Mobilna Android aplikacija za preporučivanje filmskih sadržaja na temelju analize slika lica

Šutalo, Domagoj

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni preddiplomski studij

ANDROID MOBILNA APLIKACIJA ZA PREPORUČIVANJE FILMSKIH SADRŽAJA NA TEMELJU ANALIZE SLIKE LICA

Završni rad

Domagoj Šutalo

Osijek, 2020.

SADRŽAJ

1. UVOD 4
1.1. Zadatak završnog rada 5
2. PREGLED TRENUTNOG STANJA U PODRUČJU FILMSKIH SADRŽAJA I PRIKAZ TIH SADRŽAJA NA OSNOVI EMOCIJA 6
2.1 POSTOJEĆE SLIČNE APLIKACIJE 6
2.1.1. IMDb 6
2.1.2. Flixster 7
2.1.3. Movie Database 8
2.2. Sustavi preporuka 10
2.2.1 Filtriranje na temelju sadržaja 11
2.2.2. Suradničko filtriranje 12
2.3. Postupci analize slike 13
2.3.1. Neuronske mreže 13
2.3.2 Stablo odluke 15
3. IDEJNO RJEŠENJE I MODEL MOBILNE APLIKACIJE 16
3.1. Dijagram rada aplikacije 16
3.2. Arhitektura baze podataka 18
4. PROGRAMSKO RJEŠENJE MOBILNE APLIKACIJE 20
4.1. Android okolina 20
4.1.1. Operacijski sustav Android 20
4.1.2. Povijest Androida 20
4.2. Android Studio 20
4.3. Programski jezik Java 21
4.4. XML 22
4.5. TMDb API 24
4.6. Firebase 26
4.7. Firebase MLKit 26
4.8. Najbitniji dijelovi koda 28
4.8.1. Postavljanje korisnika 28
4.8.2. Baza podataka 30

4.8.3. Korištenje API poziva u aplikaciji 32
4.8.4. Analiza lica korištenjem MLKit-a 35
5. NAČIN RADA I ISPITIVANJE APLIKACIJE 38
5.1. Korištenje mobilne aplikacije 38
5.2. Ispitivanje aplikacije 43
5.2.1. Prvo ispitivanje 43
5.2.2. Drugo ispitivanje 45
5.2.3. Treće ispitivanje 47
6. ZAKLJUČAK 49
LITERATURA 50
ŽIVOTOPIS 52
SAŽETAK 53
ABSTRACT 54
PRILOZI 55

1. UVOD

Prema [1], više od 183 milijuna ljudi danas je pretplaćeno na Netflix, najpoznatiju internet uslugu za gledanje filmova i serija, s tim da je to samo broj osoba koji redovito plaćaju ovakvu uslugu znajući da još više ljudi koristi piratske stranice gledajući filmove besplatno. Sve ovakve stranice pružaju ogroman broj filmova i serija, čak toliko da nekada može postojati "prevelik izbor". Upravo zbog tog problema, osmišljena je mobilna aplikacija "MovieEmotion" koja korisnicima omogućuje pretragu filmova prema analizi lica kako bi što manje vremena proveli tražeći film, a više gledajući ga. MovieEmotion također omogućuje izlistanje popularnih filmova u tom trenutku, kao i njihovo pretraživanje po imenu ,te označavanje filmova koje bi korisnik htio gledati kasnije spremajući ih na listu favorita.

Cilj ovog završnog rada je istražiti već postojeća rješenja, usporediti ih, te sukladno s njima napraviti vlastito rješenje mobilne aplikacije za pretragu i preporuku filmova prema analizi slike lica. Aplikacija mora omogućiti registriranje i prijavu korisnika, a nakon toga mogućnost pretrage filmova, njihovog spremanja kao i preporuku filmova s obzirom na tadašnje raspoloženje prema analizi slike lica.

U drugom poglavlju prikazane su slične aplikacije i opisano je trenutno stanje u području sustava preporuka, kao i postupci analize slike. Treće poglavlje prikazuje model mobilne aplikacije, funkcionalne zahtjeve na aplikaciju, te arhitekturu baze podataka, dok su u četvrtom poglavlju opisane programske tehnologija korištene za izradu aplikacije, kao i prikaz bitnih dijelova programskog koda. Peto poglavlje prikazuje korištenje aplikacije i njeno ispitivanje.

1.1. Zadatak završnog rada

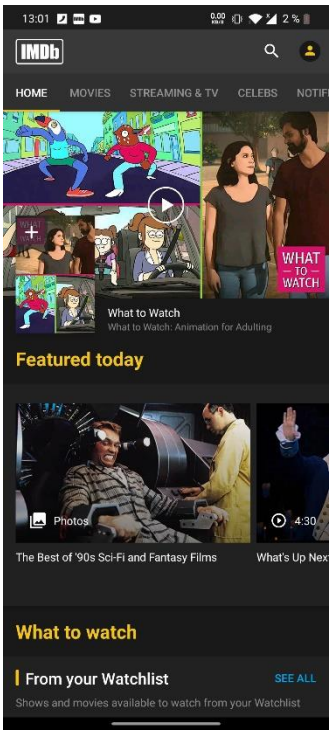
U završnom radu treba opisati mogućnosti primjene mobilnih aplikacija u sustavima preporuka, te analizirati zahtjeve na sustav za preporučivanje filmskih sadržaja. Uz to, treba predložiti model i arhitekturu mobilne Android aplikacije koja implementira sustav preporučivanja filmskih sadržaja na temelju odgovarajućih postupaka analize slika lica. Nadalje, treba izabrati i prilagoditi alat i biblioteku za prepoznavanje emocija iz slika lica i definirati kriterije preporučivanja sadržaja. Također, treba opisati programske tehnologije i okolinu za razvoj mobilne aplikacije, a u praktičnom dijelu rada programski ostvariti opisanu mobilnu aplikaciju s bazom podataka, te je ispitati i analizirati na dovoljnom skupu ulaznih podataka.

2. PREGLED TRENUTNOG STANJA U PODRUČJU FILMSKIH SADRŽAJA I PRIKAZ TIH SADRŽAJA NA OSNOVI EMOCIJA

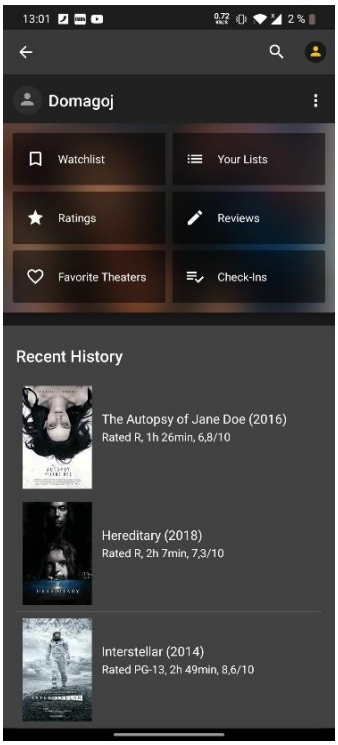
2.1 POSTOJEĆE SLIČNE APLIKACIJE

2.1.1. IMDb

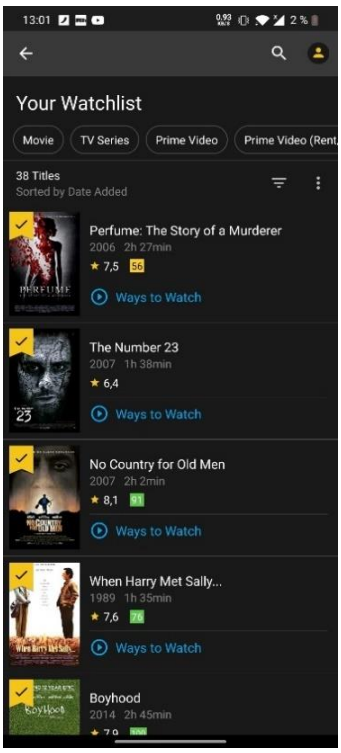
IMDb [2], odnosno Internet Movie Database predstavlja jednu od najpoznatijih baza podataka filmova i televizijskih serija što može potvrditi i podatak da ovu aplikaciju je preuzelo više od 100 milijuna ljudi te ima ocjenu 4.5/5 na Google Play Storeu. Baza podataka svoj rad je krenula prije točno trideset godina (1990. godine) te njenu stranicu mjesečno posjećuje oko trideset milijuna ljudi. Korištenje IMDb aplikacije je besplatno, no isto tako postoji plaćena verzija IMDbPro koja omogućuje pronalazak detaljnih informacija o filmovima, serijama i glumcima, pruža alate za upravljanje IMDb profilom itd. Glavne mogućnosti besplatne verzije aplikacije predstavljaju dodavanje filmova i serija na listu favorita, pisanje odraza na film, prijedlog sličnih filmova, dodavanje ocjene filmu i brojne druge. Dodavanjem ocjena filmovima, aplikacija tada koristi te informacije te stvara velike liste od 250 najbolje i 100 najgore ocjenjenih filmova. Pregled IMDb aplikacije prikazan je na slikama 2.1, 2.2, 2.3.



Slika 2.1. Prikaz početne stranice



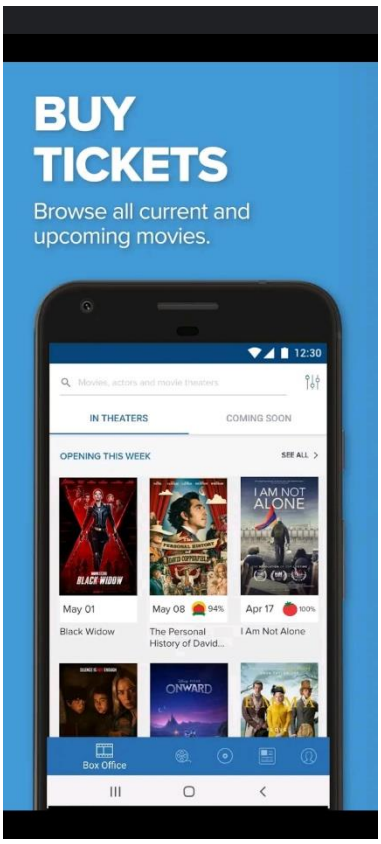
Slika 2.2. Prikaz korisničkog sučelja



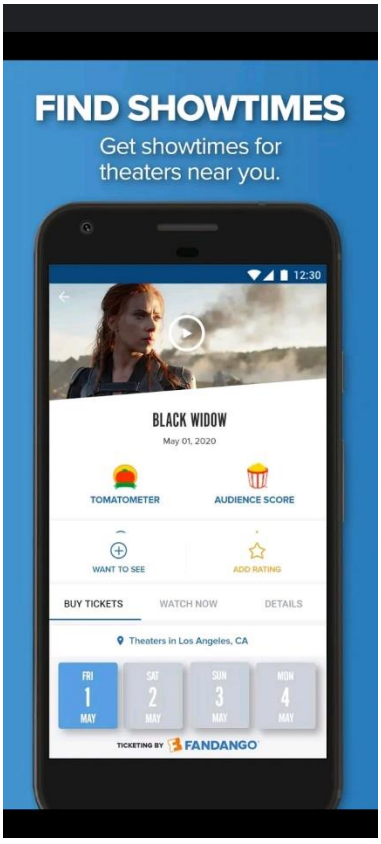
Slika 2.3. Prikaz liste gledanja filmova

2.1.2. Flixster

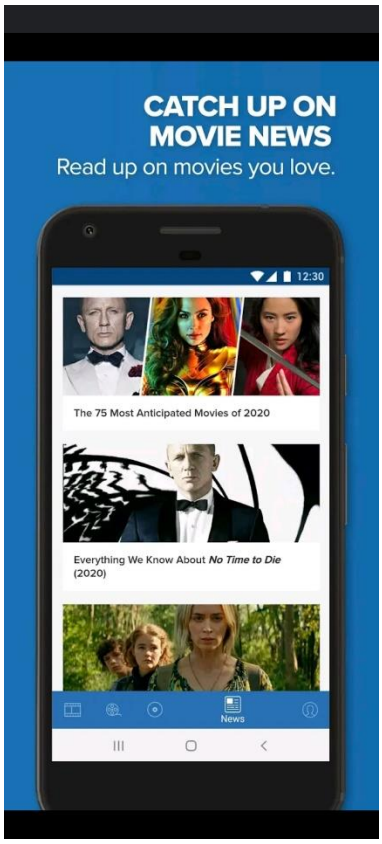
Flixster predstavlja aplikaciju koja se temelji na bazi podataka o filmovima na internetu Rotten Tomatoes [3]. Baza podataka je posvećena recenzijama, informacijama i vijestima o filmovima te sadrži arhivu filmskih kritika. Ime ove baze podataka ima dublje značenje pošto se radi o parafraziranju klišeja u kojemu gledatelji "bacaju" rajčice na loše izvedbe kao u starim danima dok su se filmovi izvodili kao predstava pred gledateljima. Na osnovu toga filmovi sa preko 60% pozitivnih recenzija predstavljaju takozvanu "svježu rajčicu" dok filmovi ispod 60% predstavljaju "lošu rajčicu". Flixster omogućuje svojim korisnicima kupovinu ulaznica za omiljene filmove, pretraživanje filmskih sadržaja svih vrsta, kao i najnovije vijesti o nadolazećim filmovima. Nalazi se u vlasništvu Warner Bros. poduzeća i još uvijek nije dostupna u Hrvatskoj. Izgled aplikacije prikazan je na sljedećim slikama (slike 2.4, 2.5, 2.6).



