Univerzitet u Kragujevcu

Fakultet inženjerskih nauka

Programiranje mobilnih aplikacija

Dokumentacija za projektni zadatak

Tema:

Android aplikacija za povezivanje korisnika preko filmova

PROFESOR:

Prof. Dr. Vukašin Slavković

STUDENT:

Đorđe Karišić 657/2019

_____

Kragujevac, . . 2023.

Sadržaj

1. Uvod

Čovek u današnjici teži da reši svoje probleme, ili pronađe zabavu upotrebom uređaja i aplikacija napravljenih uz pomoć modernih tehnologija, koje su toliko napredovale da je gotovo nemoguće naći problem koji nije rešiv njihovom upotrebom. Jedan od glavnih faktora naglog napretka tehnologija jeste ljudska potreba. Pored raznih tipova aplikacija razvijenih po potrebi čoveka, potrebno je izdvojiti društvene mreže. Društvene mreže omogućavaju korisniku da se upozna i komunicira sa drugim ljudima na globalnom nivou- potpuno besplatno. Kako bi korisnik neke društvene mreže pronašao drugog korisnika koji bi bio kompatibilan sa njim, potrebno je da se na neki način pruži mogućnost uvida o korisničkim interesovanjima, profila i ličnosti. Svaka društvena aplikacija danas omogućuje korisniku da personalizuje svoj profil- da upiše neku informaciju o sebi, i da drugi korisnici imaju mogućnost pregleda istog, kako bi bolje upoznali tog korisnika i evektualno se „sprijateljili".

Aplikacija razvijena ovim projektom se bavi povezivanjem ljudi preko preferiranih filmova. Svaki korisnik ima mogućnost izlistavanja sa pretragom svih filmova, čuvanja omiljenih, pregleda profila drugih korisnika, praćenja drugih korisnika, I uvid o njihovim pratiocima i praćenjima. Profil svakog korisnika predstavlja njegova selekcija omiljenih filmova. U pretrazi, pored imena svakog korisnika, stoji broj njegovih omiljenih filmova kako bi se prikazala aktivnost svakog korisnika.

2. Potrebe realnog sistema

Sistem ove aplikacije je osmišljen kako bi bio dostupan svakome. Aplikacija se jednostavno omogućuje i pokreće, i intefejs je sam po sebi intuitivan, što omogućuje da ova aplikacija ne bude zahtevna po performansama i ne zahteva dodatne uređaje.

Aplikacija zahteva:

1. Mobilni telefon sa Android operativnim sistemom sa minimum verzijom 8.0 Oreo
2. Pristup internetu

Kao svaka društvena aplikacija, i ova aplikacija je potpuno besplatna. Korisnik jedino, kako bi koristio aplikaciju, mora kreirati nalog, pomoću kojeg će imati potpuni pristup korisničkom delu sistema.

Administrator je prethodno definisan i ne zahteva kreiranje naloga. Nakon login-a administratora, preko administrativnog panela ima jasan i pregledan uvid o svim filmovima i korisnicima.

Potreban operativni sistem, Android 8.0 Oreo je dostupan na uređajima od 2017. godine, tako da nije potrebno koristiti najsavremeniji mobilni telefon za pristup sistemu.

Kako bi korisnik mogao da čuva i dodaje filmove ili promeni svoj profil, potrebno je da ima konekciju sa serverom i bazom podataka, što zahteva pristup internetu.

Aplikacija ne koristi veliki protok interneta i podaci koji se šalju serveru za obradu su minimalni, tako da korišćenje aplikacije preko mobilnih podataka (provider interneta) ne predstavlja problem. Primer prethodno navedene karakteristike je mogućnost sistema da čuva enkodovane slike u base64 formatu lokalno, nakon prvobitnog učitavanja preko servera, u uslužnoj klasi namenjenoj čuvanju velikih podataka.

Potrebe sistema su minimalne, samim tim dostupan je svima.

3. Softverska realizacija sistema

Sistem da bi funkcionisao, potreban mu je softver – program. Pisanje programskog koda je urađeno unutar integrisanog razvojnog okruženja – Android Studio IDE. Svi potrebni softveri su besplatni i moguće ih je jednostavno preuzeti.

3.1. Konstruisanje šeme baze podataka i grupisanje podataka po strukturama

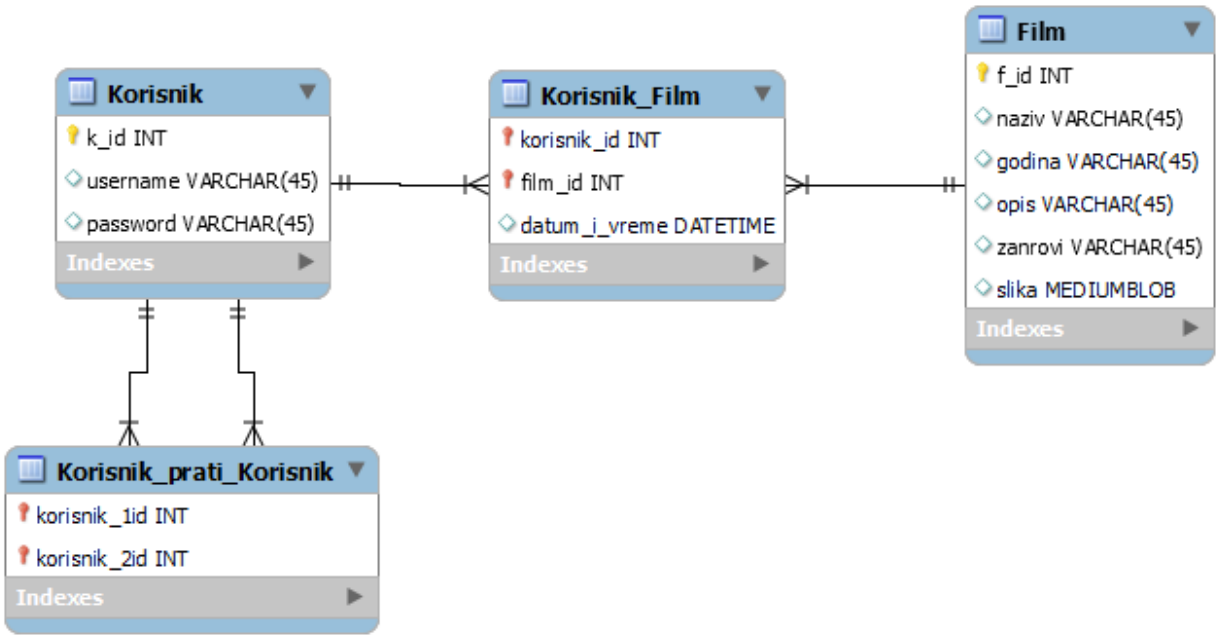
Kako bi se aplikacija na efikasan način projektovala potrebno je odrediti i grupisati potrebne podatke za korisnike, filmove, interakcije korisnika sa korisnikom, korisnika sa filmom.

