

[15] Prizvuk. Dostupno na: http://www.prizvuk.hr/old/hr_widgets.html [16. veljače 2018.]

[16] Android Authority. Dostupno na: https://www.androidauthority.com/best-material-design-apps-for-android-523420/ [16. veljače 2018.]

[17] Android. Dostupno na: https://developer.android.com/design/material/index.html [16. veljače 2018.]

[18] Material Design. Dostupno na: https://material.io/guidelines/ [16. veljače 2018.]

[19] LWN. Dostupno na: https://lwn.net/Articles/376566/ [18. veljače 2018.]

[20] FSF Linux-libre. Dostupno na: https://www.fsfla.org/ikiwiki/selibre/linux-libre/ [18. veljače 2018.]

[21] SitePoint. Dostupno na: https://www.sitepoint.com/volley-a-networking-library-for-android/ [18. veljače 2018.]
[22] GitHub. Dostupno na: https://github.com/google/volley [18. veljače 2018.]
[23] Rich ard Rose Blog. Dostupno na: https://richardroseblog.wordpress.com/2016/05/06/jolly-volley-android-networking-library/ [18. veljače 2018.]
[24] Firebase. Dostupno na: https://firebase.google.com/docs/ [18. veljače 2018.]
[25] Firebase. Dostupno na: https://firebase.google.com/docs/database/android/start/ [18. veljače 2018.]
[26] IGI Global. Dostupno na: https://www.igi-global.com/dictionary/api/1298 [18. veljače 2018.]
[27] TheMovieDatabase. Dostupno na: https://www.themoviedb.org/documentation/api [18. veljače 2018.]
[28] ELMASRI, R., NAVATHE, S.B., *Fundamentals od Database Systems. Pearson*, 2015. http://lms.uop.edu.jo/lms/pluginfile.php/2376/mod_resource/content/0/Fundamentals_of_Database_Systems%2C_6th_Edition.pdf [10. veljače 2018.]
[29] Tech target. Dostupno na: http://searchoracle.techtarget.com/definition/Very-Large-Database [10. veljače 2018.]
[30] ITProToday. Dostupno na: http://www.itprotoday.com/microsoft-sql-server/sql-server-and-vldb-playing-big-boys [10. veljače 2018.]
[31] CISION. Dostupno na: http://www.prweb.com/releases/2005/03/prweb216653.htm [10. veljače 2018.]
[32] CORONEL, C., MORRIS, S., Database Systems: *Design, Implementation, & Management*. Cengage Learning, 2016 , veljača 2018. https://books.google.hr/books?id=vDIaCgAAQBAJ&printsec=frontcover&hl=hr&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false [10. veljače 2018.]
[33] WinterCorp, Web Archive. Dostupno na: https://web.archive.org/web/20120312175519/http://www.wintercorp.com/VLDB/2005_TopTen_Survey/2005TopTenWinners.pdf [20. veljače 2018.]
[34] Toad World. Dostupno na: https://www.toadworld.com/platforms/sql-server/w/wiki/9453.very-large-databases-vldbs [10. veljače 2018.]
[35] Oracle. Dostupno na; https://docs.oracle.com/cd/E11882_01/server.112/e25523/intro.htm [10. veljače 2018.]
[36] Digital Trends. Dostupno na: https://www.digitaltrends.com/computing/facebook-server-hardware/ [13. veljače 2018.]
[37] Tech target. Dostupno na: http://whatis.techtarget.com/definition/helium-hard-drive [13. veljače 2018.]
[38] Cisco. Dostupno na: https://blogs.cisco.com/datacenter/is-facebooks-new-storage-platform-performance-hungry [13. veljače 2018.]

[39] Tom's IT PRO. Dostupno na: http://www.tomsitpro.com/articles/hgst-helium-hdd-helioseal,1-2522.html [13. veljače 2018.]
[40] Aleksey Charapko. Dostupno na: http://charap.co/gorilla-facebooks-cache-for-monitoring-data/ [13. veljače 2018.]
[41] Storage Switzerland. Dostupno na: https://storageswiss.com/2016/02/02/why-low-latency-matters/ [13. veljače 2018.]
[42] The morning paper. Dostupno na: https://blog.acolyer.org/2016/05/03/gorilla-a-fast-scalable-in-memory-time-series-database/ [14. veljače 2018.]
[43] Seagate blog. Dostupno na: https://blog.seagate.com/intelligent/facebooks-new-storage-platform-design-enables-capacity-to-scale-exponentially-easily-and-affordably/ [14. veljače 2018.]
[44] Facebook vldb 2015. Dostupno na: http://slideplayer.com/slide/10784539/, slide 11 [14. veljače 2018.]
[45] Facebook vldb 2015. Dostupno na: http://slideplayer.com/slide/10784539/ [14. veljače 2018.]
[46] Architecht. Dostupno na: https://architecht.io/facebook-open-sourced-its-time-series-database-for-systems-monitoring-aedf3c603f9f [14. veljače 2018.]
[47] Android developer. Dostupno na: https://support.google.com/googleplay/android-developer/answer/3131213?hl=en- [20. veljače 2018.]
[48] Centercode. Dostupno na: https://www.centercode.com/blog/2011/01/alpha-vs-beta-testing/ [20. veljače 2018.]
[49] Firebase. Dostupno na: https://firebase.google.com/docs/test-lab/ [20. veljače 2018.]
[50] Firebase. Dostupno na: https://firebase.google.com/docs/crash/ [20. veljače 2018.]
[51] SitePoint. Dostupno na: https://www.sitepoint.com/5-mobile-app-testing-tools/ [20. veljače 2018.]
[52] TestFairy. Dostupno na: https://testfairy.com/ [20. veljače 2018.]
[53] UserTesting. Dostupno na: https://www.usertesting.com/ [20. veljače 2018.]

Algebra
visoka škola za
primijenjeno računarstvo

NASLOV DIPLOMSKOG RADA

Pristupnik: Mirna Novak, JMBAG

Mentor: Prof. dipl. ing. Aleksander Radovan