Slika 2.4. Kupovina karata



Slika 2.5. Pretraživanje filmova

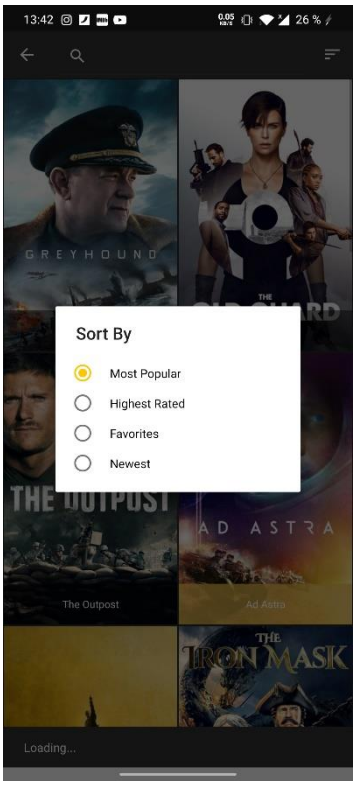


Slika 2.6. Pregled najnovijih vijesti

2.1.3. Movie Database

Movie Database predstavlja aplikaciju koja povlači informacije o filmskim sadržajima sa online baze podataka The Movie Database ili skraćeno TMDb. Ona datira iz 2008. godine i predstavlja bazu podataka koju je izgradila zajednica developera. Sa više od 200 000 developera i tvrtki koje koriste ovu platformu, TMDb je postala jedna od vodećih platformi za izvor informacija od filmovima i serijama. Dok Flixster nije dostupan u Hrvatskoj i lako moguće dosta ostalih europskih i ostalih zemalja, TMBb podržava više od 40 jezika na svojoj platformi i dostupna je širom svijeta. Ulaskom u aplikaciju na početnom ekranu možemo vidjeti filmove koji su sortirani prema popularnosti(slika 2.7), dok aplikacija omogućava sortiranje po najvećoj ocijeni i najnovijim filmovima(slika 2.8). Isto tako, Movie Database omogućuje pregled informacija kao i dodavanje filmova na listu favorita pomoću gumba u obliku srca (slika 2.9) kao i pretraživanje filmova po naslovu(slika 2.10).

Slika 2.7. prikaz filoma prema popularnosti



Slika 2.8. Sortiranje filmova

Slika 2.9. Infromacije o filmu

Slika 2.10. Pretraživanje filmova prema nazivu

2.2. Sustavi preporuka

S rastom i razvitkom interneta, a sukladno njemu i drušvenih mreža poput Facebooka, ljudi sve više počinju koristiti društvene mreže, kako iz zabave, tako isto i iz poslovnih razloga. Isto tako je danas moguće preko interneta kupiti nešto iz daleke Kine ili Japana sa samo jednim klikom miša. Sve te društvene mreže (Facebook, Instagram, Youtube itd.), kao i internet trgovine implementiraju sustav preporuka kojim se ostvaruje rast same društvene mreže odnosno trgovine. Facebook, kao i Instagram ima svoj sustav za prijedlog ljudi koje bi korisnici možda mogli znati, Ebay i Amazon ukazuju na to što su još kupili drugi korisnici koji su kupili istu stvar kao i trenutni, kao i slične predmete sa možda manjom cijenom, dok aplikacije za filmove preporučavaju filmove slične onome o kojemu korisnik upravo čita kratki opis.

Prema [4], problem preporučivanja nastao je sredinom 1990-ih godina, ali se danas još razvija i proširuje na područje primjene, te je 2006. godine populariziran zbog "Netflix Prize". Ovaj sustav predviđa koji predmet bi mogao zanimati određenog korisnika, a ta predviđanja se baziraju na podacima o:

- Korisniku kojem se preporučavaju predmeti (starost, mjesto prebivanja, posao...)
- Povratne informacije korisnika o nekim drugim predmetima (ocjene, broj pregleda...)
- Podaci vezani uz korisnike koji imaju sličan ukus
- Predmetima koji su slični uz predmet koji se korisniku sviđa

Formalna definicija problema:

U predstavlja konačan skup kojeg nazivamo skup korisnika (engl. *users*) , ***I*** konačan skup kojeg nazivamo skup predmeta (engl. *items*), a ***T*** totalno uređen skup (npr. [0, 1], {0, 1, . . . , 10}, R +) i na kraju $r : U \times I \rightarrow T$ predstavlja funkciju koja pridružuje vrijednost korisnosti nekog predmeta određenom korisniku.

Algoritmi za rješavanje ovog problema:

- **kNN** (engl. *k Nearest Neighbours*) - osnovni algoritam za rješavanje, većina ostalih ga proširuju
- **Suradničko filtriranje** (engl. *Collaborative Filtering*) - najrazvijeniji algoritam za rješavanje ovog problema. Radi na način da se traže predmeti koje su visoko ocjenili ljudi koji imaju sličan ukus kao tražena osoba.
- **Filtriranje na temelju sadržaja** (engl. *Content-based recommendation*) predstavlja algoritam za prijedlog predmeta koji je sličan onima koje je korisnik prethodno dobro ocjenio ili onom predmetu kojeg korisnik trenutno pregledava

2.2.1 Filtriranje na temelju sadržaja

Prema [5], Content-based Filtering ili filtriranje na temelju sadržaja temelji se na opisu predmeta i korisnikovim preferencijama. Ova metoda je najprikladnija za situacije u kojima postoje poznati podaci o predmetu (npr. ime, mjesto, opis itd.), ali ne postoje podaci o korisniku. Preporučitelji temeljeni na sadržaju tretiraju preporuke kao problem klasifikacije specifičan za korisnika i uče klasifikator na korisnikove "simpatije" i "antipatije" koje su temeljene na značajkama predmeta. U ovakvom sustavu koriste se ključne riječi za opisivanje predmeta, a korisnički profil se izrađuje kako bi se naznačina vrsta predmeta koji se ovom korisniku sviđa. Drugačije rečeno, ovi algoritmi pokušavaju preporučiti predmete slične onima koje su se korisniku svidjele u prošlosti ili ih ispituje u sadašnjosti.

Ovakav algoritam pokazao se dobrim za probleme vezane uz rudarenje tekstom (engl. *text mining)* uz TF-IDF funkciju za traženje sličnosti (preporučivanje blogova, znanstvenih radova i slično), no isto tako ovaj algoritam ima mana kao što su:

- Teško povezivanje s multimedijskim sadržajem
- Ako je korisnik dobro ocijenio neki predmet, moguće da će mu sustav preporučiti neki identičan, ali sa drugog izvora
- Korisniku nikada neće biti preporučeni predmeti koji bi mu se možda svidjeli, ali ih nikada nije ocjenio

- Ne prepoznaje koji članak je kvalitetniji među onima koji imaju iste ključne riječi

2.2.2. Suradničko filtriranje

Prema [5], jedan od pristupa dizajnu sustava preporuka koji se široko koristi predstavlja collaborative filtering ili suradničko filtriranje. Filtriranje u suradnji temelji se na pretpostavci da će se ljudi koji su se složili u prošlosti složiti u budućnosti i da će im se svidjeti slične vrste predmeta kao što su se sviđale u prošlosti.

Sustav generira preporuke koristeći samo informacije o profilima ocjenjivanja za različite korisnike ili predmete. Pronalaženjem vršnjačkih korisnika / predmeta s poviješću ocjena sličnom trenutnom korisniku ili predmetu, oni generiraju preporuke koristeći ovo susjedstvo. Ključna prednost zajedničkog pristupa filtriranju je u tome što se ne oslanja na sadržaj koji se može analizirati strojem te je stoga sposoban precizno preporučiti složene stavke poput filmova, a da ne zahtijeva "razumijevanje" same stavke. Mnogi algoritmi korišteni su za mjerenje sličnosti korisnika ili sličnosti predmeta u sustavima koji preporučuju.

Kao i kod filtriranja na temelju sadržaja i ovaj algoritam ima nekoliko nedostataka:

- Hladni start - Za novog korisnika ili predmet nema dovoljno podataka za davanje točnih preporuka
- Prilagodljivost - U mnogim okruženjima u kojima ti sustavi daju preporuke postoje milijuni korisnika i proizvoda, stoga je za izračunavanje preporuka često potrebna velika količina računske snage
- Škrtost - Broj predmeta prodanih na glavnim web mjestima e-trgovine izuzetno je velik. Najaktivniji korisnici ocijenit će samo mali podskup ukupne baze podataka. Dakle, čak i najpopularniji predmeti imaju vrlo malo ocjena.

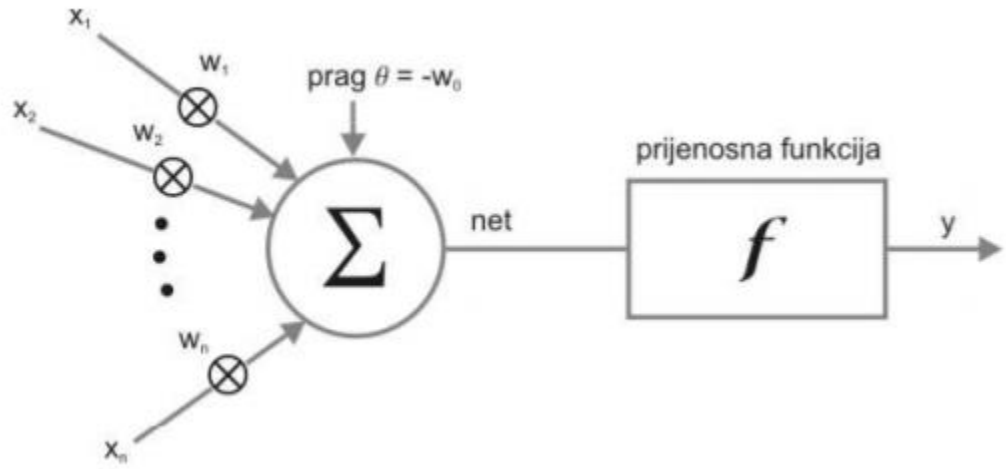
2.3. Postupci analize slike

Za analizu slike i otkrivanje tko ili što se na njoj nalazi korištena je klasifikaciju slika. Ona predstavlja postupak kojim se slika rastavlja na polja kojima je potrebno odrediti klasu. Prema [6], klasifikacija slika primarna je domena u kojoj duboke neuronske mreže igraju najvažniju ulogu u analizi slika, ali isto tako predstavlja složen postupak koji se oslanja na različite komponente. Jedne od najboljih i najvažnijih metoda klasifikacije slika predstavljaju neuronske mreže i stablo odluke.

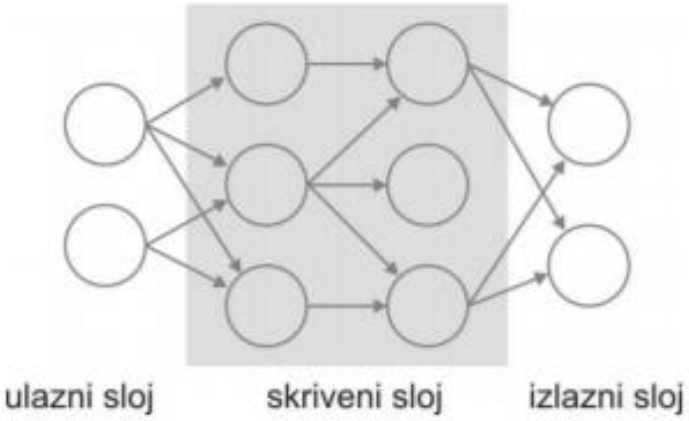
2.3.1. Neuronske mreže

Umjetne neuronske mreže tvore grupu povezanih čvorova. Pošto se umjetna neuronska mreža temelji se na mrežama neurona u ljudskom mozgu, odnosno živčanom sustavu, svaki čvor trebao bi predstavljati jedan neuron u ljudskome mozgu. Neuron predstavlja osnovnu gradivnu jedinicu živčanoga sustava i jednom kada prestanu sa radom više se ne mogu povratiti. Svaki neuron ima ulazni i izlazni signal, gdje svaki njegov izlazni signal može biti ulazni drugome neuronu. Slično je i kod umjetne neuronske mreže koja može imati više ulaznih signala ali samo jedan izlazni signal. Umjetne neuronske mreže na ovaj način pokušavaju kopirati ljudski mozak kako bi se mogli ostvariti postupci učenja [7].

Prikaz umjetnog neurona možemo vidjeti na slici 2.11 gdje su ulazni signali predstavljeni s $x_1,x_2,...,x_n$, a izlazni signal sa y. Povezivanjem tih umjetnih neurona dolazimo do neuronske mreže, gdje se najjednostavnija mreža sastoji od barem tri sloja: ulazni sloj, skriveni sloj i izlazni sloj (slika 2.12).



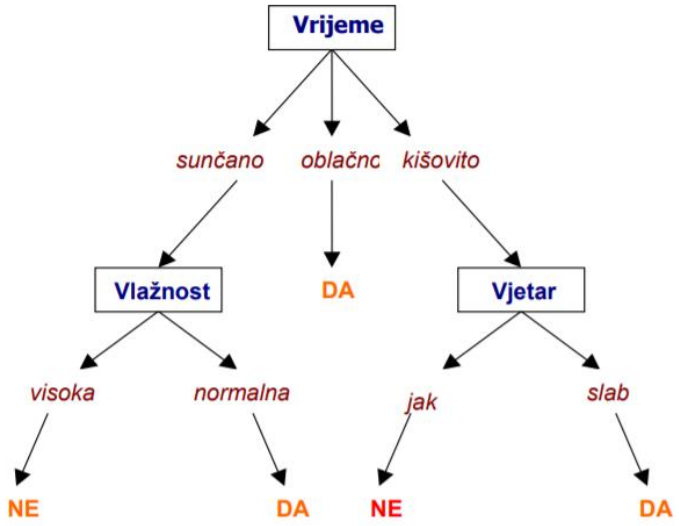
Slika 2.11. Prikaz umjetnog neurona [7]



Slika 2.12. Prikaz umjetne neuronske mreže [7]

2.3.2 Stablo odluke

Prema [8], stablo odluke ili odlučivanja predstavlja strukturu nalik stablu ili dijagramu toka, koji će biti korišten u sljedećem poglavlju. Sastoji se od grana, čvorova i lista koji je ujedno i zadnji čvor. Svaki čvor predstavlja oblik testa na atributu, dok svaka grana dijagrama predstavlja rezultat testa, a svaki list predstavlja oznake klasa. Stablo koje na svaki upit ima samo dvije odluke još nazivamo i binarnim stablom. Binarno stablo struktuira se na način da se najvažnija pitanja, odnosno testovi naprave na početku i svaki manje važan test se stavlja sljedeći. Na primjeru binarnoga stabla, dvije odluke koje predstavljaju odgovor na određeno pitanje su često suprotnih vrijednosti. Za razliku od drugih algoritama, stablima odluke treba manje truda za pripremanje podataka prije obrade, isto tako stablo odluke ne zahtijeva normalizaciju podataka kao niti njihovo skaliranje. Modeli stabla odluke su veoma intuitivni i lako ih je za objasniti. Zbog svega ovoga, ono predstavlja jako popularan algoritam za korištenje u strojnome učenju gdje se podaci pretvaraju u stabla. Isto tako neki od problema stabla odluke mogu biti da su previše istrenirana (engl. *overtrained),* odnosno uvelike trenirano stablo koje se previše prilagodilo skupu podataka. Male promjene podataka mogu proizvesti velike promjene u strukturi što dovodi do nestabilnosti. Također, treniranje ovakvih struktura je dosta skupo zbog kompleksnosti i količini utrošenog vremena.



