

*Prolećni semestar, 2017/18*

*PREDMET: CS330 Razvoj mobilnih aplikacija*

NMD – Dokumentacija

Ime i prezime: **Nemanja Kuzmanović**

Broj indeksa: **2851**

Profesor: **Vladimir Milićević**

Sadržaj

[UVOD 3](#_Toc515927985)

[TEHNOLOGIJE 4](#_Toc515927986)

[IMPLEMENTACIJA 5](#_Toc515927987)

[DEMONSTRACIJA 15](#_Toc515927988)

[ZAKLJUČAK 22](#_Toc515927989)

[LITERATURA 22](#_Toc515927990)

# UVOD

NMD Aplikacija je aplikacija izrađena u okruženju Android Studio 3.0. Aplikacija ima svrhu da korisnik može pogledati kratak opis filmova, koji su najpopularniji, najnoviji… Dakle korisnik bira jednu od kategorija datih, potom dobija listu filmova na kojima se nalazi slika filma (poster), naziv I rejting. Kada korisnik odabere željeni film sa liste, dobiće detaljniji opis tog filma, u vidu godine kada je izašao film, opis, kao I youtube traileri za film (može ih imati više).

# TEHNOLOGIJE

Aplikacije je kao što je već navedeno, izrađena u okruženju Android Studio. Pored nje korišćen je Genymotion kao emulator, na kome su instalirani GAPPS (google apps) servisi, a pored kojih je podešen play store nalog i skinuti gmail i youtube aplikacija, zarad potreba projekata, i demonstracije.

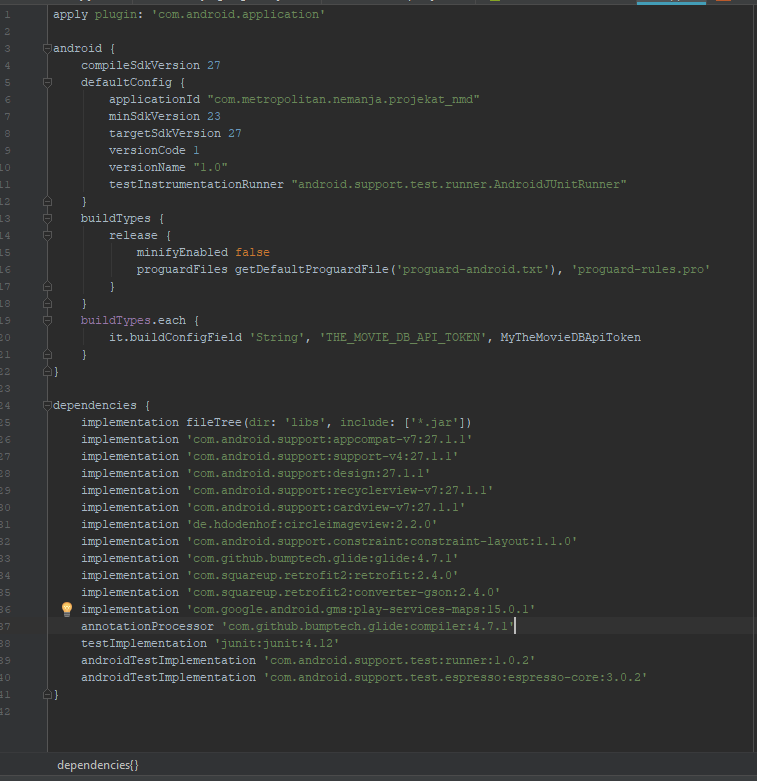
Što se tiče same aplikacije, ona ne koristi bazu podataka kako bi se filmovi čuvali u njoj sa propratnim detaljima, nego se koristi API sajta themoviedb.org koji vraća u JSON formatu niz filmova sa opisima, koje dalje parsiramo u aplikaciji u željenu model klasu Movie, pa dalje isporučujemo pogledima po željenom nahođenju.

Kako bi preuzeli, dakle, taj JSON niz koji API vraća, moramo imati neki HTTP klijent koji bi obrađivao GET zahteve. Obzirom da je na vežbama rađen Apache, koji je u međuvremenu od verzije 6.0 deprecated (napušten), alternativa je pronađena u vidu Retrofit biblioteke, koja je opensource i jednostavno se može integrisati u aplikaciju tako što u gradle properties definišemo datu implementaciju sa github-a Retrofita (iz dokumentacije). Retrofit je jednostavan i lak za korišćenje, a sintaka je veoma laka za razumevanje i odlično je struktuirana, pa biblioteku nije bilo teško osposobiti za rad.

Pošto sa API-a kupimo i slike (postere) filmova, potrebna je biblioteka koja će se time baviti (kešovati slike, procesuirati ih...). Ovde je uzeta preporuka samog Google-a koji preporučuje Glide biblioteku, koja je takođe opensource i može se jednostavno implementirati u grade properties kako bi je koristili. Pored Glide-a postoji i Picasso, međutim kako je neko od kolega već koristio tu biblioteku, odlučeno je da se izabere druga, i odluka je pala na Glide. Prednosti imaju i Glide i Picasso, uglavnom su tehničke, primera radi, kako se radi kešovanje i sl. A najveća prednost Glide-a je, pored što je preporučen od strane Google-a, da umesto prosleđivanja kontekstra (context), može se proslediti mnogo više parametara, kao što su Fragmenti, Aktivnosti i slično.

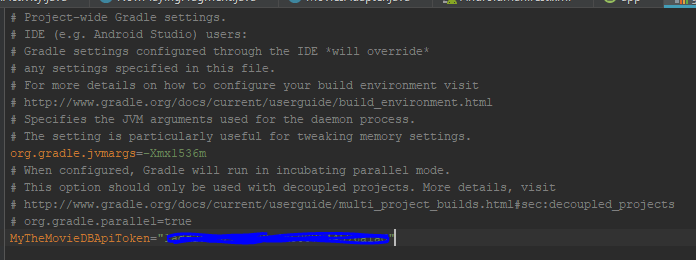
Konačno, korišćene su biblioteke Android Design Support Library, koje Google preporučuje za Material Design. U to spadaju Recycleview,NestedScrollView i tako dalje. Ovo su Google-ova usmerenja kako napraviti bolju lepšu korisniju i konstantniju aplikaciju, pa ih nije loše pratiti, što je u ovom projektu i učinjeno. Takođe, koristi se još nekoliko dodatnih opensource rešenja, a jedno od njih je CircleImageView, koji sliži da prikaže sliku u definisanom krugu, koji možemo jednostavno, lako i brzo prilagoditi po našoj meri.