Šema podataka bi trebalo da pokriva sledeće entitete:

1. Korisnik (korisničko ime, lozinka)
2. Film (naziv, godina izdavanja, opis, žanrovi, slika)

Za predstavljanje veze između korisnika sa filmova i drugim korisnicima:

1. Korisnik_Film(korisnik_id,film_id, datum_i_vreme)
2. Korisnik_prati_Korisnik(korisnik_1id,korisnik_2id)



Korisnik se identifikuje uz pomoć korisničkog imena i lozinke koji su jedinstveni.

Film se sastoji iz naziva, godine izdavanja, opisa, žanrova kojim pripada, slikom.

Kako bi se predstavili podaci koji govore o tome koji korisnik prati kojeg korisnika, koristi se tabela koja uzima dva identifikatora iz tabele Korisnik kako bi definisala uređen par {pratilac,prati}.

Za tabelu koja govori o tome koji film je koji korisnik označio, koristi se identična struktura, sa razlikom postojanja datuma označavanja, kako bi se moglo sortirati radi boljeg prikaza podataka korisniku.

Celokupna prethodno definisana struktura predstavlja konačnu bazu podataka.

Svaki entitet predstavlja složeni tip podataka i ne može se predstaviti primitivnim tipovima definisanim u okviru programskog jezika Java. Potrebno je koristiti objektno orjentisani pristup kako bi se klasifikovali tipovi podataka i predstavili entiteti kao objekti tih klasa.

3.1.1. Klasa DatabaseHelper

Klasa DatabaseHelper se bavi povezivanjem i korišćenjem SQLite baze podataka.

```
package com.example.myapplication;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.widget.Toast;

import java.text.SimpleDateFormat;
import java.util.Date;

public class DatabaseHelper extends SQLiteOpenHelper {

    private Context context;

    public static final String DATABASE_NAME = "Filmovi.db";
    public static final String FILMOVI = "Filmovi";
    public static final String F_ID = "ID";
    public static final String F_IME = "IME";
    public static final String F_GODINA = "GODINA";
    public static final String F_OPIS = "OPIS";
    public static final String F_ZANR = "ZANR";
    public static final String F_SLIKA = "SLIKA";

    public static final String USER = "User";
    public static final String U_ID = "ID";
    public static final String U_USERNAME = "USERNAME";
    public static final String U_PASSWORD = "PASSWORD";

    public static final String USER_FILM = "USER_FILM";
    public static final String UF_ID = "ID";
    public static final String UF_UID = "U_ID";
    public static final String UF_FID = "F_ID";
    public static final String UF_VREME = "VREME";  //prijatelji


    public static final String KORISNIK_PRATI = "Korisnik_Prati";
    public static final String KP_ID = "ID";
    public static final String KP_U1_ID = "U_ID1";
    public static final String KP_U2_ID = "U_ID2";
```

Od polja sadrži informacije o tabelama i bazi podataka.

```
public DatabaseHelper(Context context) {

    super(context, DATABASE_NAME, null, 2);
    this.context = context;
}
@Override
public void onCreate(SQLiteDatabase db) {
    db.execSQL("create table " + FILMOVI +
            "("+ F_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
            F_IME+ " TEXT," +
            F_GODINA+" TEXT,"+
            F_OPIS+ " TEXT," +
            F_ZANR+ " TEXT," +
            F_SLIKA+ " MEDIUMBLOB"
            + ")");

    db.execSQL("create table " + USER +
            " ("+ U_ID+ " INTEGER PRIMARY KEY AUTOINCREMENT, " +
            U_USERNAME+" TEXT, "+
            U_PASSWORD+" TEXT"
            + ")");
    db.execSQL("create table " + USER_FILM +
            " ("+ UF_ID+ " INTEGER PRIMARY KEY AUTOINCREMENT , " +
            " "+ UF_FID+ " INTEGER , " +
            UF_UID+" INTEGER, "+
            " "+ UF_VREME+ " DATETIME , " +
            "FOREIGN KEY ("+UF_FID+") REFERENCES Filmovi("+F_ID+") ON DELETE
CASCADE, "+
            "FOREIGN KEY ("+UF_UID+") REFERENCES User("+U_ID+") ON DELETE
CASCADE"
            + ")");

    db.execSQL("create table " + KORISNIK_PRATI +
            " ("+ KP_ID+ " INTEGER PRIMARY KEY AUTOINCREMENT , " +
            " "+ KP_U1_ID+ " INTEGER , " +
            KP_U2_ID+" INTEGER, "+
            "FOREIGN KEY ("+KP_U1_ID+") REFERENCES User("+U_ID+") ON DELETE
CASCADE, "+
            "FOREIGN KEY ("+KP_U2_ID+") REFERENCES User("+U_ID+") ON DELETE
CASCADE"
            + ")");
    db.execSQL("PRAGMA foreign_keys=ON;");
}
```

Konstruktor kreira bazu podataka i postavlja kontekst na kontekst aktivnosti koja ga poziva.