Slika 2.13. Prikaz lako razumljivog stabla odluke [9]

3. IDEJNO RJEŠENJE I MODEL MOBILNE APLIKACIJE

3.1. Dijagram rada aplikacije

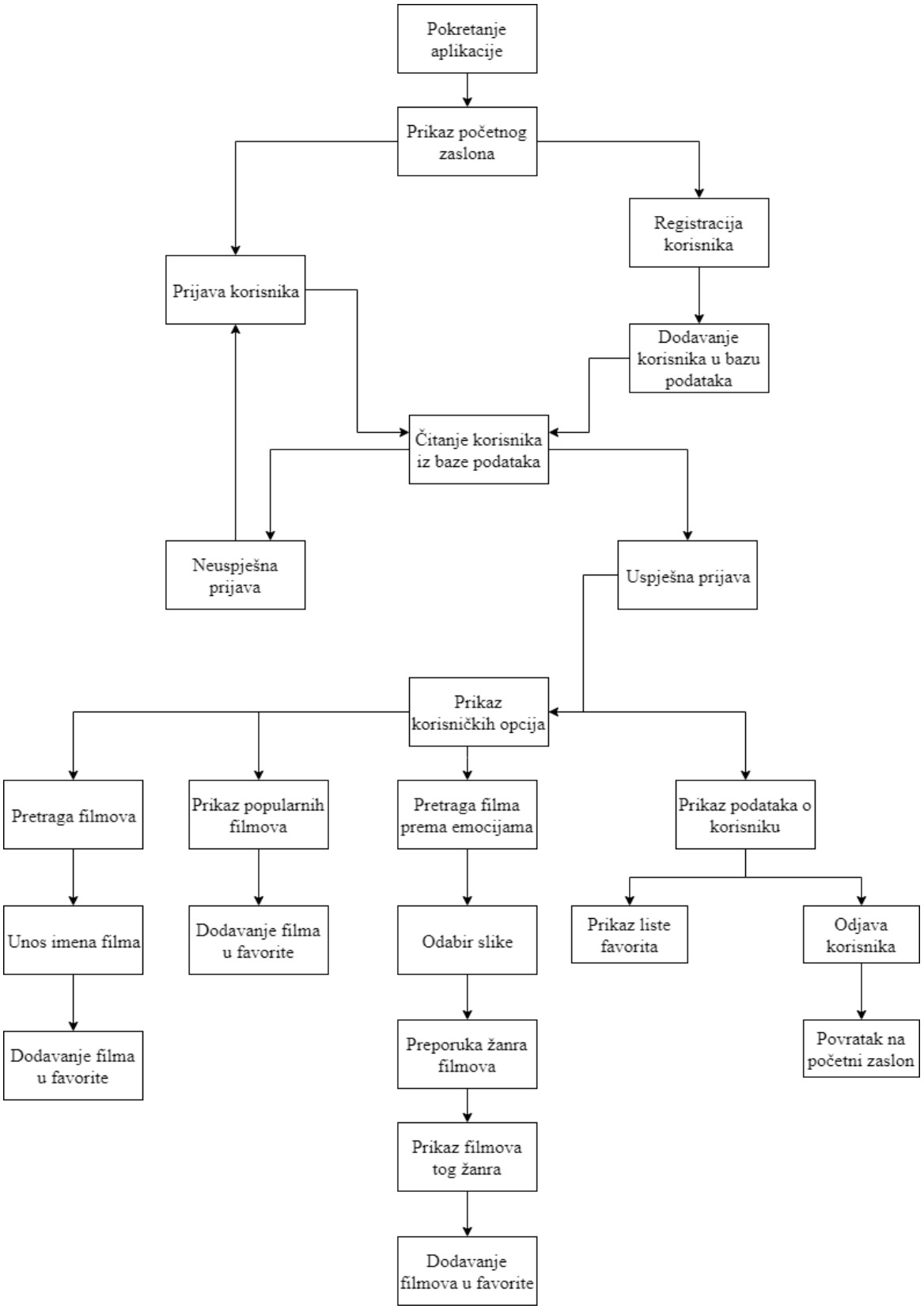
Radi lakšeg ostvarenja aplikacije i prikaza njenih funkcionalnosti, kreiran je dijagram rada prikazan na slici 3.1. Kao što je i vidljivo na dijagramu prilikom otvaranja aplikacije prikazuje se početni zaslon koji od korisnika traži njegovu prijavu odnosno registraciju. Nakon što se korisnik uspješno prijavi u sustav, dolazi do zaslona na kojemu ima više mogućnosti.

Počevši od vrha korisnik ima mogućnost pronalaska filma prema imenu kao i dodavanje filma na listu favorita nakon uspješne pretrage. Korisniku se također na ovome zaslonu prikazuje lista filmova koji su popularni u ovome trenutku, te mogućnost i njihovog dodavanja na listu favorita. Ako korisnik ne želi tražiti filmove ima također opciju predlaganja filma prema analizi lica, gdje korisnik odabire fotografiju te prema njenoj analizi dobiva listu preporučenih filmova prema žanru. Na kraju, korisnik ima mogućnost pregleda informacija o samome sebi, kao i pregled filmova koje je spremio na listu favorita, a na kraju i odjavu iz sustava.

Kao što je viđeno iz priloženoga, funkcionalni zahtjevi na aplikaciju su sljedeći:

- Pri ulasku u aplikaciju, korisnik mora obaviti registraciju, odnosno prijavu
- Pregled popularnih filmova
- Pretraga filmova po imenu
- Preporuka filmova prema analizi lica
- Dodavanje filmova na listu favorita
- Prikaz podataka o korisniku
- Prikaz liste favorita
- Odjava korisnika

Implementacija ovih funkcionalnosti u sustav i prikaz njihovog algoritma biti će detaljnije opisano u sljedećem poglavlju pod nazivom "Najbitni dijelovi koda".

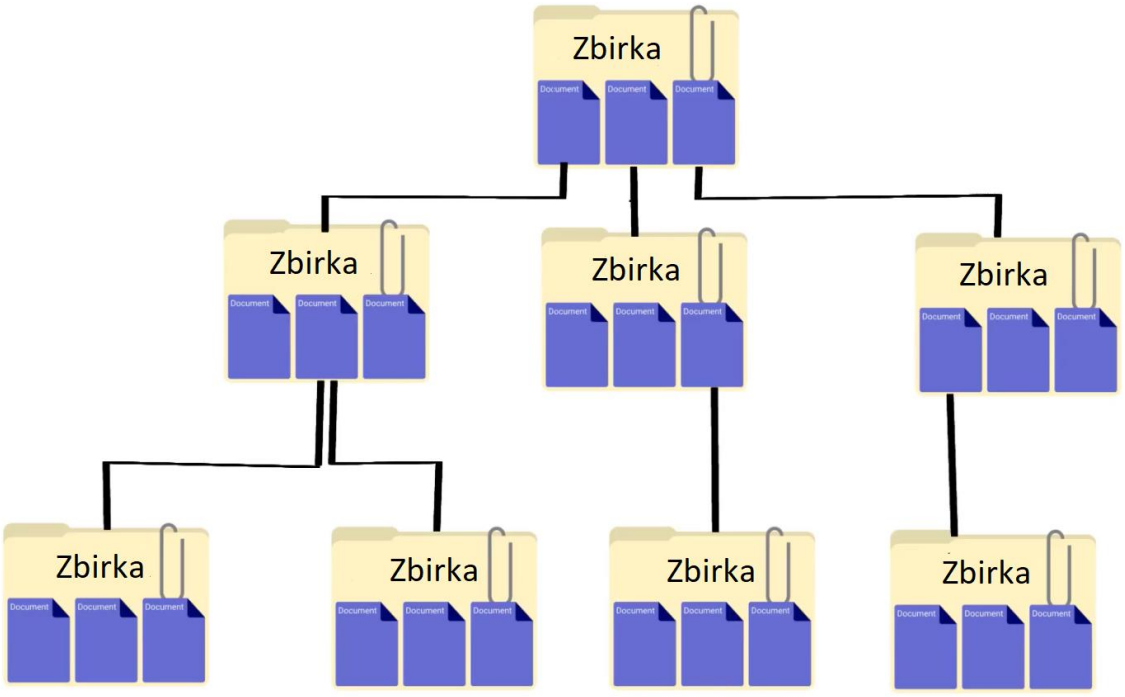


Slika 3.1. Dijagram rada aplikacije

3.2. Arhitektura baze podataka

Kako bi spremali korisnike prijavljene u sustav, kao i filmove koje je korisnik stavio u favorite, bilo je potrebno koristiti nekakvu bazu podataka, kako nam se svi podaci ne bi izgubili prilikom izlaska iz aplikacije. U ovome slučaju korištena je Firebase Firestore baza podataka.

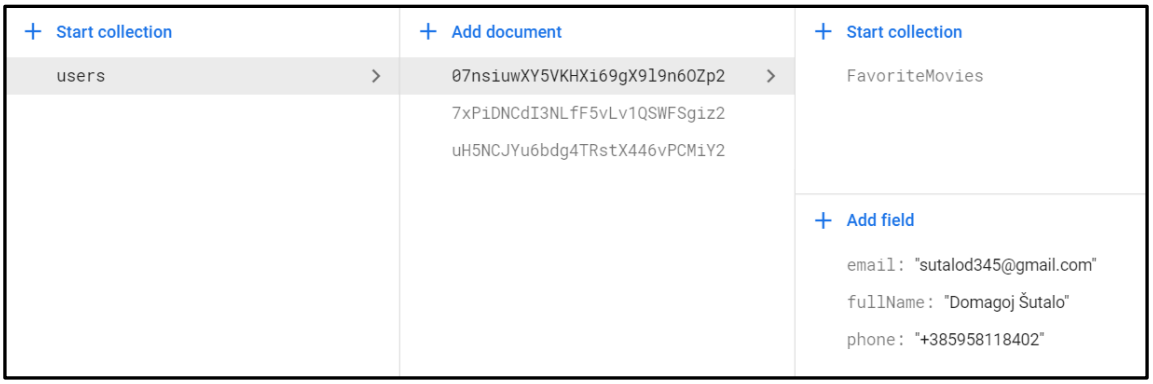
Kako bi lakše razumjeli Firestore bazu podataka potrebno je navesti da ona ne izgleda kao standardna SQL (*Standard Query Language)* gdje se podaci spremaju u tablice, već je ona tipa NoSQL gdje se podaci spremaju u dokumente i zbirke. Kao što je prikazano na slici 3.2 zbirka (engl. *collection*) može sadržavati više dokumenata, ali ne može sadržavati novu zbirku, kao što dokumenti ne mogu sadržavati druge dokumente u sebi. Novu zbirku mogu sadržavati samo dokumenti unutar zbirke.



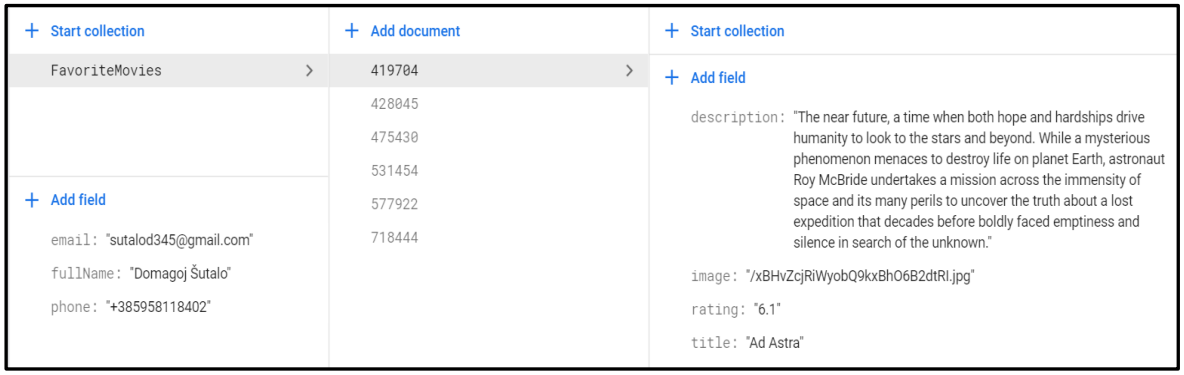
Slika 3.2. Prikaz hijerarhije dokumenata i zbirki u Firebase Firestore

U ovome primjeru korištene su dvije zbirke:

- *Users* - zbirka koja sadrže dokumente koji predstavljaju korisnike koji su registrirani u sustav, gdje su korisnici opisani pomoću *email-a, punog imena i broja mobitela*(slika 3.3)
- *Favorite movies* - zbirka koja se nalazi unutar svakog pojedinog korisnika (dokumenta) i sadrži filmove dodane na listu favorita (dokument) koji su opisani pomoću *imena, opisa, ocjene i slike* (slika 3.4)



Slika 3.3. Prikaz User zbirke



Slika 3.4. Prikaz Favorite movies zbirke

4. PROGRAMSKO RJEŠENJE MOBILNE APLIKACIJE

4.1. Android okolina

Budući da je aplikacija završnoga rada namijenjena korištenju na mobilnim uređajima koji podržavaju Android operacijski sustav, u ovome poglavlju objašnjena je Android okolina kao i razvojne okoline, korištene tehnologije i jezici.