# IMPLEMENTACIJA

****

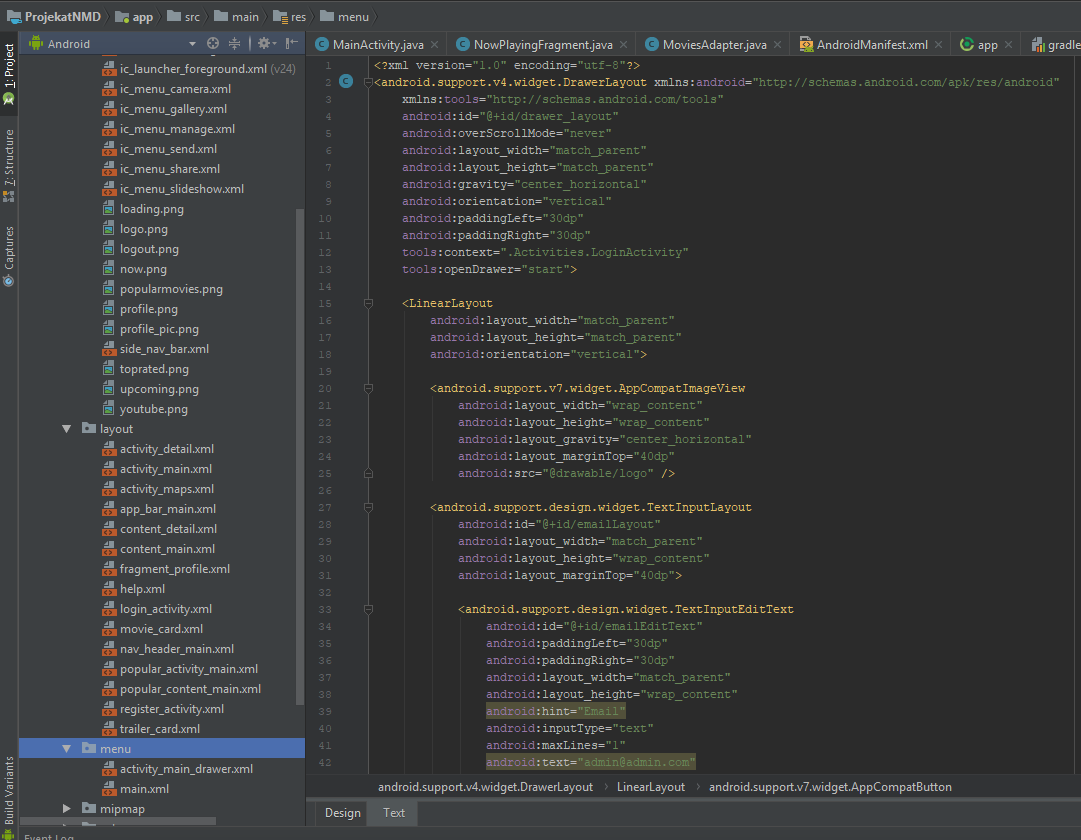
*Slika 1 – Gradle Build*

Dakle, kao što je već objašnjeno u implementaciji, koriste se nekoliko dodatnih biblioteka, opensource rešenja uglavnom. Pored toga, definisan je API Token, koji je preuzet sa sajta themoviedb.org, gde je prethodno kreiran nalog, opisana aplikacija za šta i čemu služi kako bi bio API odobren.



*Slika 2 – API Token*

U fajlu gradle.properties smo definisali API koji se koristi dalje kroz celu aplikaciju

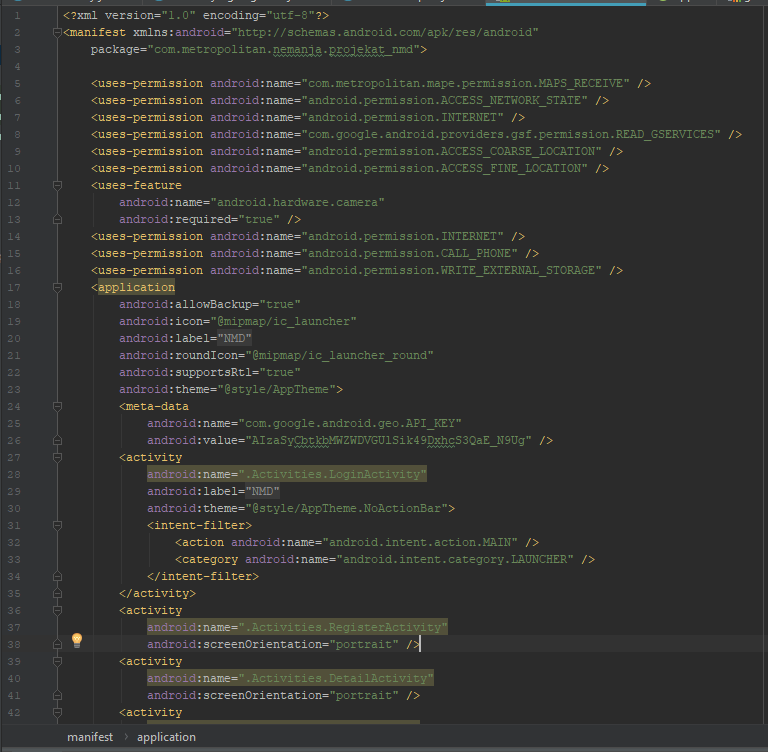


*Slika 3 – Layouti*

Obzirom da layout nije ništa do slaganje komponenata u željenu formu, neće biti svaki prikazan I detaljno objašnjen, jer relano nema za tim potrebe. U konketnom primeru loginlayout-a prikazana je delimično logika kojom se vodilo kroz celu aplikaciju, I tu uglavnom nema šta preterano detaljisati. Dakle koristimo poglede za različite potrebe, kao I Dugmad, EditTextove, ImageView I slično.