Metoda onCreate kreira šemu baze podataka.

```
public boolean insertFilm(String table,String[] vals,byte[] img){
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues contentValues = new ContentValues();
    contentValues.put(F_IME, vals[0]);
    contentValues.put(F_GODINA, vals[1]);
    contentValues.put(F_OPIS, vals[2]);
    contentValues.put(F_ZANR, vals[3]);
    contentValues.put(F_SLIKA,img);
    long result = db.insert(table, null, contentValues);

    if (result == -1)
        return false;
    else {
        return true;
    }
}
public boolean insertData(String table, String[] vals) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues contentValues = new ContentValues();
    if(table.equals(FILMOVI)){
        contentValues.put(F_IME, vals[0]);
        contentValues.put(F_GODINA, vals[1]);
        contentValues.put(F_OPIS, vals[2]);
        contentValues.put(F_ZANR, vals[3]);
    }
    else if(table.equals(USER)){
        contentValues.put(U_USERNAME, vals[0]);
        contentValues.put(U_PASSWORD, vals[1]);
    }
    else if(table.equals(USER_FILM)){
        contentValues.put(UF_FID, vals[0]);
        contentValues.put(UF_UID, vals[1]);
        contentValues.put(UF_VREME, new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss").format(new Date()));
    }
    else if(table.equals(KORISNIK_PRATI)){
        contentValues.put(KP_U1_ID, vals[0]);
        contentValues.put(KP_U2_ID, vals[1]);
    }
    long result = db.insert(table, null, contentValues);

    if (result == -1)
        return false;
    else {
        return true;
    }
}
```

Funkcija insertFilm se bavi unosom filma u bazu.

Funkcija insertData koristi generalizovani pristup unosa tako što je moguće uneti ime tabele i vrednosti koje je potrebno uneti.

Obe funkcije imaju Boolean kao povratnu vrednost, koja ukazuje na to da li je funkcija uspešno izvršena.

```
public Cursor execRawQuery(String query){
    SQLiteDatabase db = this.getWritableDatabase();
    Cursor cursor = db.rawQuery(query,null);

    return cursor;
}
public Integer getIDOfUser(String username){
    Integer id;
    SQLiteDatabase db = this.getWritableDatabase();
    String query = "SELECT * FROM User WHERE USERNAME ='"+username+"'";
    Cursor cursor = db.rawQuery(query,null);
    if(cursor.moveToFirst()){
        id=cursor.getInt(0);
    }
    else{
        return -1;
    }
    return id;
}

public boolean verifyUser(String un,String pw){
    SQLiteDatabase db = this.getWritableDatabase();
    String query = "SELECT * from User WHERE USERNAME='"+un+"' AND
PASSWORD='"+pw+"'";
    Cursor cursor = db.rawQuery(query,null);
    if(cursor.moveToFirst()){
        return true;
    }
    return false;
}
public Integer deleteUser(String id){
    SQLiteDatabase db = this.getWritableDatabase();

    if(db.delete("User", "ID=?", new String[]{id})!=-1){
        db.delete("USER_FILM","U_ID=?",new String[]{id});
        return db.delete("Korisnik_Prati","U_ID1=? OR U_ID2=?",new
String[]{id,id});
    }
    return -1;
}
```

Funkcija execRawQuery izvršava query koji programer može u potpunosti definisati.

getIDOfUser vraća korisnički ID na osnovu korisničkog imena.

verifyUser vraća boolean vrednost koja ukazuje da li postoji korisnik sa prosleđenim argumentima koji označuju username i password.

deleteUser se bavi brisanjem korisnika čiji ID je prosleđen kao argument iz svih tabela u kojima može postojati.

```
public Integer deleteData (String table, String where, String what){

    SQLiteDatabase db = this.getWritableDatabase();
    return db.delete(table, where+"=?", new String[]{what});
}
public Cursor getData(String table, String clause) {
    SQLiteDatabase db = this.getWritableDatabase();
    Cursor res;
    if(clause.equals("")) {
        res = db.rawQuery("select * from " + table, null);
    }
    else{
        res = db.rawQuery("select * from " + table + " WHERE " +
clause,null);
    }
    return res;
}
public String getNumOfFollowers(String id){
    String num="";
    SQLiteDatabase db = this.getWritableDatabase();
    Cursor res;
    String query = "select COUNT(*) from Korisnik_Prati WHERE U_ID2="+id;
    res = db.rawQuery(query,null);
    if (res.getCount() == 0) {
        //Toast.makeText(ProfilActivity.this, "Nema podataka",
Toast.LENGTH_LONG).show();
        num = "0";
    } else {
        if (res.moveToFirst()) {
            num = res.getString(0);
        }
    }
    return num;
}

public String getNumOfFollowing(String id){
    String num="";
    SQLiteDatabase db = this.getWritableDatabase();
    Cursor res;
    String query = "select COUNT(*) from Korisnik_Prati WHERE U_ID1="+id;
    res = db.rawQuery(query,null);
    if (res.getCount() == 0) {
        //Toast.makeText(ProfilActivity.this, "Nema podataka",
Toast.LENGTH_LONG).show();
        num = "0";
    } else {
        if (res.moveToFirst()) {
            num = res.getString(0);
        }
    }
    return num;
}
```

Funkcije za generalizovano brisanje podataka, vraćanje podataka, i brojanje praćenja i pratioca na osnovu korisničkog ID-ja

```
public Cursor getAllData(String table) {
    SQLiteDatabase db = this.getWritableDatabase();
    Cursor res;
    res = db.rawQuery("select * from " + table, null);
    return res;
}
public void updateUser(String id,String un,String pw){
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues cv = new ContentValues();
    cv.put(U_USERNAME, un);
    cv.put(U_PASSWORD, pw);
    if(un.length()<1 || pw.length()<1){
        Toast.makeText(context, "Korisničko ime i lozinka moraju imati barem
jedan karakter.", Toast.LENGTH_SHORT).show();
        return;
    }
    long res = db.update(USER, cv, "ID=?", new String[]{id});
    if (res == -1) {
        Toast.makeText(context, "Neuspesno", Toast.LENGTH_SHORT).show();
    } else {
        Toast.makeText(context, "Uspesno azurirano",
Toast.LENGTH_SHORT).show();
    }
}
public void updateData(String id,String ime,String godina,String opis,String
zanr,byte[] img){
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues cv = new ContentValues();
    cv.put(F_IME, ime);
    cv.put(F_GODINA, godina);
    cv.put(F_OPIS, opis);
    cv.put(F_ZANR, zanr);
    cv.put(F_SLIKA, img);
    long res = db.update(FILMOVI, cv, "id=?", new String[]{id});
    if (res == -1) {
        Toast.makeText(context, "Neuspesno", Toast.LENGTH_SHORT).show();
    } else {
        Toast.makeText(context, "Uspesno azurirano",
Toast.LENGTH_SHORT).show();
    }
}
void deleteData(String id1,String id2){
    SQLiteDatabase db = this.getWritableDatabase();
    long res = db.delete( USER_FILM,"U_ID=? AND F_ID=?",new
String[]{id1,id2});
    if (res == -1) {
        Toast.makeText(context, "Neuspesno", Toast.LENGTH_SHORT).show();
    } else {
        Toast.makeText(context, "Uspesno obrisano",
Toast.LENGTH_SHORT).show();
    }
}
```