4.1.1. Operacijski sustav Android

Prema [10], Android predstavlja operacijski sustav(OS u nastavku) temeljen na Linux jezgri koji je prvenstveno namijenjen za mobilne uređaje i tablete. Osim pametnih telefona i tableta, Android OS je danas korišten također u autima (Android Auto) kao i u pametnim televizorima (Android TV) i satovima (Wear OS). Najnovija inačina Android OS-a predstavlja Android 10, ali će ga uskoro zamijeniti Android 11. Glavni konkurent Googlu i njegovom Android OS-u predstavlja Apple sa svojim najpoznatijim uređajem, iPhone-om, te operacijskim sustavom iOS.

4.1.2. Povijest Androida

Prema [11] povijest Androida započela je u 2003. godini, kada su Andy Rubin, Rick Miners, Chris White i Nick Sears osnovali Android Inc. 2005. godine Google preuzima tvrtku i donosi se odluka kako će se Linux koristiti kao osnova za Android operacijski sustav(u nastavku OS). To je značilo da bi se OS mogao ponuditi nezavisnim proizvođačima mobitela besplatno, te iz tog razloga danas Android jedan od najrasprostranjenijih operacijskih sustava. Prva verzija Android OS-a objavljena je 5. studenog 2007. godine, a prvi Android pametni telefon najavljen je u rujnu 2008. godine.

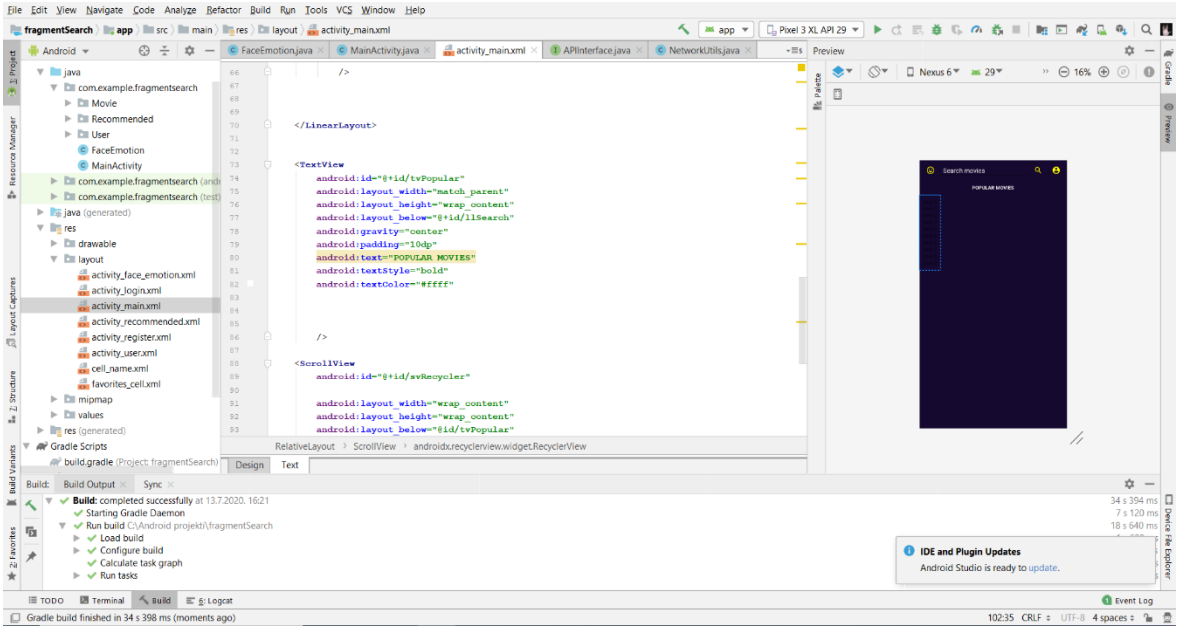
4.2. Android Studio

Prema [12], Android Studio je službeno razvojno okruženje za razvoj Android aplikacija koje se temelji na integriranom razvojnom okruženju (engl. *integrated development environment application, IDEA*) te je izgrađeno specifično za Android development. Dostupno je na više operacijskih sustava kao što su Windowsi, macOS i Linux. U 5. mjesecu 2019. godine, Kotlin je

zamijenio Javu kao Googlov preferirani jezik za razvoj Android aplikacija, ali s obzirom na to Java se još uvijek može koristiti u Android Studiju kao i C++.

Neke od glavnih značajki Android Studija su:

- Gradle sustav za izgradnju
- Refaktoriranje specifično za Android i brz ispravak grešaka
- Lint alati za zahvat performansi
- Brojni alati za ispitivanje i biblioteke
- Android Virtualni Uređaj koji je korišten za pokretanje aplikacija u Android Studiju bez potrebe instalacije na mobilni uređaj

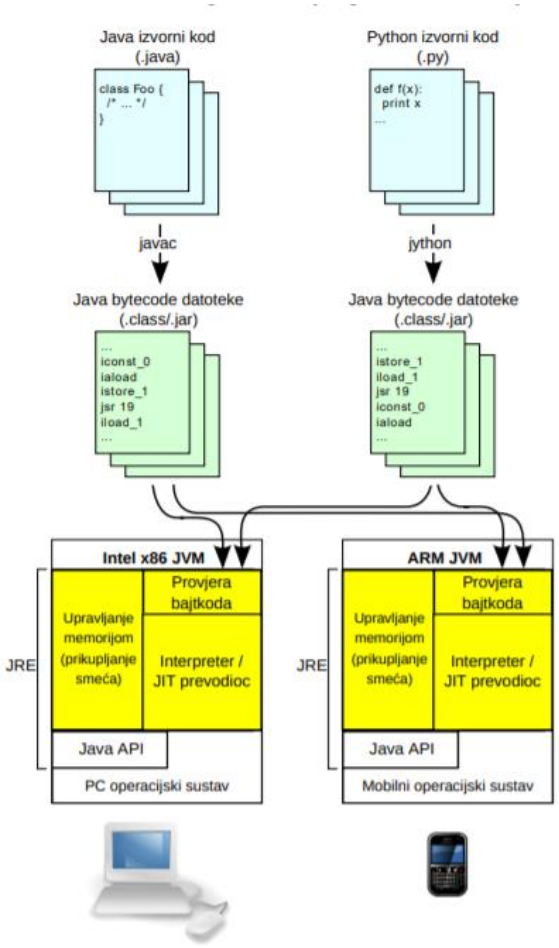


Slika 4.1. Prikaz Android studija

4.3. Programski jezik Java

Prema [131111], Java predstavlja programski jezik kojeg su razvli inženjeri u tvrtci Sun MicroSystems. Java slijedi objektno orijentiranu paradigmu i velika prednost u odnosu na većinu dotadašnjih programskih jezika bila je ideja *"napiši jednom pokreni bilo gdje"*, odnosno programi napisani u Javi su se mogli izvoditi na bilo kojem operativnom sustavu za kojega postoji Java

Virtual Machine, dok je na ostalim jezicima, kao na primjer u C-u, bilo potrebno prilagođavati programe platformi. Danas je Java jedan od najkorištenijih programskih jezika u svijetu, gdje broj korisnika dostiže broj i do 10 milijuna. Primjer postupka prevođenja i izvođenja Java programa prikazan na slici 4.3.



Slika 4.3. Postupak prevođenja i izvođenja Java programa [13]

4.4. XML

Prema [14], XML predstavlja kraticu za *Extensible Markup Language.* Riječ je o tekstualnom jeziku za označavanje podataka. XML koristi oznake koje služe za pohranu i organizaciju podataka, dok njemu sličan jezik za označavanje (engl. *markup language),* HTML (*Hyper Text Markup Language*) koji pomoću oznaka određuje kako te podatke prikazati. XML ne bi trebao zamijeniti HTML u skoroj budućnosti, ali uvodi nove mogućnosti kao što su: XML omogućuje

stvaranje vlastitih samoopisnih oznaka i on pohranjuje podatke,a ne predstavlja ih. Primjer XML koda možemo vidjeti na slici 4.4.

```
<ImageButton
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:src="@drawable/ic_mood_black_24dp"
    android:background="@android:color/transparent"
    android:padding="10dp"
    android:onClick="detectFace"

    />

<EditText
    android:id="@+id/etSearch"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_weight="5"
    android:hint="Search movies"
    android:padding="10dp"
    android:textColor="#ffff"
    android:textColorHint="#ffff"
```

Slika 4.4. Primjer XML koda za stvaranje gumba i prostora za unos texta

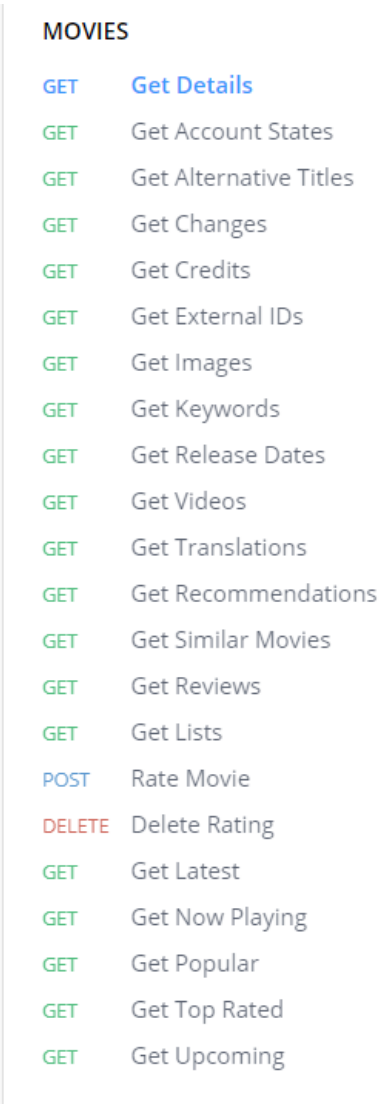
4.5. TMDb API

The Movie Database ili skraćeno TMDb predstavlja online bazu podataka filmova i TV serija, koja sa preko 570 tisuća spremljenih filmova i približno 100 tisuća serija predstavlja jednu od boljih online baza. Isto tako, osim filmova i serija, TMDb također pruža informacije o glumcima i redateljima. Zbog toga, a i zbog lakoće korištenja, korištena je u aplikaciji ovog završnog rada.

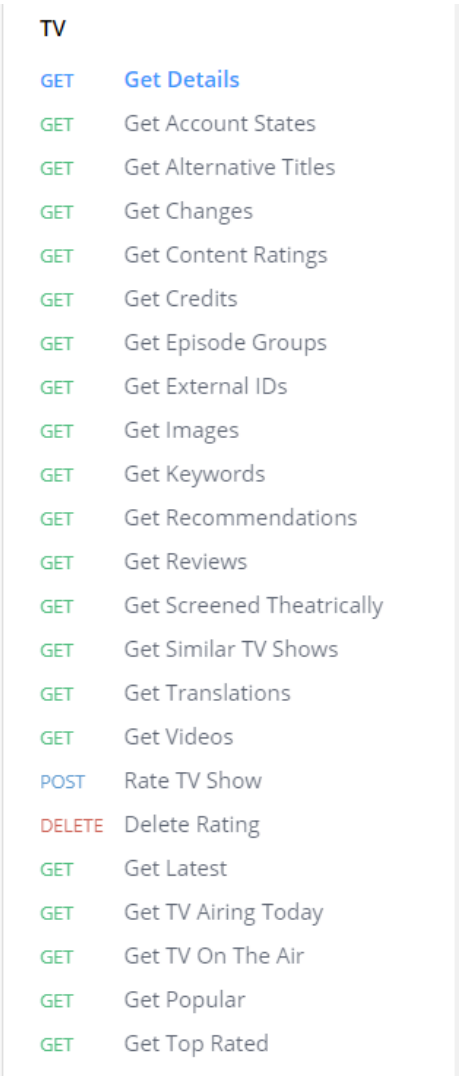
Kako bi bilo moguće koristiti TMDb API u navedenoj aplikaciji, prvo je potrebno naučiti upravljati osnovnim URL-om, koji je u ovome slučaju: *https://api.themoviedb.org/*. Pošto aplikacija zahtjeva pronalazak filmova, dohvaćanje popularnih filmova i dohvaćanje žanrova filmova, na osnovni URL potrebno je dodati nekoliko ključnih URL atributa (engl. *Queryija)* odnosno *nastavaka* kao što su:

- */search/movie* koji omogućuje pretragu filmova prema imenu
- */movie/popular* koji omogućuje dohvaćanje popularnih filmova u stvarnom vremenu

Osim ovih opcija, TMDb omogućuje brojne druge opcije kao što su pretrage glumaca, dohvaćanje sličnih filmova, dohvaćanje najbolje ocijenjenih filmova itd. Prikaz više opcija na filmovima i TV serijama prikazan je na slikama 4.5 i 4.6, a više o korištenju i manipulaciji URL-a detaljnije je opisano u potpoglavlju 4.8.3 "Korištenje API poziva u aplikaciji".