Android.support.design.widget biblioteka se odnosi na prethodno pomenuti Material Design. Jedina razlika je u sintaksi I krajnjem izgledu same komponente. Recimo, obično dugme je Button, dok je ovde AppCompatButton itd.

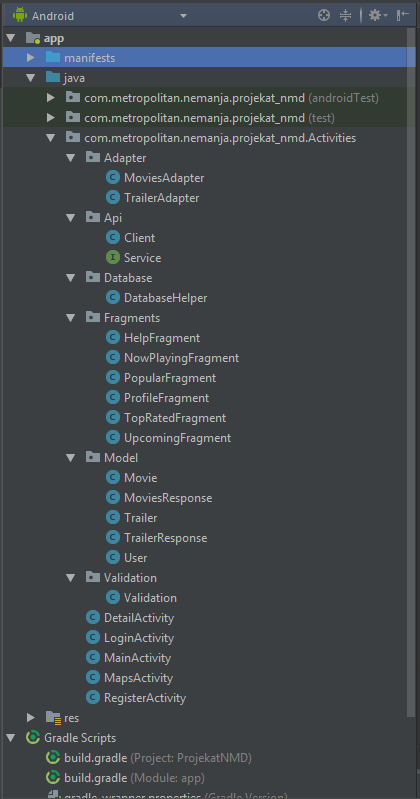
Takođe s leve strane se mogu videte razne slike koje su ubačene, i one su uglavnom, većinom, male ikonice koje se koriste kroz aplikaciju, koja će biti demonstrirana u nastavku.



*Slika 4 – Android Manifest*

U Android Manifestu su definisane dozvole koje su potrebne za aplikaciju, gde je osnovna I najbitnija/najpotrebnija dozvola Internet. Pored nje potrebno je definisati dozvolu za WRITE EXTERNAL STORAGE iz razloga što korisnici imaju mogućnost upload-ovanja svoje slike na njihov nalog, koja se uzima iz memorije telefona. Ostale dozvole se uglavnom odnose na Mapu.

Takođe u Android Manifestu je neophodno definisati sve aktivnosti koje se koriste u aplikaciji, a to su Login, Register, Detail I Maps Activity. Ostatak aplikacije funkcioniše na bazi fragmenata.



*Slika 5 – Hijerarhija klasa projekta*

Na sledećoj slici se može videte hijerarhija klasa samog projekta.

Prvenstveno imamo model klase. U klasi Movie nalaze se atributi filma koje vraća JSON, od kojih su neki naziv filma, opis, godina kada je film izašao, ocena korisnika I slično. Analogno tome, User klasa ima ime, prezime, lozinku, mail…

Trailer model je morao biti implementiran iz razloga što film može imati više videa koji se zapravo prosleđuju kao više atributa, a analogno sa time kako je implementirano parsiranje (call, callback) iz JSONa u listu, ovako je bilo najlakse I najefikasnije uraditi I mplementaciju.

Klase MoviesResponse I TrailerResponse, rade logiku preuzimanja, I implementiraju metodu getResults() koja vraća rezultate prosleđene od API-a GET requestom, korišćenjem Retrofit-a.

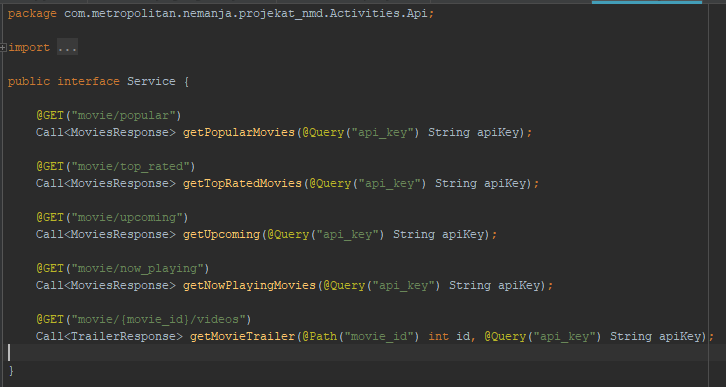
Klasa Validation, poseduje nekoliko metoda, I služi za validaciju unosa kod logina I registracije. Dakle, implementiraju se metode za proveru da li je unet bilo kakav tekst, I ukoliko nije u Layout EditTexta se prosleđuje crvena greška sa porukom da to polje nije uneto. Takođe, implementira se metoda za proveru da li uneti String zapravo u formatu maila, kao I da li se šifre poklapaju, jer se kod registracije mora potvrditi uneta lozinka.

Aktivnosti I Fragmenti su poprilično jednostavni za objasniti I razumeti pa se neće detaljnije obraditi.

DatabaseHelper služi za kreiranje SQLite baze podataka ukoliko ne postoji (ukoliko aplikacija nije već bila instalirana) ili manipulisanje podatcima (ukoliko aplikaija već postoji). Baza se čuva lokalno u memoriji aplikacije, I simulira bazu podatak koja bi postojala na nekom serveru. U bazi podataka čuvamo Korisnika kao entitet, sa atributima od kojih su neki, ime, prezime, email, lozinka kao I slika tipa Blob.