Funkcije za vraćanje svih podataka iz baze, generalizovano ažuriranje i ažuriranje korisnika, kao i brisanje sačuvanog filma od strane prosleđenog korisnika.

```
boolean checkIfFollowing(String id1,String id2){
    SQLiteDatabase db = this.getWritableDatabase();
    String query="SELECT * FROM "+KORISNIK_PRATI+ " WHERE U_ID1="+id1+" AND
U_ID2="+id2;
    long res = db.rawQuery(query,null).getCount();
    if (res == 0) {
        return false;
    } else {
        return true;
    }
}
void deleteFollowing(String id1,String id2){
    SQLiteDatabase db = this.getWritableDatabase();
    long res = db.delete( KORISNIK_PRATI,"U_ID1=? AND U_ID2=?",new
String[]{id1,id2});
    if (res == -1) {
        Toast.makeText(context, "Neuspesno", Toast.LENGTH_SHORT).show();
    } else {
        Toast.makeText(context, "Uspesno obrisano",
Toast.LENGTH_SHORT).show();
    }
}
void deleteRow(String id) {
    SQLiteDatabase db = this.getWritableDatabase();
    long res = db.delete(FILMOVI, "id=?", new String[]{id});
    if (res == -1) {
        Toast.makeText(context, "Neuspesno", Toast.LENGTH_SHORT).show();
    } else {
        Toast.makeText(context, "Uspesno obrisano",
Toast.LENGTH_SHORT).show();
    }
}
boolean checkIfUserLikedMovie(String uid,String mid){
    Toast.makeText(context, uid+mid, Toast.LENGTH_SHORT).show();
    SQLiteDatabase db = this.getWritableDatabase();
    String query="SELECT * FROM "+USER_FILM+ " WHERE U_ID="+uid+" AND
F_ID="+mid;
    long res = db.rawQuery(query,null).getCount();
    if (res == 0) {
        return false;
    } else {
        return true;
    }
}
```

Provera da li korisnik sa ID1 prati korisnika ID2, brisanje praćenja, brisanje jednog filma i provera da li je korisnik označio određeni film.

3.1.2. Klasa Korisnik

Klasa Korisnik sadrži osnovne operacije za manipulaciju objektom korisnika.

```
public class Korisnik {
    private DatabaseHelper DB;
    private String username;
    private String password;

    public Korisnik(String username, String password) {
        this.username = username;
        this.password = password;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

Sadrži Getter-e, Setter-e i konstruktor. Od polja ima bazu, username i password.

3.1.3. Klasa Film

Klasa Film sadrži osnovne operacije za manipulaciju objektom filma, poput klase Korisnik.

```
public class Film {
    private String ime;
    private String godina;
    private String opis;
    private String zanrovi;
    private byte[] slika;