Slika 4.5. prikaz opcija nad filmovima [15]



Slika 4.6. prikaz opcija nad TV serijama [15]

4.6. Firebase

Prema [16], Firebase predstavlja skup alata koji služe za poboljšanje i razvoj aplikacije, kao i za njenu izgradnju. Neke od usluga koje Firebase pruža su: provjera autentičnosti, mjerenje korisnikove aktivnosti, ispitivanje aplikacije, a pri izradi aplikacije ovog projekta, Firebase je pomogao pri spremanju podataka (filmova) na online bazu podataka koja se još zove Firebase Firestore, a također i pri stvaranju korisnika preko Firebase Authentication kao i pri raspoznavanju raspoloženja odnosno postotka nasmiješenosti preko Firebase MLKit-a (*Machine Learning kit*).

4.7. Firebase MLKit

Firebase omogućuje brojne proizvode i usluge, a kao jedan od proizvoda je Firebase MLKit koji predstavlja kraticu za Firebase-ov Machine Learning Kit odnosno alat za strojno učenje. On zapravo predstavlja jezgru i rješenje problema pronalaska filmova prema određenoj ekspresiji lica. Prema [17], proces očitavanja izraza lica, odnosno postotka nasmiješenosti sastoji se od nekoliko koraka. Prvi od njih je taj da ovaj alat najprije mora prepoznati nalazi li se uopće na slici ljudsko lice. Kako bi alat prepoznao ljudsko lice, slici se moraju moći pridružiti određena svojstva kao što su pridruženi položaj, veličinu lica i orijentaciju. Isto tako se mogu, a i ne moraju, pridružiti nekakve značajke lica, kao što su nos i oči.

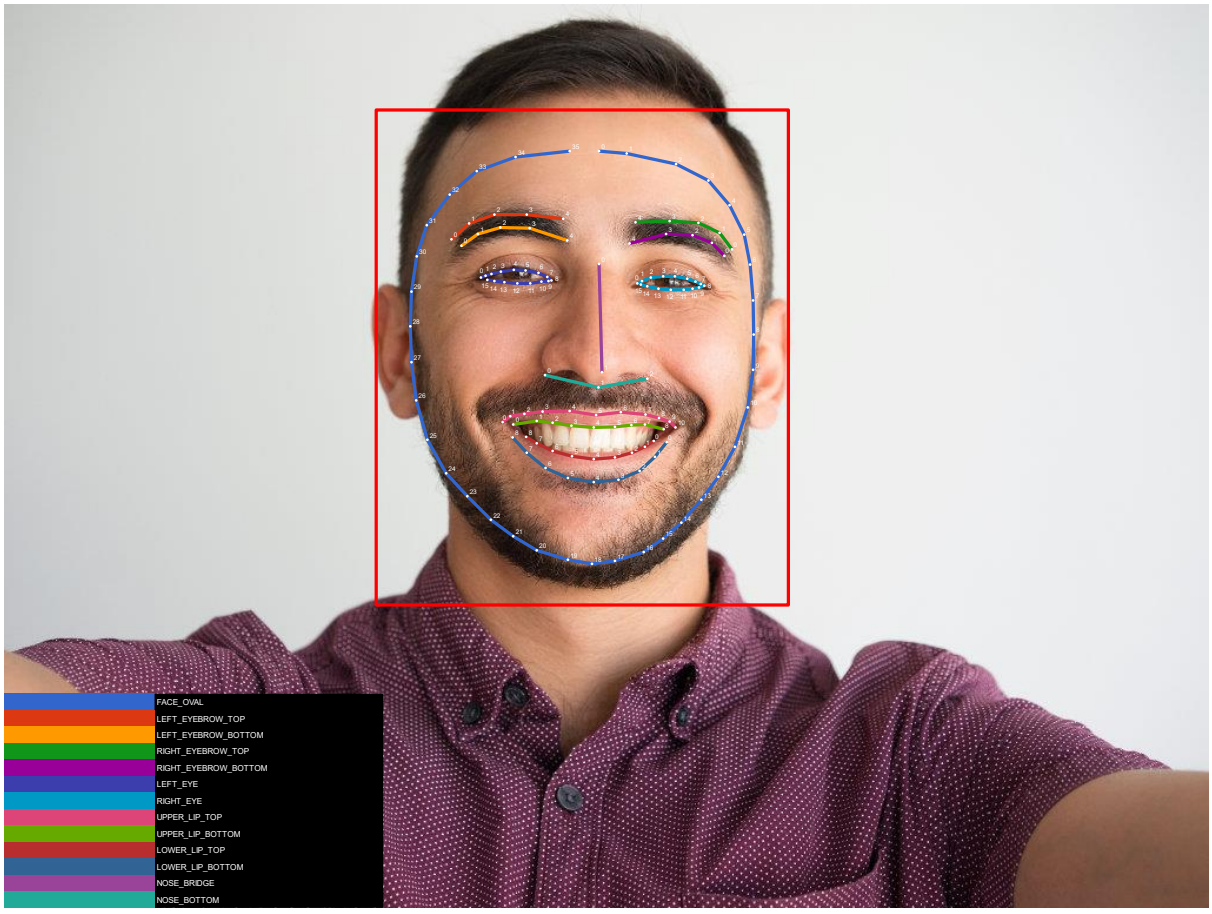
Neki od pojmova koji se koriste sa značajkom otkrivanja lica su:

- Praćenje lica, koji omogućuje prepoznavanje lica čak i na video zapisima. Tako se u videozapisu tijekom određenog vremenskog razdoblja mogu pratiti sva lica koja se pojavljuju u njemu.
- Značajka lica predstavlja točku interesa na licu, te pod nju spadaju lijevo i desno oko, kao i baza nosa.
- Kontura predstavlja skup točaka koje slijede oblik značajki lica
- Klasifikacijom se određuje nalazi li se određena karakteristika lica, kao primjer lice se može klasificirati po tome jesu li oči otvorene ili zatvorene, te smiješi li se osoba na slici ili ne.

Kontura kao što je maloprije navedeno je skup točaka koji predstavljaju oblik crta lica. Svaka crta lica koju MLKit detektira predstavljena je fiksnim brojem bodova, odnosno svaka crta lica predstavljena je određenim brojem točki, na primjer:

- Svaka obrva predstavlja 5 bodova, odnosno 5 točki
- Prepoznavanje oblika lica u sebi sadrži čak 36 točki
- Gornja usna sadrži 11 točki, dok donja usna ima malo manje, odnosno 9 točki itd.

Kada se cijela kontura lica obavi, ona na kraju predstavlja skup od 133 točke, što je vidljivo na slici 4.7.



Slika 4.7. Prikaz konture lica, odnosno skupa točki konture lica [17]

Klasifikacija određuje je li prisutna određena karakteristika lica. MLKit trenutno podržava dvije klasifikacije kao što je navedeno gore, a to su: otvorene oči i nasmiješenost. Klasifikacija je vrijednost sigurnosti, ona određuje pouzdanost da je određena karakteristika lica prisutrna, kao na primjer: ako je vrijednost nasmiješenosti 0.7 odnosno 70% ili viša, tada se sa sigurnošću može reći da se osoba smiješi. Više o korištenju ovog alata objašnjeno je u potpoglavlju 4.8.4 "Analiza lica korištenjem MLKit-a".

4.8. Najbitniji dijelovi koda

4.8.1. Postavljanje korisnika

Ako prilikom izrade aplikacije nije omogućeno stvaranje korisnika tada aplikacija nije personalizirana već generička. Aplikaciju koristi samo jedan korisnik i korisnikove podatke mora spremati na lokalnu bazu podataka, ili uopće ne spremati podatke, dok svi filmovi prethodno stavljeni na listu favorita bi bili obrisani jednom kada se izađe iz aplikacije. U rješavanju ovoga problema korišten je Firebase.

Kako bi korisnik mogao ući u aplikaciju prvo se mora registrirati. Registracija korisnika odvija se na način da se korisniku pruži mjesto za ispunu adrese e-pošte i lozinke, koju kasnije koristi pri prijavi u sustav. Prema slici 4.8, do tih informacija je moguće doći tako što se u nove varijable spremaju adresa e-pošte i lozinka pomoću funkcije *getText().toString().trim()*. Aplikacija korisniku daje na znanje da njegova zaporka mora sadržavati više od 6 znakova, te da polja adrese e-pošte i zaporke moraju biti ispunjena inače se korisnik ne može kreirati.

```
btnRegister.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        final String email = etEmail.getText().toString().trim();
        String password = etPassword.getText().toString().trim();
        final String fullName = etfullName.getText().toString();
        final String phoneNumber = etPhone.getText().toString();

        if(TextUtils.isEmpty(email)){
            etEmail.setError("Email is Required");
            return;
        }
        if(TextUtils.isEmpty(password)){
            etEmail.setError("Password is Required");
            return;
        }

        if(password.length() < 6){
            etPassword.setError("Password must be 6 characters at least");
        }
        progressBar.setVisibility(View.VISIBLE);
```

Slika 4.8. Dohvaćanje informacija o korisniku

Ako je korisnik unio valjanu adresu e-pošte, zaporku i ostala polja koja se traže od njega, tada se korisnik uspješno registrira u sustav i njegove informacije se spremanju u Firestore, a u slučaju da je nešto krivo uneseno pojavljuje se Toast poruka sa navedenom greškom kako bi korisnik to mogao ispraviti (slike 4.9 i 4.10).

```
//register user in firebase

firebaseAuth.createUserWithEmailAndPassword(email,password).addOnCompleteListener((task) → {
        if(task.isSuccessful()){
            Toast.makeText( context: Register.this, text: "User Created", Toast.LENGTH_SHORT).show();
            userID = firebaseAuth.getCurrentUser().getUid();
            DocumentReference documentReference = firebaseFirestore.collection( collectionPath: "users").document(userID);
            Map<String,Object> user = new HashMap<>();
            user.put("fullName", fullName);
            user.put("email", email);
            user.put("phone", phoneNumber);
            documentReference.set(user).addOnSuccessListener((OnSuccessListener) (aVoid) → {
                    Log.d(TAG, msg: "onSuccess: user Profile is created for" +userID);
            });

            startActivity(new Intent(getApplicationContext(), MainActivity.class));
        }else{
            Toast.makeText( context: Register.this, text: "Error" + task.getException().getMessage(), Toast.LENGTH_SHORT).show();
            progressBar.setVisibility(View.GONE);
        }
});
```

Slika 4.9. Spremanje korisnika u Firebase

Slika 4.10. Prikaz spremljenih korisnika u Firebase konzoli

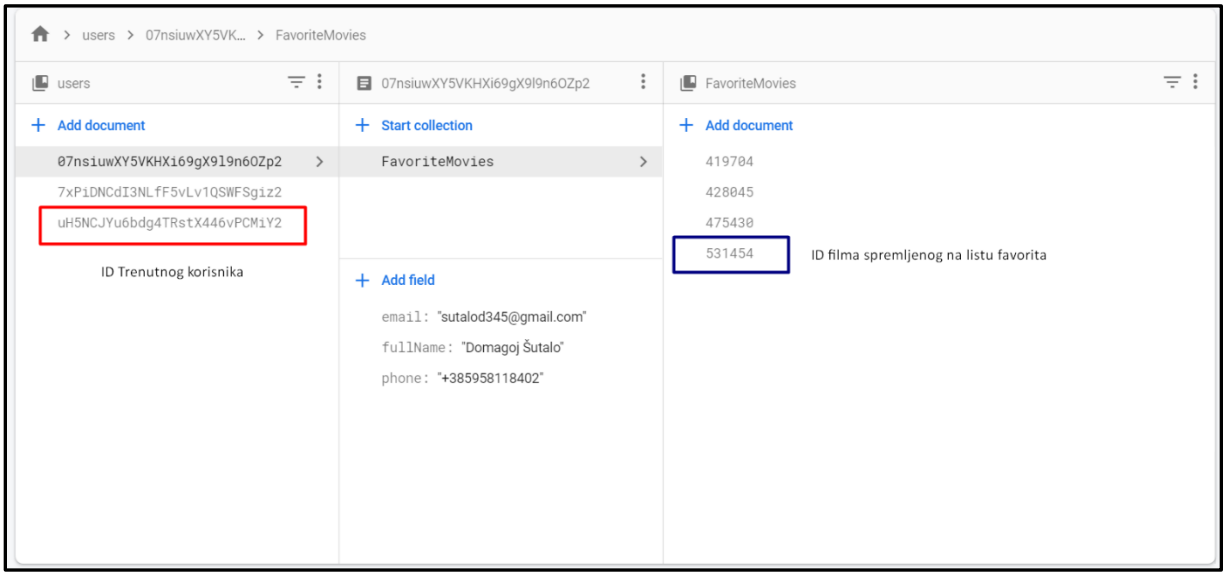
4.8.2. Baza podataka

Kao što je i ranije rečeno za spremanje podataka odnosno filmova korištena je online baza podataka odnosno Cloud Firestore u sklopu Firebase-a.

```
@Override
public void onFavoritesClickListener( int position) {


    Map<String,Object> movie = new HashMap<>();
    movie.put(KEY_TITLE,adapter.getDataList().get(position).getName());
    movie.put(KEY_RATING,adapter.getDataList().get(position).getRating());
    movie.put(KEY_DESCRIPTION,adapter.getDataList().get(position).getDescription());
    movie.put(KEY_IMAGE, adapter.getDataList().get(position).getImage_link());

    movieID = adapter.getDataList().get(position).getMovieID().toString();

    firebaseFirestore.collection( collectionPath: "users").document(userID).collection( collectionPath: "FavoriteMovies").document(movieID).set(movie)
            .addOnSuccessListener((OnSuccessListener) (aVoid) → {
                    Toast.makeText( context: MainActivity.this,  text: "Movie added to Favorites", Toast.LENGTH_SHORT).show();
            })
            .addOnFailureListener((e) → {
                    Toast.makeText( context: MainActivity.this,  text: "Error!", Toast.LENGTH_SHORT).show();
                    Log.d(TAG, e.toString());
            });


}
```

Slika 4.11. Spremanje filma u bazu podataka

Spremanje filmova u bazu podataka korišteno je samo u pozivu funkcije *onFavoritesClickListener()* koja od RecyclerAdaptera dobiva film na kojega je korisnik kliknuo te njega stavlja na listu favorita kod korisnika koji je trenutno prijavljen u sustav pomoću funkcije *put()* koja stavlja određenu informaciju u mapu filma (npr. ime filma, zatim njegovu ocjenu itd.) (slika 4.11). Kada je dobivena svaka tražena informacija u vezi filma, tada se preko identifikacijske oznake filma taj film sprema u zbirku *"FavoriteMovies"* kao što je i vidljivo na slici 4.11 Kao i pri registraciji korisnika, kod spremanja filma na listu favorita odnosno u bazu podataka, ona se odmah pojavljuje u Firebase konzoli (slika 4.12).