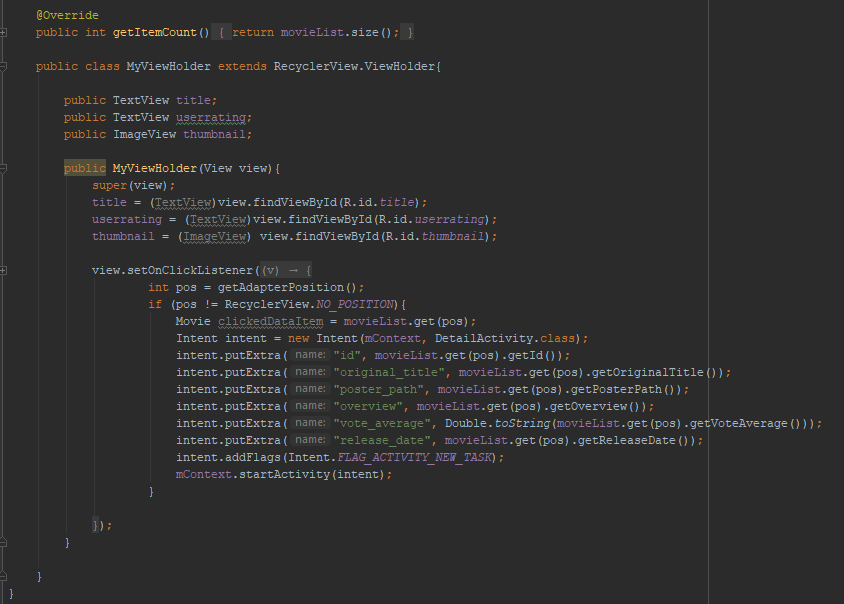
Takođe, u bazi imamo nekoliko metoda koje vrše logiku upisivanja novog korisnika, provere da li korisnik sa emailom već postoji u bazi, provera emaila I lozinke kod loginovanja u aplikaciju, uzimanje svih atributa korisnika I slično.

Client I Service su klase Retrofita gde se u Klijentskom delu definiše URL api-a, kod se u Service delu klase, definišu GET zahtevi sa željenom akcijom (Slika 5.1)



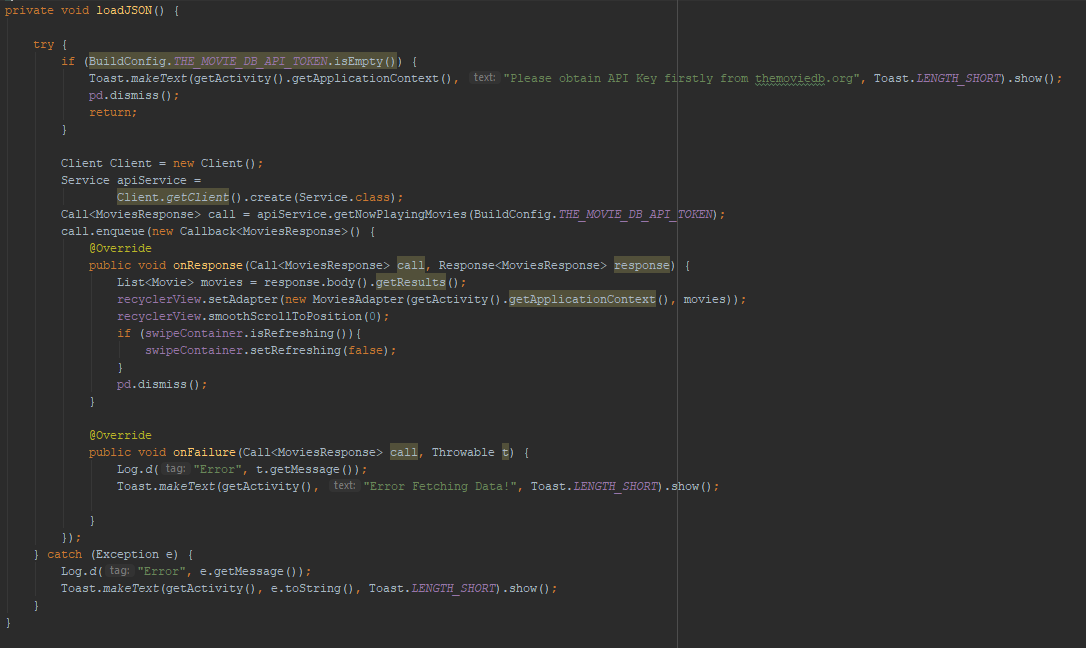
*Slika 5.1 – Service klasa*

Klase adapteri služe za prosleđivanje podataka u poglede.



*Slika 5.2 – Adapter Klase*

Prosleđuju se atributi filma iz JSONa u recycleview I dalje u DetailActivity kako ne bi ponovo slali zahtev.

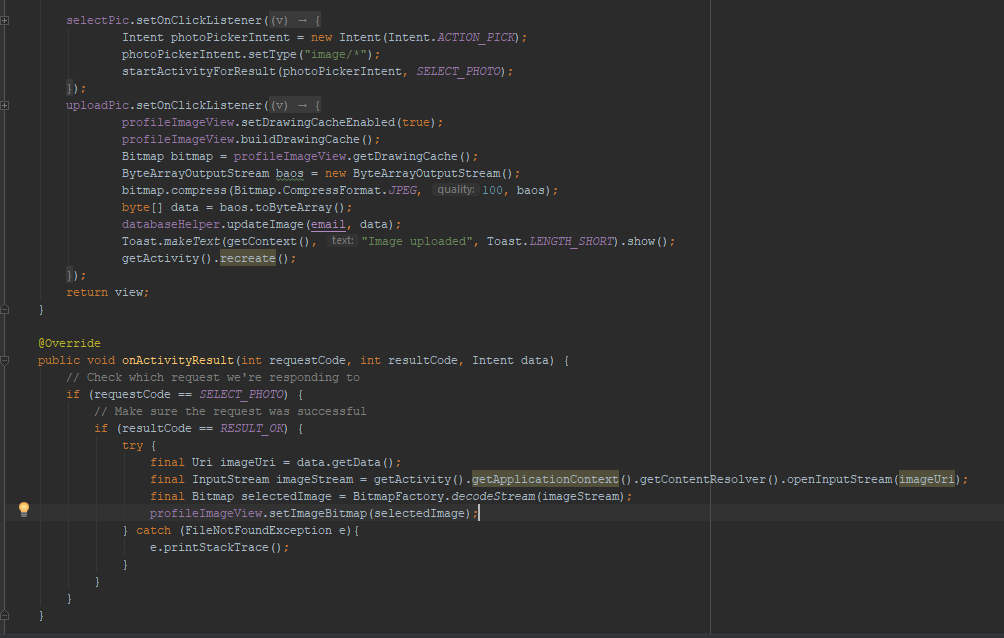


*Slika 6 – JSON*

Metod loadJSON se nalazi u svakom fragmentu skoro, jer su fragmenti NowPlayingFragment, PoplarMoviesFragment, TopRatedMoviesFragment, I logika same metode je da pošalje http get zahtev api-u, pokupi rezultate, parsira ih, pošalje u određeni adapter (movie ili trailer), odakle se dalje vrši ispisivanje tih podataka u željeni deo aplikacije (detailactivity, strana gde se prikazuju naslovi I slike).