    public Film(String ime, String godina, String opis, String zanrovi,
byte[] slika) {
        this.ime = ime;
        this.godina = godina;
        this.opis = opis;
        this.zanrovi = zanrovi;
        this.slika = slika;
    }
    public String getIme() {
        return ime;
    }
    public void setIme(String ime) {
        this.ime = ime;
    }
    public String getGodina() {
        return godina;
    }
    public void setGodina(String godina) {
        this.godina = godina;
    }
    public String getOpis() {
        return opis;
    }
    public void setOpis(String opis) {
        this.opis = opis;
    }
    public String getZanrovi() {
        return zanrovi;
    }
    public void setZanrovi(String zanrovi) {
        this.zanrovi = zanrovi;
    }
    public byte[] getSlika() {
        return slika;
    }
    public void setSlika(byte[] slika) {
        this.slika = slika;
    }
}
```

Konstruktor, Getter-i, Setter-i. Polja ime, godina, opis, zanrovi i slika.

3.2. Prikaz podataka na ekranu korisnika

Za prikaz podataka na korisničkom ekranu korišćen je RecyclerView, koji omogućava „custom“ prikaz podataka u listama. Potrebno je proći samo kroz funkcionalnost svakog adaptera, bez objašnjavanja koda.

3.2.1. CustomAdapter

Klasa CustomAdapter se bavi definisanjem prikaza filma na administrator panelu. Prikazuje informacije o filmu i omogućava administratoru da na klik elementa liste koji predstavlja jedan film pristupi aktivnosti koja omogućava brisanje i ažuriranje filma.

3.2.2. KorisnikAdapter