Slika 4.12. Prikaz spremljenih filmova na listu favorita

4.8.3. Korištenje API poziva u aplikaciji

Kako bi se moglo koristiti ovu online bazu podataka, prvo je bilo potrebno napraviti račun na njihovoj službenoj stranici kako bi se dobio ključ koji treba biti korišten pri svakom pozivu API-ja, a za svako povezivanje sa TMDb-om i slanjem zahtjeva korištena je biblioteka Retrofit. Prilikom dohvaćanja podataka u aplikaciji, u programu Android Studio korišteno je više klasa, kako bi se smanjila složenost svih klasa, odnosno kako bi svaka od njih radila jedan posao i stoga bila lakša za razumijeti. Klase korištene u dohvaćanju podataka sa TMDb-ja su sljedeće:

- *NetworkUtils* - Stvara se instanca Retrofit sučelja i koristi se kao Singleton
- *APIInterface* - Navode se krajnje točke, odnosno *queryji* pojedinih HTTP zahtjeva unutar GET anotacije (predstavlja HTTP metodu), omogućuje nam da pozovemo određenu funkciju (search movies, get popular movies, itd.)
- *MovieListResponse* - HTTP zahtjev izvršava se *enqueue* metodom nakon koje je moguće dohvatiti Response objekt, čije tijelo predstavlja klasu *MovieListResponse*
- *Movie* - predstavlja izgled pojedinog filma, odnosno njegove dijelove, npr. ime filma, ocjena, opis, poster itd.

U nastavku je prikazan kod svake pojedine klase (slike 4.13, 4.14, 4.15 i 4.16).

```
public class NetworkUtils {

    private static APIInterface apiInterface;

    public static APIInterface getApiInterface(){
        if(apiInterface == null){
            Retrofit retrofit = new Retrofit.Builder()
                    .baseUrl("https://api.themoviedb.org/")
                    .addConverterFactory(GsonConverterFactory.create())
                    .build();
            apiInterface = retrofit.create(APIInterface.class);
        }
        return apiInterface;
    }
}
```

Slika 4.13. Prikaz klase NetworkUtils

```
public interface APIInterface {
    @GET("3/search/movie")
    Call<MovieListResponse> getMovies(
            @Query("api_key") String api_key,
            @Query("query") String query
    );

    @GET("3/movie/popular")
    Call<MovieListResponse> getPopular(
            @Query("api_key") String apiKey,
            @Query("language") String language,
            @Query("page") int number

    );

    @GET("3/movie/{movie_id}")
    Call<MovieGenreList> getGenre(
            @Path("movie_id") int movieID,
            @Query("api_key") String apiKey,
            @Query("language") String language
    );

}
```

Slika 4.14. Prikaz klase APIInterface

```
public class MovieListResponse {

    private int page;
    private int total_results;
    private int total_pages;
    private List<Movie> results;

    public List<Movie> getResults() {
        return results;
    }

    public int getPage() { return page; }

    public int getTotal_results() { return total_results; }

    public int getTotal_pages() { return total_pages; }


    public void setPage(int page) { this.page = page; }

    public void setTotal_results(int total_results) { this.total_results = total_results; }

    public void setTotal_pages(int total_pages) { this.total_pages = total_pages; }

    public void setResults(List<Movie> results) { this.results = results; }
}
```

Slika 4.15. Prikaz klase MovieListResponse

```java
public class Movie {
    @SerializedName("title")
    private String name;
    @SerializedName("vote_average")
    private String rating;
    @SerializedName("overview")
    private String description;
    @SerializedName("poster_path")
    private String image_link;
    @SerializedName("id")
    private Integer movieID;
    private String genre;
    @SerializedName("genre_ids")
    private List<Integer> genreIDs;
    public List<Integer> getGenreIDs() {
        return genreIDs;
    }

    public Movie(){}
    public Movie(String name, String rating,String description,String image_link){
        this.name = name;
        this.rating = rating;
        this.description = description;
        this.image_link = image_link;
    }

    public String getName() { return name; }

    public String getRating() { return rating; }

    public String getDescription() { return description; }

    public String getImage_link() { return image_link; }

    public Integer getMovieID() {
        return movieID;
    }
}
```

Slika 4.16. Prikaz klase Movie

4.8.4. Analiza lica korištenjem MLKit-a

Kao što je i ranije navedeno, kao rješenje problema pronalaska filma na osnovi emocije lica, korišten je Firebase-ov Machine Learning Kit. Prilikom korištenja ovoga alata sva njegova funkcionalnost predstavljena je u klasi FaceEmotion čiji su najvažniji dijelovi koda su prikazani na slijedećim slikama.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == CropImage.CROP_IMAGE_ACTIVITY_REQUEST_CODE) {
        CropImage.ActivityResult result = CropImage.getActivityResult(data);
        if (resultCode == RESULT_OK) {
            if (result != null) {
                Uri uri = result.getUri(); //path of image in phone
                imageView.setImageURI(uri); //set image in imageview
                textView.setText(""); //so that previous text don't get append with new one
                detectFaceFromImage(uri);
            }
        }
    }
}
```

Slika 4.17. Prikaz metode onActivityResult

Korištenjem *onActivityResult* metode omogućujemo korisniku odabir fotografije lica, gdje korisnik može odabrati želi li se uslikati u stvarnom vremenu (engl. *real-time*), a isto tako može odabrati fotografiju iz galerije. Radi lakšeg objašnjenja, pored svake bitne funkcije korišteni su komentari. Moguće je vidjeti ako je fotografija uspješno odabrana, te se postavlja putanja slike kao i stvarna slika u objekt *uri* koji je korišten u metodi *detectFaceFromImage* koja se poziva na kraju ove metode.

```
private void detectFaceFromImage(Uri uri) {
    try {
        image = FirebaseVisionImage.fromFilePath( context: FaceEmotion.this, uri);
        FirebaseVisionFaceDetectorOptions highAccuracyOpts =
                new FirebaseVisionFaceDetectorOptions.Builder()
                        .setPerformanceMode(FirebaseVisionFaceDetectorOptions.ACCURATE)
                        .setLandmarkMode(FirebaseVisionFaceDetectorOptions.ALL_LANDMARKS)
                        .setClassificationMode(FirebaseVisionFaceDetectorOptions.ALL_CLASSIFICATIONS)
                        .setContourMode(FirebaseVisionFaceDetectorOptions.ALL_CONTOURS)
                        .build();
        FirebaseVisionFaceDetector detector = FirebaseVision.getInstance()
                .getVisionFaceDetector(highAccuracyOpts);

        detector.detectInImage(image)
                .addOnSuccessListener((OnSuccessListener) (faces) → {
                        for (FirebaseVisionFace face : faces) {...}
                })
                .addOnFailureListener(
                        new OnFailureListener() {
                            @Override
                            public void onFailure(@NonNull Exception e) {
                                // Task failed with an exception
                                // ...
                            }
                        });

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

Slika 4.18. Prikaz metode detectFaceFromImage

Metodom *detectFaceFromImage* dolazimo do prvog korištenja MLKita, te kao što je ranije navedeno ova metoda kao argument prima objekt *uri* u kojemu je spremljena putanja do fotografije koju je korisnik ranije odabrao. Ova fotografija se sprema u objekt klase *FirebaseVisionImage* koji je u ovome slučaju nazvan *image*, te se pomoću *detektora* klase *FirebaseVisionFaceDetector* pokušavaju detektirati značajke lica, konture te karakteristike lica. U metodi *detectInImage* dobiva se vjerojatnost osmjeha (engl. *smile probability*), odnosno postotak nasmiješenosti korisnika koje je prikazano na sljedećoj slici, slika 4.19.

```
if (face.getSmilingProbability() != FirebaseVisionFace.UNCOMPUTED_PROBABILITY) {
    float smileProb = face.getSmilingProbability();
    smileProbability = smileProb * 100;
    textView.append("SmileProbability: " + ("" + smileProb * 100).subSequence(0, 4) + "%" + "\n\n");
}
```

Slika 4.19. Dobivanje postotka nasmiješenosti

Program prvo provjerava je li postotak nasmiješenosti nekakva vrijednost s kojom se može računati, te ako je, tu vrijednost sprema i množi sa 100 kako bi iz decimalne vrijednosti dobio postotak.

```
private void getMovieGenre() {
    if (smileProbability < 0){
        Toast.makeText( context: this,  text: "Error - You don't have a face", Toast.LENGTH_SHORT).show();
    }
    else if(smileProbability > 0 && smileProbability <= 20){
        Toast.makeText( context: this,  text: "Recommend category based on your emotions: Comedy", Toast.LENGTH_SHORT).show();
        category = "Comedy";
        categoryID = 35;

    }
    else if(smileProbability > 20 && smileProbability <= 40){
        Toast.makeText( context: this,  text: "Recommend category based on your emotions: Action", Toast.LENGTH_SHORT).show();
        category = "Action";
        categoryID = 28;

    }
    else if(smileProbability > 40 && smileProbability <= 60){
        Toast.makeText( context: this,  text: "Recommend category based on your emotions: Drama", Toast.LENGTH_SHORT).show();
        category = "Drama";
        categoryID = 18;

    }
    else if(smileProbability > 60 && smileProbability <= 80){
        Toast.makeText( context: this,  text: "Recommend category based on your emotions: Thriller", Toast.LENGTH_SHORT).show();
        category = "Thriller";
        categoryID = 53;

    }
    else if(smileProbability > 80 && smileProbability <= 100){
        Toast.makeText( context: this,  text: "Recommend category based on your emotions: Horror", Toast.LENGTH_SHORT).show();
        category = "Horror";
        categoryID = 27;

    }
    else{
        Toast.makeText( context: this,  text: "Something's wrong", Toast.LENGTH_SHORT).show();
    }
```

Slika 4.20. prikaz metode getMovieGenre

Na kraju pozvana je metoda *getMovieGenre* koja provjerava postotak nasmiješenosti korisnika sa fotografije, te utvrđuje po tom postotku najbolji žanr filmova za korisnika:

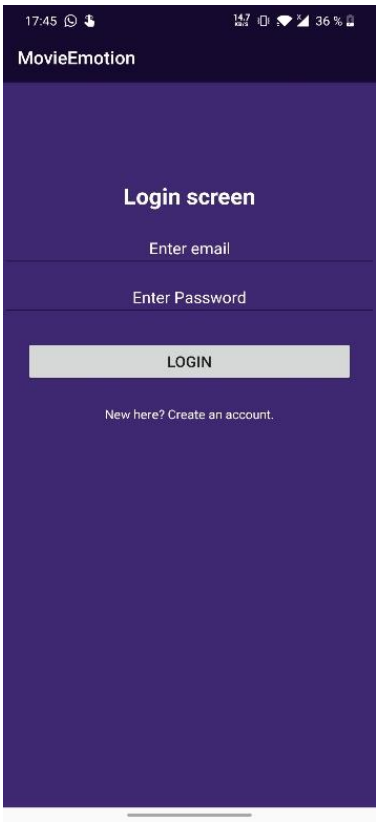
- Postotak nasmiješenosti <0,20] - preporučeni žanr : Komedija
- Postotak nasmiješenosti <20,40] - preporučeni žanr : Akcija
- Postotak nasmiješenosti <40,60] - preporučeni žanr : Drama
- Postotak nasmiješenosti <60,80] - preporučeni žanr : Thriller
- Postotak nasmiješenosti <80,100] - preporučeni žanr : Horror

5. NAČIN RADA I ISPITIVANJE APLIKACIJE

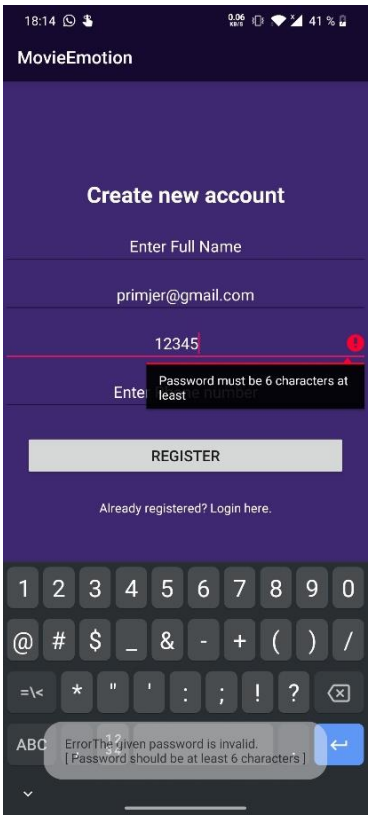
U ovome poglavlju nalazi se detaljan opis korištenja aplikacije i prolazak kroz sve njene funkcionalnosti, od registracije korisnika do preporuke filmova na osnovi emocija te nakon toga ispitivanje kvalitete aplikacije.

5.1. Korištenje mobilne aplikacije

Ulaskom u aplikaciju dobiva se prikaz početnog zaslona odnosno *Login screen-a* koji od korisnika očekuje njegovu adresu e-pošte kao i zaporku s kojom se registrirao u aplikaciju (slika 5.1).