Kod je poprilično jednostavan I ne treba previse detaljisati oko njega, jedina razlika koja se menja u svim fragmentima gore navedenim je apiService.METODA, u zavisnosti šta želimo da pokupimo od API-a.

Uglavnom, definišemo servis koji poziva iz klijenta URL, nadovezuje mu jedinstveni API Token u kome je prethodno već bilo reči, I onResponse, ukoliko je zahtev uspešan, filmovi se parsiraju (fetchuju) u listu, Išalju u odgovarajući adapter. Ukoliko dođe do nepredviđene greške I to će se prikazati.

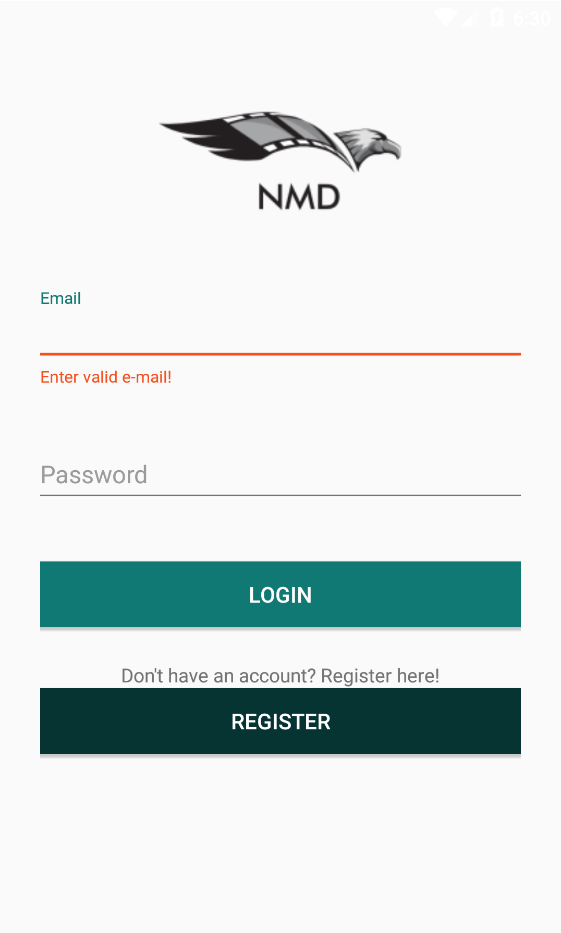
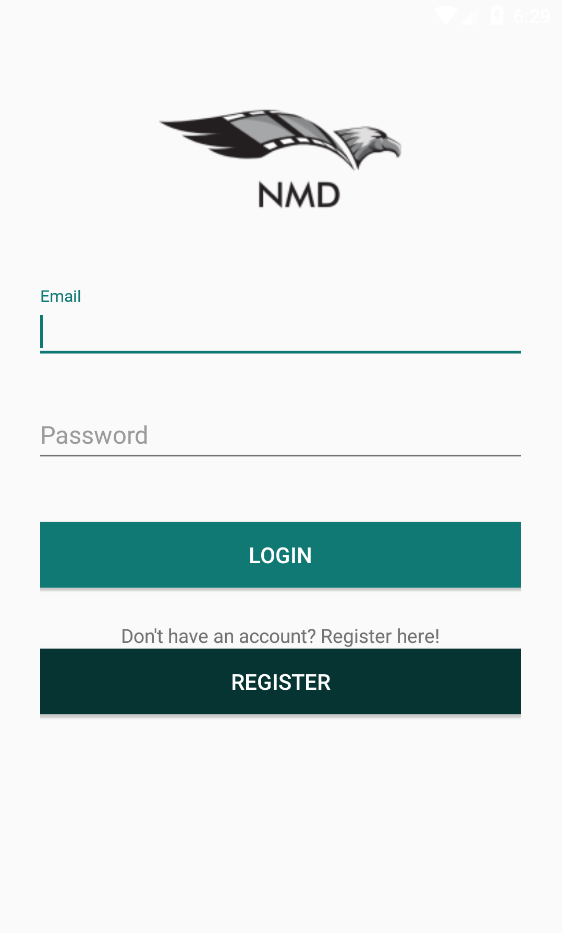
*Slika 7 – Profile Fragment*

Jedna od zanimljivih stvari je upload slike, gde korisnik na dugme Select Picture, putem intenta, mu se otvara galerija iz koje može izabrati sliku. Kada korisnik izabere sliku, ona će se postaviti (OnActivityResult). Ukoliko korisnik ne pritisne upload, prilikom promene aktivnosti ili fragmenta, stanje će se obrisati, I vratiće se prethodno podešena (ili default) slika. Ukoliko korisnik pritisne na upload, pravi se objekat Bitmap, čijom se metodom vrši kompresija slike, potom se prebacuje u ByteArray, I upisuje u bazu podataka tipa Blob. Potom se aktivnost recreate(), a u logici prikaza profile fragmenta I aktivnosti svaki put kada je korisnik otvori se vrši uzimanje slike iz baze ukoliko podatak u bazi nije == null.

Ostale klase se mogu videti u projektu, i metode su krajnje jednostavne za shvatiti, ili je logika većinom vezana za iscrtavanja, te nije potrebno detaljnije objašnjavati.