Klasa KorisnikAdapter definiše pristup prikazivanju postojećih korisnika u listi. Postoji više opcija prikaza, ukoliko je potrebno prikazati administratoru korisnike, prikazaće se informacije o korisniku i mogućnost klika na element kako bi se obezbedilo ažuriranje ili brisanje izabranog korisnika, ukoliko je potrebno prikazati korisniku koji želi da vidi sve korisnike koje ne prati, imaće mogućnost praćenja i pregleda profila. Ukoliko korisnik pregleda svoje pratioce, imaće mogućnost pregleda njihovih profila, i u zavisnosti da li ih prati, dugme za odpratiti/zapratiti datog korisnika. Ukoliko gleda korisnike koje prati, korisnik ima mogućnost brisanja praćenja i prikazivanja profila.

3.2.3. ObjaveAdapter

Klasa ObjaveAdapter sadrži definiciju prikaza glavnog feed-a, to jest, prikaz aktivnosti kojoj korisnik pristupa kako bi video objave ljudi koje on prati. Korisnik ima mogućnost beleženja ili brisanja filma iz omiljenih filmova koji je korisnik kojeg on prati objavio.

3.2.4. PrikazAdapter

Ova klasa se bavi prikazom filmova i omogućava brisanje ili čuvanje filma. U zavisnosti od toga gde se prikazuje, može imati različite funkcionalnosti. U koliko se dati film sadrži u sačuvanim, dugme za brisanje će biti aktivno, u suprotnom, dugme za čuvanje je aktivno.

3.3. Uslužne klase

Od uslužnih klasa, aplikacija sadrži sledeće:

1. DbBitmapUtility
2. ExtendedDataHolder

3.3.1. DbBitmapUtility klasa

Ova uslužna klasa se bavi operacijama enkodovanja i dekodovanja slike u ili iz base64 formata.

Sadrži dve metode, getBytes i getImage koji odgovaraju enkodovanju I dekodovanju slike, respektivno.