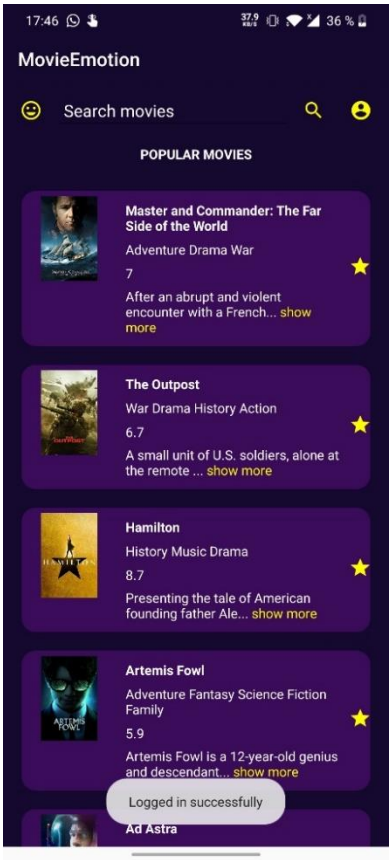
Slika 5.1 Login activity



Slika 5.2. Register activity

Međutim, ako korisnik još uvijek nema izrađen račun za pristup aplikaciji, njega može lako napraviti klikom na *"New here? Create an account"* prilikom kojega se otvara nova aktivnost (engl. *activity)* koja od korisnika traži njegovo puno ime, adresu e-pošte i zaporku, s kojom kasnije može pristupiti aplikaciji, kao i njegov broj mobitela. Prilikom ispunjavanja korisniku će se javiti greška ako mu adresa e-pošte nije ispravne strukture *primjer@gmail.com* te ako mu zaporka sadrži manje od šest znakova (slika 5.2).

Klikom na *Register* odnosno *Login,* pretpostavljajući da je sve ispravno upisano, aplikacija sprema korisnika u Firebase bazu podataka, te ulazi u glavni dio aplikacije (slika 5.3). Manji dio glavnog dijela aplikacije, koji predstavlja gornji dio ekrana, zauzimaju dva gumba, kao i traka za pretraživanje koja se nalazi između njih, dok većina glavnog dijela predstavlja *Recycler View* u kojemu se nalazi lista popularnih filmova koji u sebi sadrže naslov, žanr filma, njegovu ocjenu kao i kratak opis koji se može produljiti klikom na *show more*, te gumb koji je prikazan pomoću žute zvijezdice. Klikom na taj gumb film je spremljen na online Firebase bazu podataka, a taj film je moguće kasnije vidjeti u listi favorita koja se nalazi u *user activityju*.

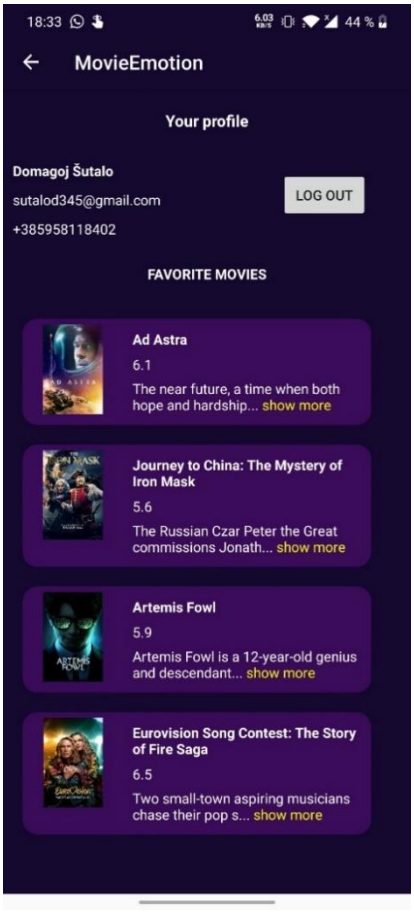


Slika 5.3. Glavni izbornik

Kako bi se došlo do liste favorita, odnosno *user activity*ja, potrebno je kliknuti na gumb u gornjem desnom kutu koji je prikazan crnim čovječuljkom na žutoj pozadini, prilikom kojega se otvara nova aktivnost. U njemu je moguće jasno vidjeti listu favorita, kao i osnovne informacije o korisniku i gumb za odjavu iz sustava (slika 5.3).

Vraćanjem na glavni dio aplikacije, pomoću trake za pretraživanje (engl. *search bar*) moguće je pretraživati filmove po njihovom nazivu, te se prilikom toga lista popularnih filmova zamjenjuje listom filmova koji u svome nazivu sadrže barem jednu riječ iz unosa (slika 5.4).

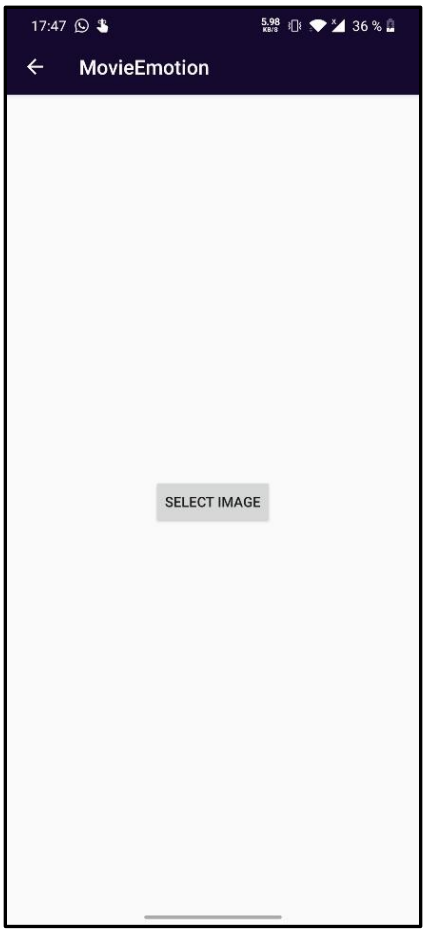
Slika 5.5. primjer search bar-a pomoću ključne riječi "batman"



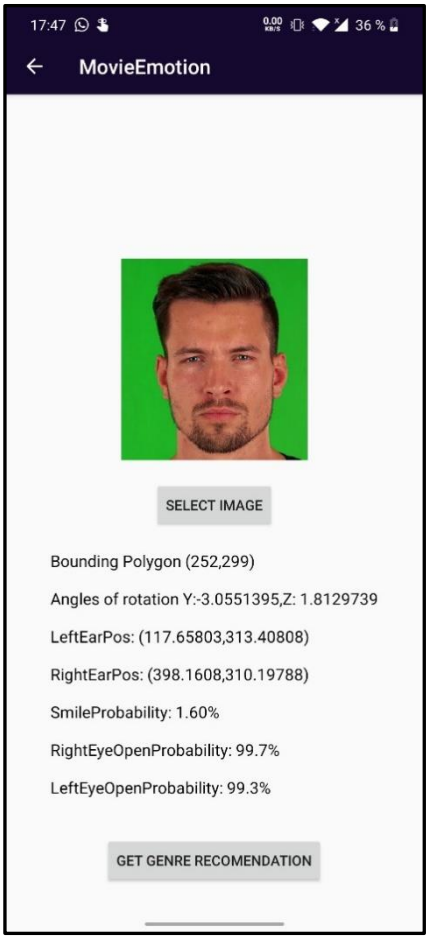
Slika 5.4. Korisnički (engl. *user*) activity

Ako korisnik ne zna koji film želi pretražiti, taj problem je riješen pomoću gumba u gornjem lijevom kutu (slika 5.5) koji otvara sučelje za pretraživanje filmova na osnovi raspoloženja

korisnika što je zapravo i glavna značajka ove aplikacije. Ulaskom u novu aktivnost (slika 5.6) korisniku je ponuđen gumb za izbor slike, te klikom na njega dobiva opciju želi li se direktno uslikati iz aplikacije, ili želi li odabrati fotografiju koja se već nalazi u njegovoj galeriji.

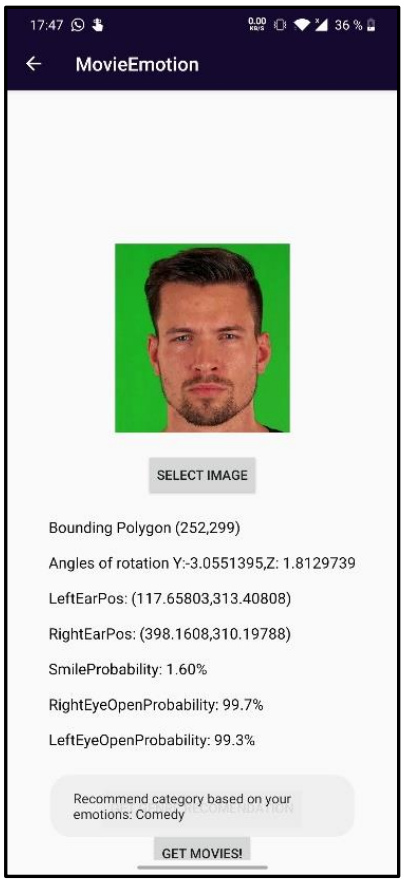


Slika 5.6. Emotion activity

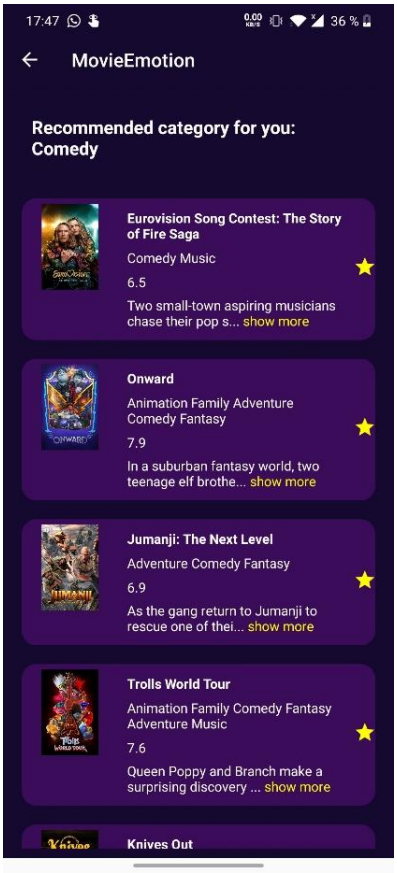


Slika 5.7. Prikaz informacija o korisniku

Odabirom slike korisniku se prikazuju informacije (slika 5.7) kao što su *LeftEarPosition*, *RightEarPosition*, *LeftEyeProbability*, *RightEyeProbability* gdje aplikacija prikazuje poziciju lijevog i desnog uha kao i vjerojatnost lijevog ili desnog oka, ali u ovome slučaju bitna je informacija *SmileProbability* koja označava vjerojatnost osmijeha na licu korisnika, te pomoću tog postotka aplikacija preporučuje žanr filma korisniku.



Slika 5.8. Toast poruka I novostvoreni gumb za dohvaćanje predloženih filmova



Slika 5.9. Prikaz predloženih filmova na osnovi ekspresije lica

Klikom na *Get genre recomendation* generira se *Toast* poruka sa žanrom filmova, te se također prikazuje novi gumb, *Get movies* (slika 5.8) koji korisnika vodi u novu aktivnost.

U novoj aktivnosti korisniku se prikazuje kategorija odnosno žanr filmova, kao i sami filmovi koji su generirani na osnovi emocija lica (slika 5.9).

5.2. Ispitivanje aplikacije

Prilikom ispitivanja aplikacije odabrana su tri različita modela sa tri različita izraza lica. Za primjer ispitivanja odabrana su tri granična slučaja postotka nasmiješenosti radi jednostavnosti kao i iz razloga toga ako granični slučaji ne dobiju očekivani rezultat, odmah je moguće pretpostaviti da alat, kao i aplikacija, nisu ispravni.

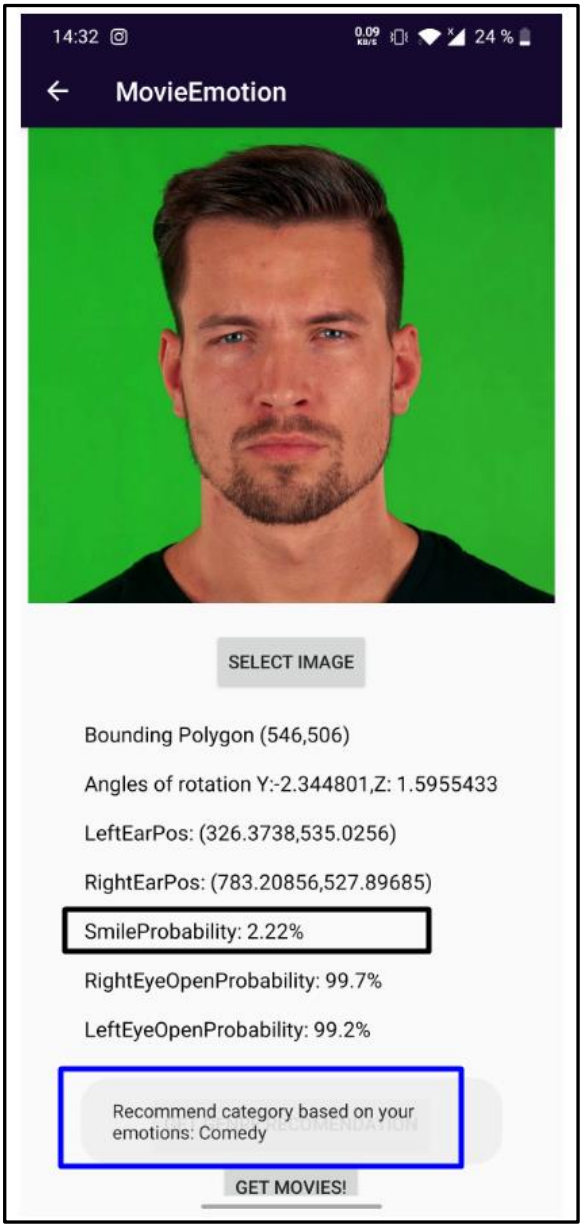
5.2.1. Prvo ispitivanje

Za prvo ispitivanje odabran je model sa ozbiljnim izrazom lica (slika 5.10).