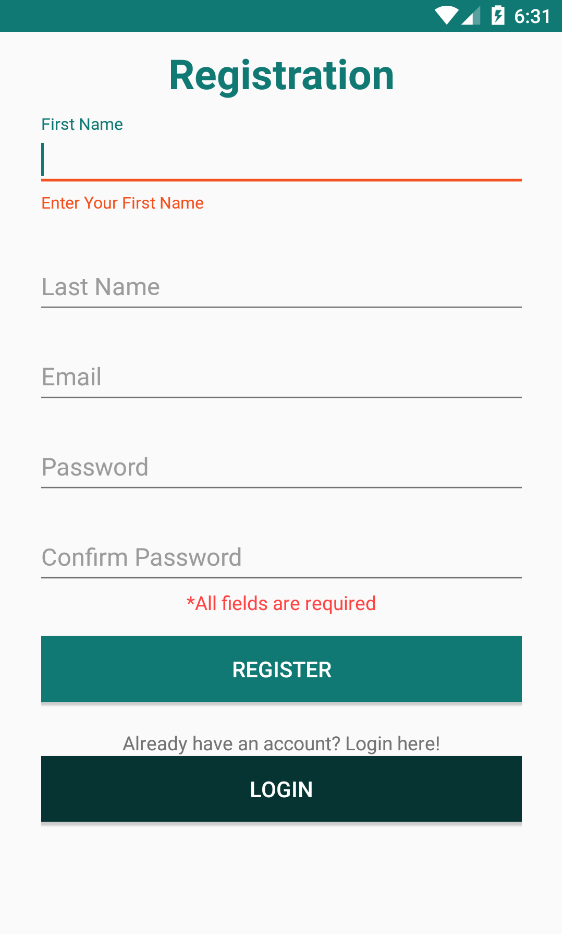
(Autor smatra da je to dizajnerski deo, i logika i lepota koda nije bitna, dokle god to izgleda prijatno i korisnik je zadovoljan).

# DEMONSTRACIJA



*Slika 8 – Login I Validacija*

Prva strana koju korisnik vidi kada otvori aplikaciju je slika 8, gde je potrebno uneti podatke koje je korisnik uneo prilikom registracije. Ukoliko korisnik ne unese validan email, uopste email, kao I lozinku, dobiće crveno upozorenje validacije sa desne slike.

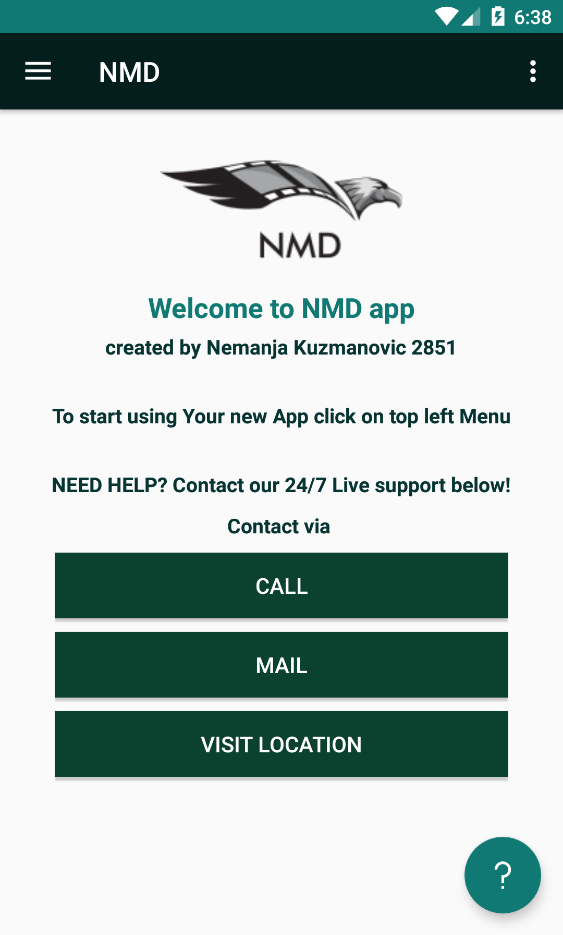


*Slika 9 – Registracija*

Kada korisnik otvori stranu sa registracijom, ukoliko već nema nalog, moraće popuniti sva polja, I takođe će dobiti validacionu grešku ukoliko nešto ne popuni, ili mail ne popuni u pravom formatu, ili se lozinke ne poklapaju.

Kada se korisnik registruje pojaviće mu se Snack back skroz dole, sa porukom da se uspesno registovao, I sada može pristupiti Login strain I ući u aplikaciju.

Register je nova aktivnost, I kada korisnik pritisne na login, ona se ne pauzira nego se ubija (finish()) zbog performansi I štednje memorije.

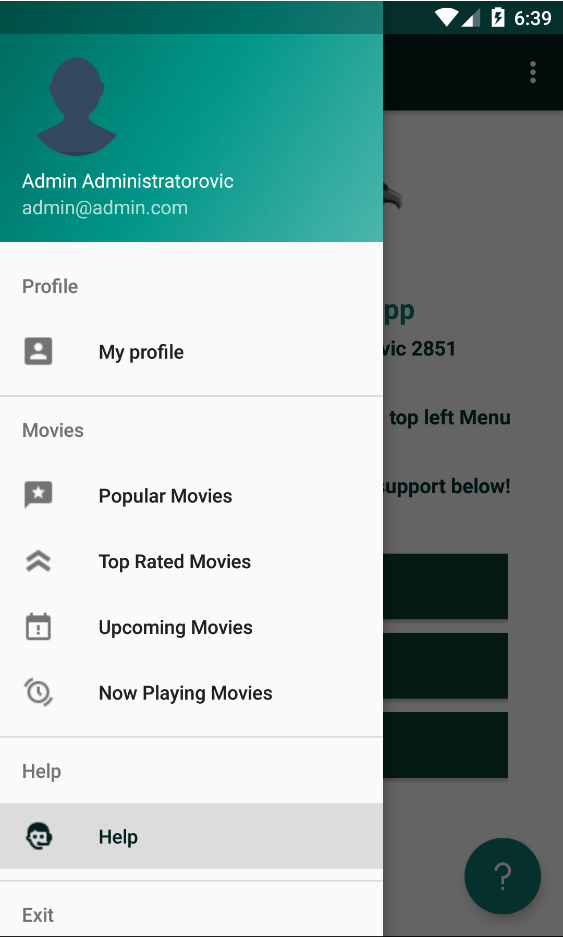


*Slika 10 – Početna strana help*

Početna strane je zapravo fragment help. Na njemu pored logoa aplikacije, postoje tri dugmeta, gde prvo upisuje broj telefona u call dialog (Action Dial) te se moće pozvati podrška ukoliko je potrebno. Dugme mail otvara prozor u kome korisnik može odabrati željeni mail klijent (gmail), gde će aplikacija sama popuniti mail korisnika, kome se šalje mail, subject, pozdravnu poruku, kao I potpis ulogovanog korisnika u vidu njegovog imena I prezimena, te korisniku samo ostaje da upiše problem.