3.3.2. ExtendedDataHolder klasa

Ova klasa dopunjava propust getExtra metode Intent-a. Propust je to što slika u formatu Base64 zahteva veću količinu memorije nego što je dozvoljeno da se pošalje putem intenta, Ova klasa sadrži metode identične Intent klasi, samo što nema definisano ograničenje.

4. Grafički prikaz sistema, definisanje stranica i menija

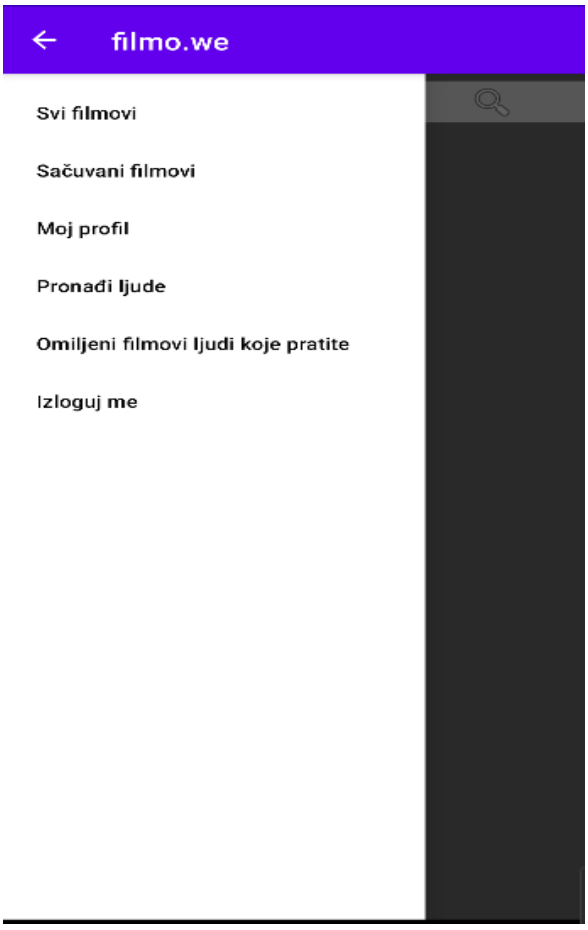
U ovoj sekciji biće prikazane slike dizajna aplikacije. Dizajn aplikacije je podeljen u više sekcija:

1. Meniji
2. Stranice
3. Kartice za prikaz u listama
4. Ikonice

4.1. Meniji

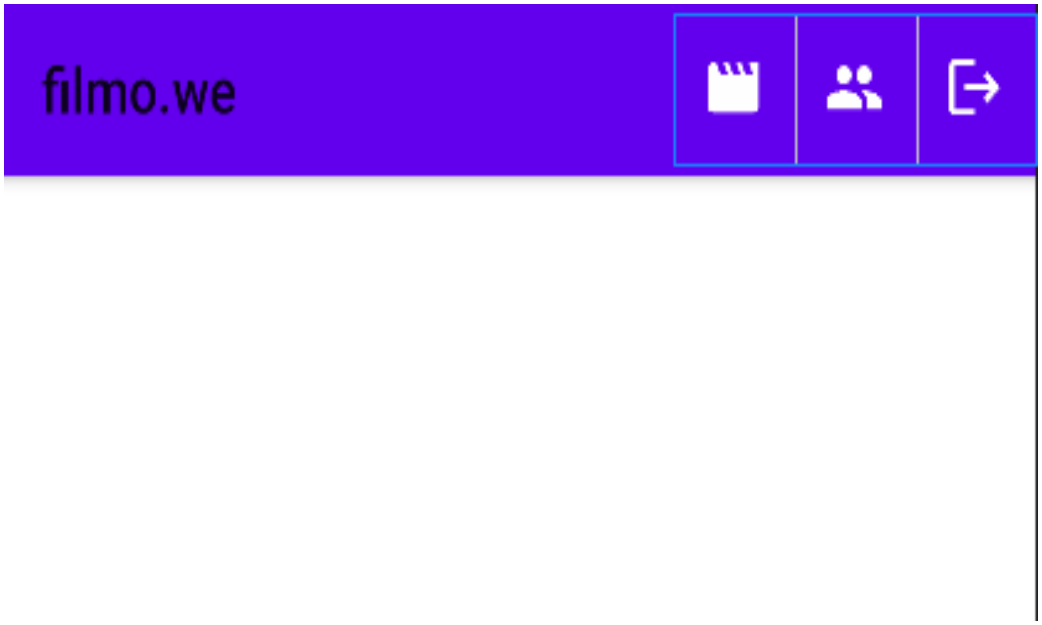
Postoje dva različita menija, jedan za navigaciju korisnika, drugi za navigaciju administratora.

Meni za navigaciju korisnika je smešten sa leve strane aplikacije, može se proširiti i otkriti više opcija za redirekciju. Izbor ovog dizajna je definisan postojanjem više mogućnosti stranica izbora korisnika.



Ovaj meni je prikazan na svakoj stranici korisničkog interfejsa.

Meni napravljen za administratora je jednostavniji, kako bi olakšali administratoru navigaciju sistemom.



Prva opcija prikazuje listu filmova.

Druga opcija prikazuje listu korisnika.

Treća opcija vrši operaciju logout-a i otvara stranicu za login.

Jednostavan i intuitivan dizajn omogućava administratoru da lako navigira kroz svoj interfejs ove aplikacije.

Ovaj meni je prikazan na svakoj stranici administratorovog interfejsa.

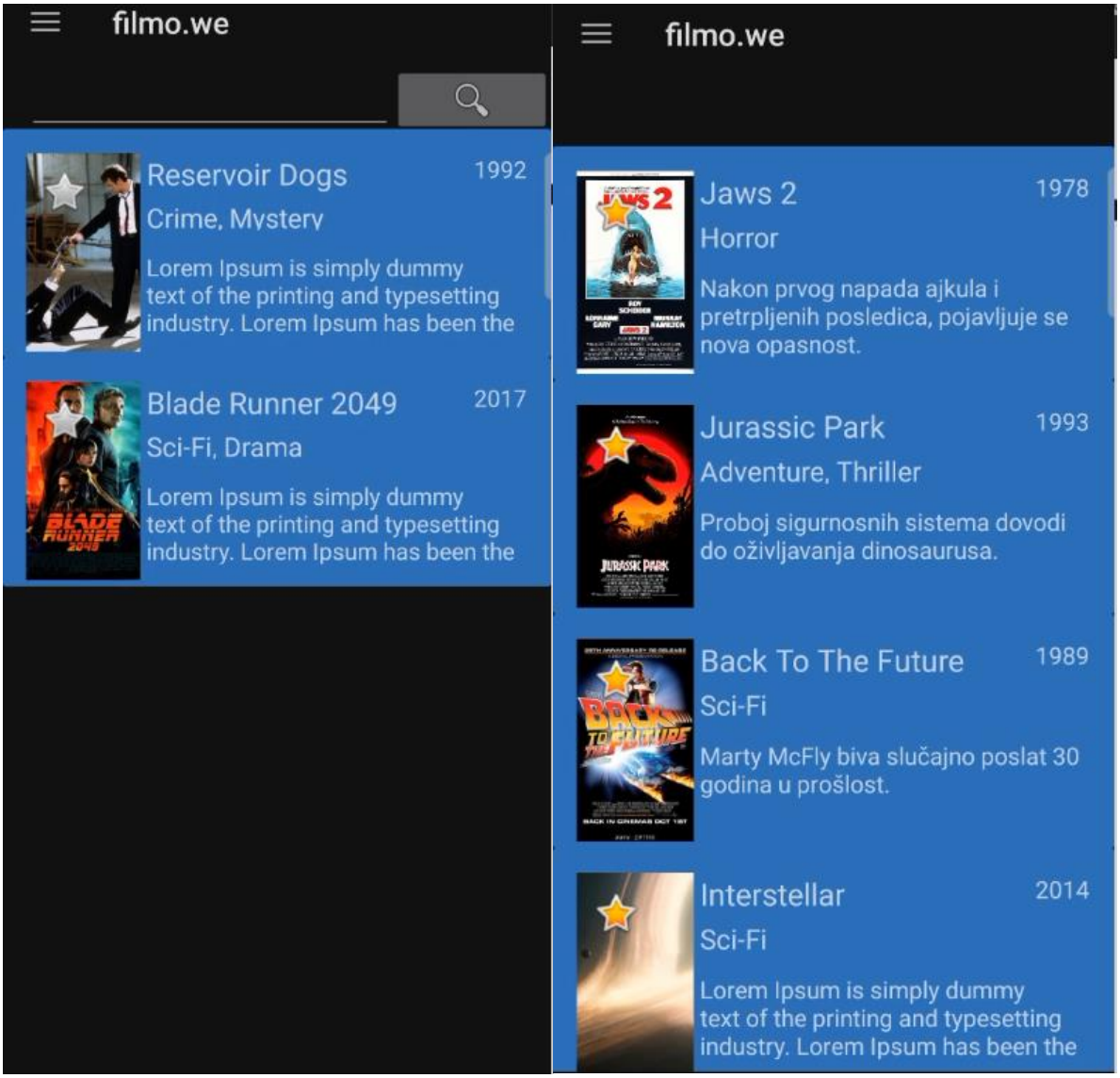
4.2. Stranice interfejsa za registraciju i login