Slika 5.10. Prikaz modela sa ozbiljnim izrazom lica

Radi jednostavnosti i preglednosti napravljena je tablica koja prikazuje pretpostavljene rezultate kao i stvarne rezultate, te je ispod nje također prikazan stvarni rezultat (slika 5.11).

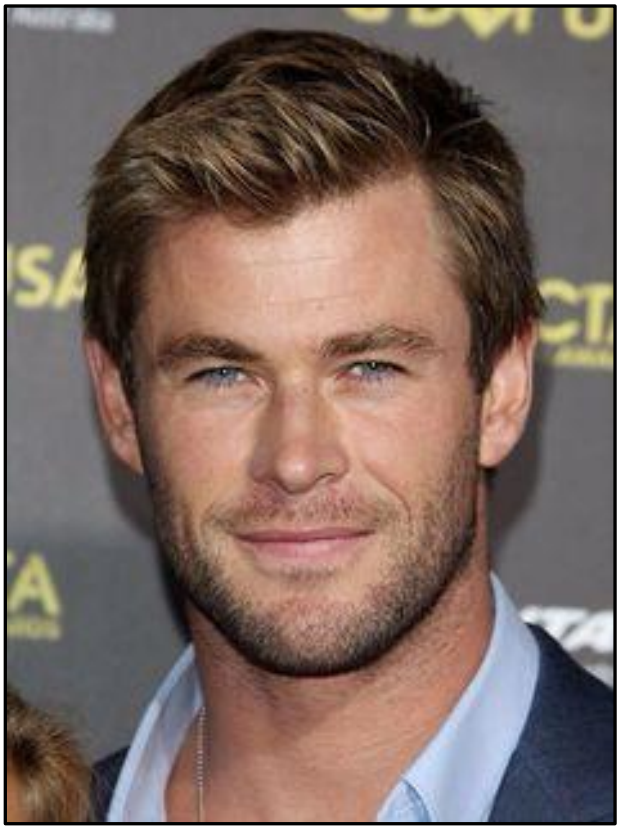
	Postotak nasmiješenosti	Preporuka žanra
Očekivani rezultat	Do 5%	Komedija
Stvarni rezultat	2.22%	Komedija



Slika 5.11. Prikaz stvarnih rezultata

5.2.2. Drugo ispitivanje

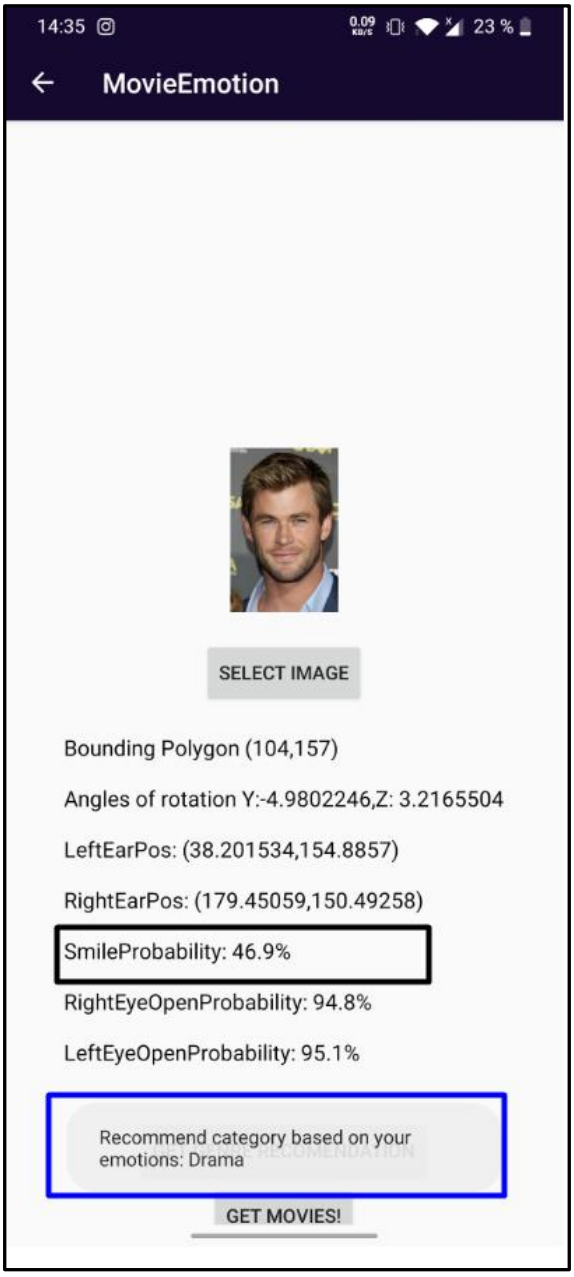
Za drugo ispitivanje odabran je model koji nije ni pretjerano ozbiljan, a isto tako ni pretjerano nasmiješen kao što je vidljivo na slici 5.12.



Slika 5.12. Prikaz modela za drugo ispitivanje

Kao i u prvom ispitivanju, napravljena je tablica očekivanih i stvarnih rezultata kao i dokaz o stvarnim rezultatima na slici 5.13.

	Postotak nasmiješenosti	**Preporuka žanra**
Očekivani rezultat	Oko 50%	Drama
Stvarni rezultat	46.9%	Drama



Slika 5.13. Prikaz stvarnih rezultata

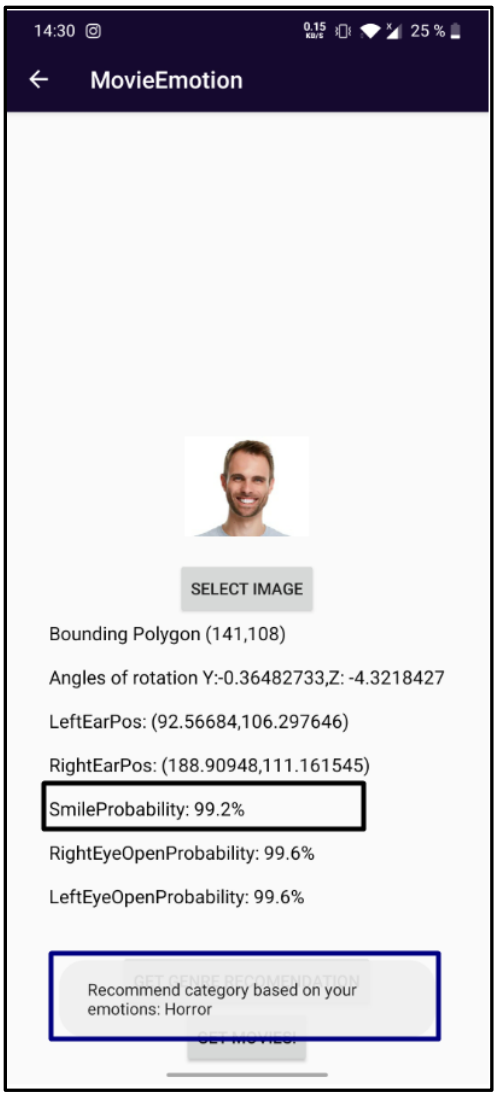
5.2.3. Treće ispitivanje

Za treće ispitivanje odabran je nasmiješen model što je i vidljivo na slici 5.14.

Slika 5.14. Prikaz modela trećeg ispitivanja

Isto kao i u prethodna dva slučaja, napravljena je tablica sa očekivanim i stvarnim ishodima ispitivanja, kao i prikaz stvarnih rezultata (slika 5.15).

	Postotak nasmiješenosti	**Preporuka žanra**
Očekivani rezultat	Između 90% i 100%	Horror
Stvarni rezultat	99.2%	Horror



Slika 5.15. prikaz stvarnih rezultata

6. ZAKLJUČAK

Zbog rasta stranica za prikaz serija i filmova u zadnjih nekoliko godina, dolazi do sve većeg populariziranja filmova i serija, te je cilj ovog završnog rada omogućiti lakšu pretragu filmova, odnosno njihovu preporuku kako bi korisnik u što kraćem roku pronašao film. U odnosu na slična rješenja, ova aplikacija nudi opciju preporuke filmova koja se ostvaruje analizom korisnikovog lica nakon kojega korisnik dobiva filmski žanr preporučen za njega kao i listu filmova iz toga žanra. Osim toga, korisniku je omogućena pretraga filmova prema imenu, kao i lista popularnih filmova, gdje se svaki od njih može staviti na listu favorita u koju korisnik ima uvid.

Za izradu aplikacije korišteno je programsko okruženje Android studio. Korištenjem XML-a napravljen je izgled sučelja aplikacije, dok su sve funkcije koje rade na strani poslužitelja ostvarene u programskom dijelu putem programskog jezika Java. Za potrebe aplikacije, korišten je Firebase koji je omogućio: korištenje baze podataka (Firebase Firestore) i analizu lica (Firebase Machine Learning Kit). Također, kako bi filmovi mogli biti dohvaćeni sa on-line baze podataka (The Movie Database), korišten je TMDb API. Prijedlog modela mobilne aplikacije uspješno je ostvaren i isto tako ispitan za granične slučajeve kako bi se pokazao valjanim i točnim.

Nakon ispitivanja, rješenje mobilne aplikacije pokazalo se ispravnim, no isto tako bi se moglo poboljšati uvođenjem televizijskih serija, dodavanjem korisnikovih ocjena pojedinim filmovima i serijama kao i uvođenjem opcije korisničkog odabira kriterija za način preporuke koji može biti u skladu s raspoloženjem ili obrnuto proporcionalno raspoloženju.

LITERATURA

[1] D. Moskovitz, Who Are Netflix's Main Competitors? https://www.investopedia.com/articles/markets/051215/who-are-netflixs-main-competitors-nflx.asp [posjećeno 10.9.2020.]

[2] IMDb, https://help.imdb.com/article/imdb/general-information/what-is-imdb/G836CY29Z4SGNMK5?ref_=helpart_nav_1#

[3] Rotten Tomatoes, https://www.rottentomatoes.com/about

[4] J. Šumečki, Sustavi za preporučivanje, 2010., str. 1-38.

[5] F.O. Isinkaye, Y.O. Folajimi, B.A. Ojokoh, Recommendation Systems: Principles, Methods and Evaluation, Egyptian Informatics Journal, Vol.16, 2015., str. 261-273.

[6] R. Khatami, G. Mountrakis, S. V. Stehman, A meta-analysis of Remote Sensing Research on Supervised Pixel-Based Land-cover Image Classification Processes: General Guidelines for practitioners and Future Research, Remote Sensing of Environment, Vol. 177, 2016, str. 89-100.

[7] M. Čupić, B. Dalbelo Bašić, J. Šnajder, Umjetne neuronske mreže, FER Zagreb 2008, str. 7-8.

[8] P. Gupta, Decision Trees in Machine Learning, https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052 [posjećeno 10.9.2020.]

[9] B. Dalbelo Bašić, Stabla odluke, FER Zagreb 2002., str. 1.

[10] J. Chen, Android Operating System https://www.investopedia.com/terms/a/android-operating-system.asp [posjećeno 8.7.2020.]

[11] J. Callaham, The History of Android: The Evolution of the Biggest Mobile OS in the World https://www.androidauthority.com/history-android-os-name-789433/ [posjećeno 1.9.2020.]

[12] Android Studio https://developer.android.com/studio/intro [posjećeno 1.9.2020.]

[13] M. Čupić, Programiranje u Javi, 2012., str. 1-3.

[14] XML, https://www.tutorialspoint.com/xml/xml_overview.htm [posjećeno 7.7.2020.]

[15] The Movie Database, https://www.themoviedb.org/

[16] D. Stevenson, What is Firebase? The Complete Story, Abridged. https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0 [posjećeno 7.7.2020.]

[17] Detekcija lica,
https://developers.google.com/ml-kit/vision/face-detection/face-detection-concepts#contours
[posjećeno 7.7.2020.]

ŽIVOTOPIS

Domagoj Šutalo rođen je 25.3.1999. u Đakovu. Nakon pohađanja Osnovne škole Vladimir Nazor u Đakovu, upisuje jezičnu gimnaziju Antun Gustav Matoš. 2017. godine završava gimnaziju te upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija u Osijeku, smjer računarstvo. Od programskih jezika poznaje C, C++, Javu I HTML te najviše uživa u razvoju Android aplikacija.

SAŽETAK

Zadatak završnog rada je izrada Android mobilne aplikacije za preporučivanje filmskih sadržaja pomoću analize slike lica. S obzirom da većina ljudi danas koristi Android pametne mobilne uređaje, aplikacija je izrađena u programskome okruženju Android Studio pomoću programskih jezika XML i Java. Korištenjem platforme Firebase, omogućeno je spremanje podataka o filmovima u njenu bazu kao i korištenje alata za strojno učenje (MLKit) za analizu lica. Aplikacija korisniku omogućuje registriranje, odnosno prijavu u sustav kao i prikaz popularnih filmova, pretraživanje filmova prema imenu i preporuku filmskih sadržaja na temelju analize slike lica. Ispitivanjem aplikacije, pokazalo se da je ona učinkovita za ispitne slučajeve u završnom radu.

Ključne riječi: analiza lica**,** Android mobilna aplikacija, filmovi, preporuka filmova.

ABSTRACT

The task of this paper is to create an Android mobile application for recommending movie content using face analysis. Since most people today use Android smartphones, the application was created in Android studio programming environment using a combination of XML and Java programming languages. Using the Firebase platform, it is possible to save movie data to its database as well as using its machine learnirng kit (MLKit) for face analysis. The application allows the user to register or to log in to the system, as well as viewing popular movies, searching movies by name and recommending movie content using face analysis. By testing the application, it proved to be useful in all three cases presented in the paper.

Key words: face analysis**,** Android mobile application, movies, movie recommendation.

PRILOZI

Prilog 1. Završni rad u datoteci docx

Prilog 2. Završni rad u datoteci pdf

Prilog 3. Programski kod projekta mobilne aplikacije