Ukoliko korisnik želi lično da potraži pomoć ili konsultaciju, na dugme Visit Location, otvara se googlemaps lokacijski servisi sa svim mogućnostima (lokacija, kompas itd…), korisnik može videte našu lokaciju, a takođe ukoliko odabere neku drugu ili pritisne dugme desno gore da ga gsp locira, može u jednom kliku dobiti direkcije do naše kancelarije.

Sa leve strane (gore levo) korisnik otvara meni aplikacije I počinje da koristi aplikaciju. U svakom trenutku ukoliko mu hitno zatreba pomoć može pritisnuti desno dole znak pitanja I dobiće da izabere poziv ili mail kontakt. Ova opcija je dostupna uvek, kroz celu aplikaciju.



*Slika 11 – Main Menu*

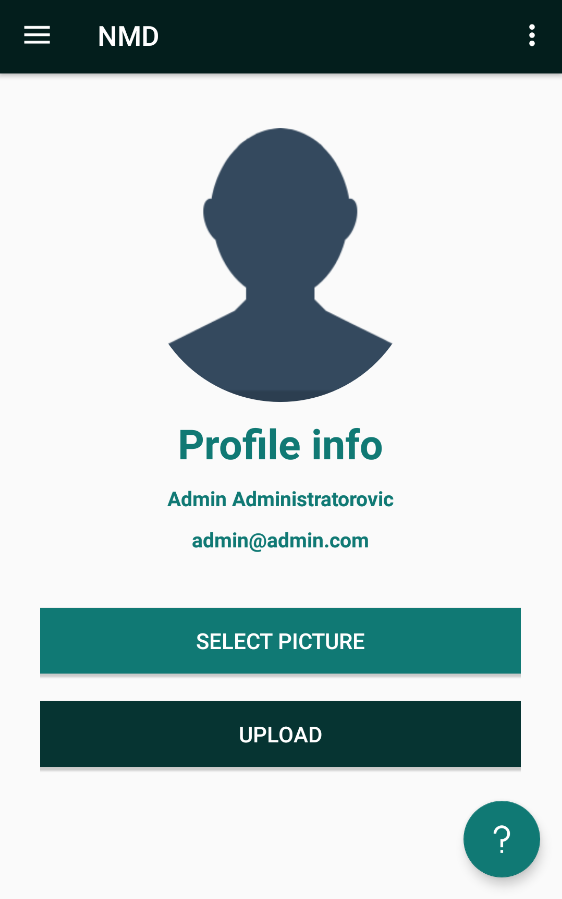
Korisnik navigira kroz aplikaciju putem menija, levo gore se prikazuje korisnikova slika iz baze podataka, ukoliko je podešena, ukoliko nije, prikazuje se podrazumevana. Ispod toga korisnik vidi svoje podešeno ime I prezime, kao I email adresu sa koje je ulogovan na nalog.

MyProfile, void korisnika na stranicu sa profilom korisnika gde može za sada videte podatke o sebi, I uploadovati sliku na system po želji.

Ukoliko korisnik odabere neki fragment iz kategorije Movies, on će se otvoriti I prikazati na ekranu.

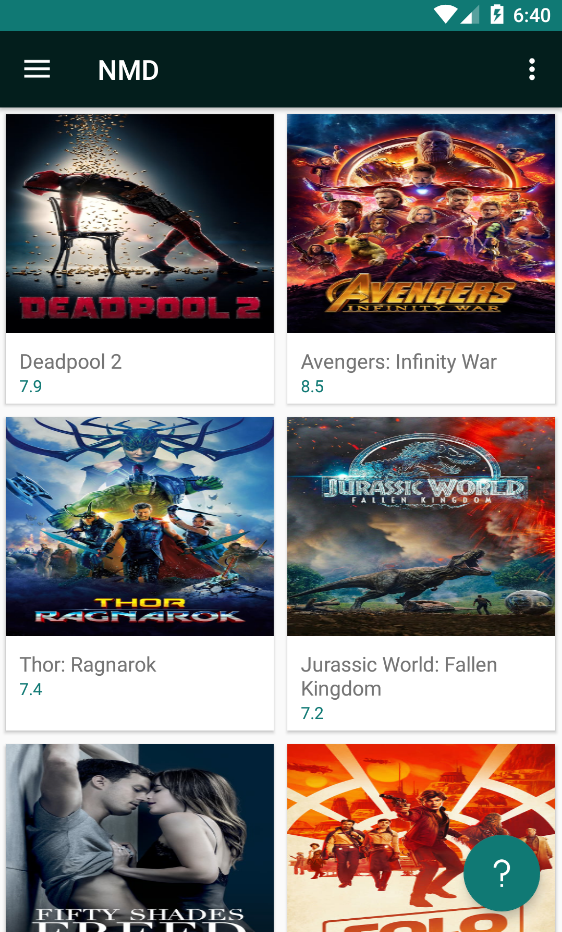
Korisnik moće skrolovati meni ka dole I pritisnuti logout da izađe iz aplikacije (poziva se finish() metoda, koja će ubiti mainactivity u kome je cela logika aplikaicje I iz koje se pozivaju potrebni fragmenti, pogledati kod u projektu).

Takođe korisnik ne mora pritisnuti na meni, nego može (gesture) prstom prevući sa leva na desno I meni će se otvoriti. Ukoliko korisnik više puta pritisne back dugme, dobiće dijalog da li stvarno želi da se izloguje iz aplikacije sa Yes I Cancel opcijama.



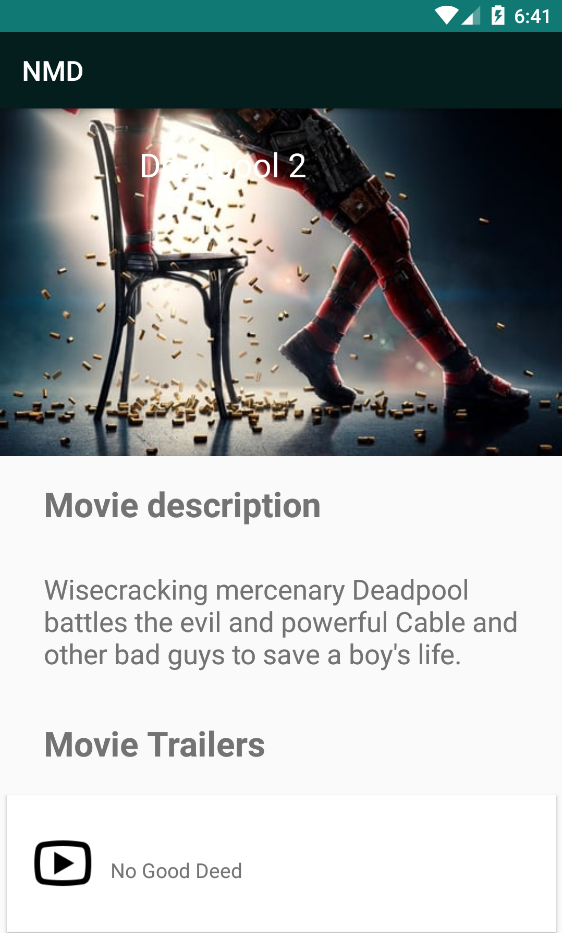
*Slika 12 – Profile Fragment*

Kao što je već bilo reči, korisnik na profile meniju, ima mogućnosti uploadovanja slike, to radi tako što pritisne select picture, te bira sliku, nakon što mu se otvori galerija. Na dugme upload potvrdiće ovu akciju, I slika će biti dalje u bazi podataka. Pored toga korisniku će se slika (kada klikne upload) takođe ažurirati u glavnom meniju aplikacije(poziva se metoda recreate gde se glavna mainactivity ponovo poziva u kojoj je metoda za getovanje slike iz baze, a sam meni je u glavnom aktivnosti pa se samim tim I on ponovo recreatuje, pogledati kod u projektu za detaljnije razumevanje).



*Slika 13 – Odabir kategije (Popular Movies)*

Kada korisnik odabere kategoiju iz menija, aplikacije će pokušati da pozove api I prikupi tražene podatke. Za to vreme korisnik dobija dijalog koji ga onemogućava da bilo šta pritisne na kome piše Fetching Movies... Kada se filmovi skinu I prikažu (nekoliko milisekundi, zavisi od brzine interneta) korisnik će dobiti ovakvu sliku. Dakle, dobija se slika (poster) filma, ocena korisnika sajta themoviedb.org, kao I naslov filma. Ukoliko korisnik prevuče ceo Recycle View ka dole pojaviće se refresh ikona (Swipe Container) koju kada pusti korisnik, ponovo će pokušati da učita filmove tj da ih refreshuje (zapravo se radi samo recreate kao da je korisnik izašao I ponovo ušao u deo aplikacije tj fragment).



*Slika 14 – Detalji filma*

Korisnik prvenstveno, kada otvori željeni film, dobija veliku sliku preko 80% ekrana (parallex) koju kada prevuče ka gore, ona se skupi I u gornjem meniju se samo prikazuje u jednoj traci naziv filma, ukoliko korisnik prevuče taj naziv ka dole, slika će se proširiti (naslov se expanduje I dobije se ponovo cela slika I obratno). Ispod toga, korisnik moće videte neke osnovne detalje o filmu, kao što je naziv filma, godina kada je izašao film, rejting filma, I kratak opis filma (synopsis). Na samom dnu korisnik dobija listu trailera ili videa za određeni film (minimum 1), koje omogućava API. Skrolovanjem ka dole, ukoliko ima više filmova, mogu se izlistati svi. Pritiskom na film, ukoliko korisnik ima youtube aplikaciju biće upitan da li želi preko nje da pusti trailer ili preko web browsera, ukoliko nema youtube aplikaciju, video će se pustiti preko browsera (korisnik bira preko kog ukoliko ima više).

# ZAKLJUČAK

Aplikacija koristi sve što je navedeno u zahtevima projekta, od aktivnosti, preko fragmenata, lokacijskih servisa, naprednih pogleda, naprednih servisa kao što su slanje mail-a, intentovi, baza podataka…

Dakle,sve što je rađeno kroz semester je uključeno u aplikaciju, I može se jasno videte nivo razumevanja samog predmeta, razvoja mobilnih aplikacija. Kroz razvoj aplikacije je dosta naučeno, pored rukovanja sa APIem, naučeno je korišćenje eksternih biblioteka, I integracija u android projekat, obnovljena su znanja stečena tokom semestra, I konačno, izrađena je aplikacija, spremna za potpisivanje I objavljivanje na market.

# LITERATURA

1. prof. dr Milićević, Vladimir. Predavanja i Vežbe iz Predmeta CS330 – Razvoj Mobilnih Aplikacija. Izdanje za nastavnu godinu 2017/2018. Beograd: Univerzitet Metropolitan 2017. Web. LAMS Univerziteta Metropolitan. Dostupno na:

<http://lms.metropolitan.ac.rs>.

1. Android Developers. Google, 2018. Web. Dostupno na: developer.android.com/.
2. <http://square.github.io/retrofit/>
3. <https://github.com/bumptech/glide>
4. <https://inthecheesefactory.com/blog/get-to-know-glide-recommended-by-google/en>
5. Stackoverflow