Stranice za registraciju i login sadrže formu koje bi korisnici trebalo da popune kako bi pristupili sajtu. Forma za registraciju vrši upis u bazu, dok forma za login vrši čitanje i proveru validnosti podataka iz baze.

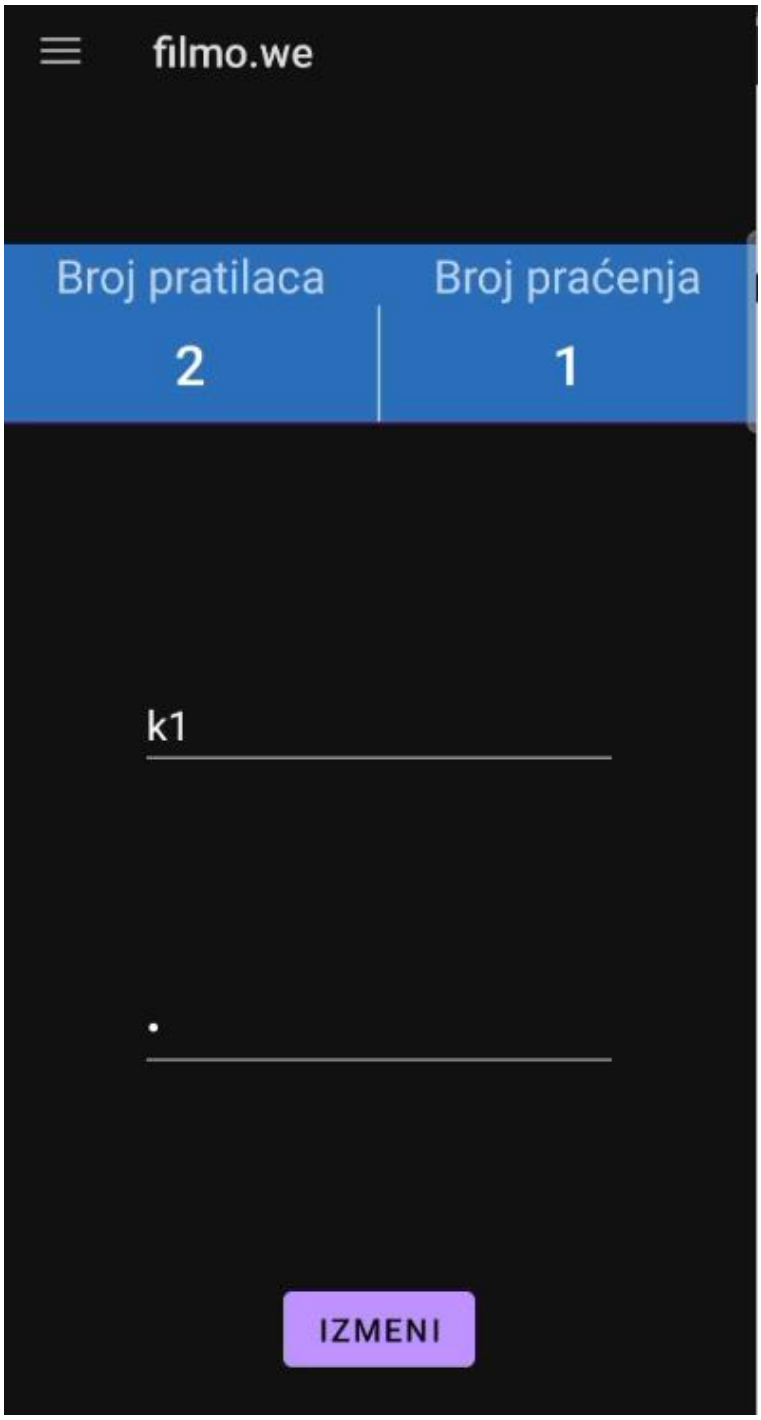

filmo.we
Registracija
FILMO.WE
Korisnicko ime
Lozinka
Ponovite lozinku
POTVRDI
LOGIN STRANA
filmo.we
Login
FILMO.WE
Korisničko ime
Lozinka
POTVRDI
STRANA ZA REGISTRACIJU

4.3. Dizajn stranica korisničkog interfejsa

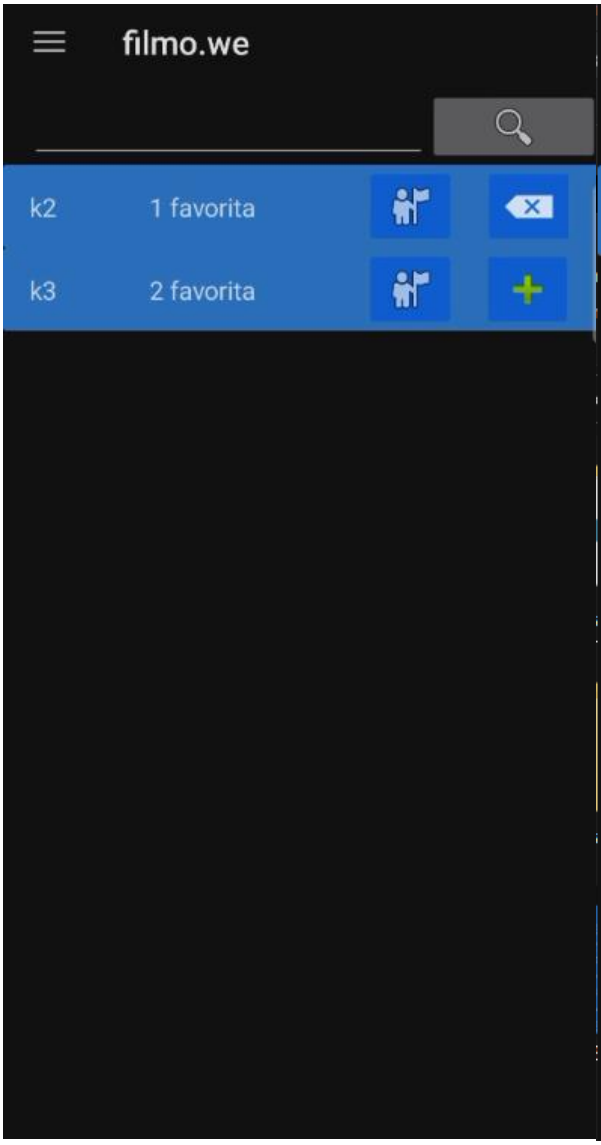
Slike ispod predstavljaju prikaz stranica „Svi filmovi“ i „Omiljeni filmovi“. Klik na praznu zvezdicu očitava dati film kao omiljeni, dok klik na punu zvezdicu briše iz omiljenih. Stranica „Svi filmovi“ prikazuje isključivo filmove koje korisnik nije označio kao omiljene, dok „Omiljeni filmovi“ prikazuje samo filmove označene kao omiljeni. Mogućnost pretrage svih filmova.



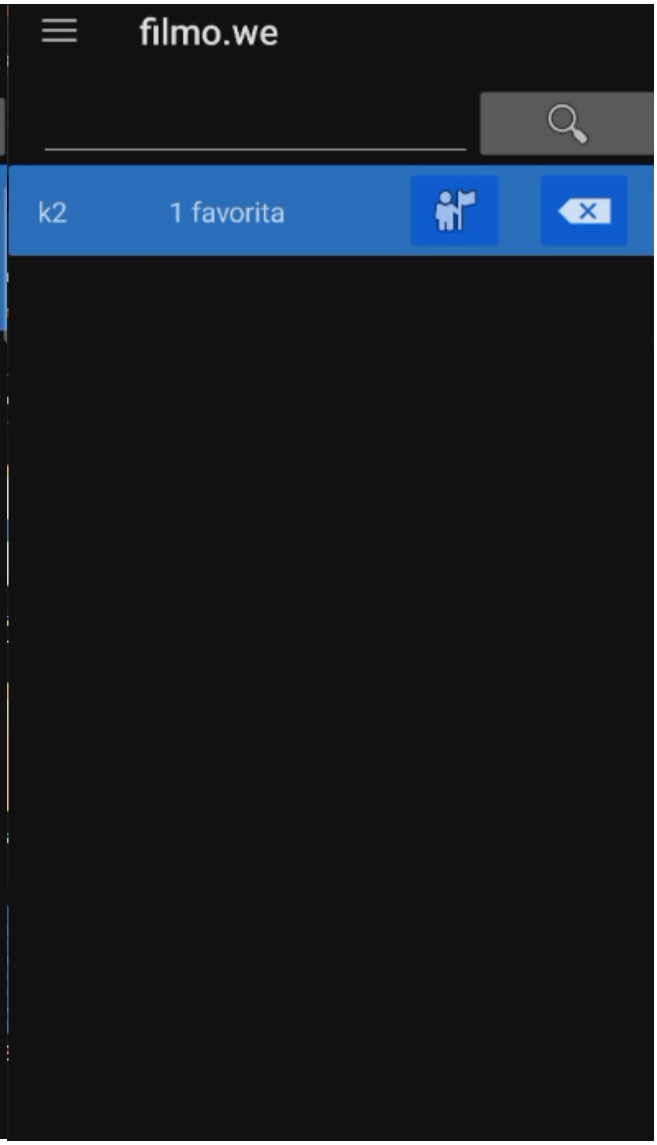
Stranica ispod omogućava korisniku da promeni korisničko ime ili lozinku, takođe, pruža informacije o broju pratilaca i broju praćenja. Klikom na dugmad „Broj pratilaca“ i ,,Broj praćenja“ korisnik biva prosleđen na stranicu koja prikazuje o kojim korisnicima se radi.



Slike ispod prikazuju pratioce i praćenja korisnika, sa pretragom, respektivno.

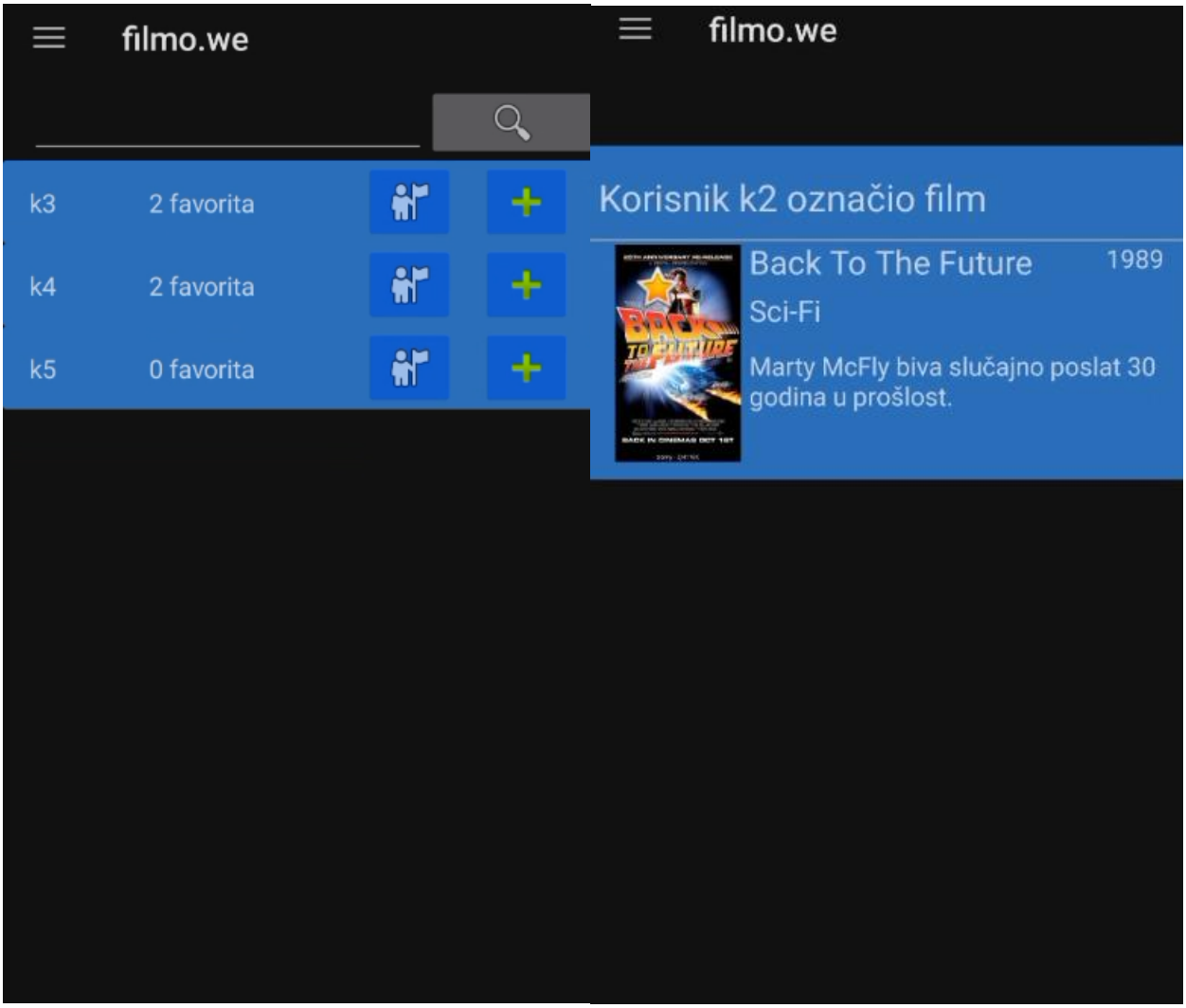


Opcije:Zaprati, odprati i pregledaj profil



Opcije: Odprati, pregledaj profil

Prikaz svih korisnika sa opcijama prikaza profila i praćenja sa pretragom, i prikaz aktivnosti tj. Sačuvanih filmova korisnika koje ulogovani korisnik prati, sa opcijom čuvanja prikazanog filma.

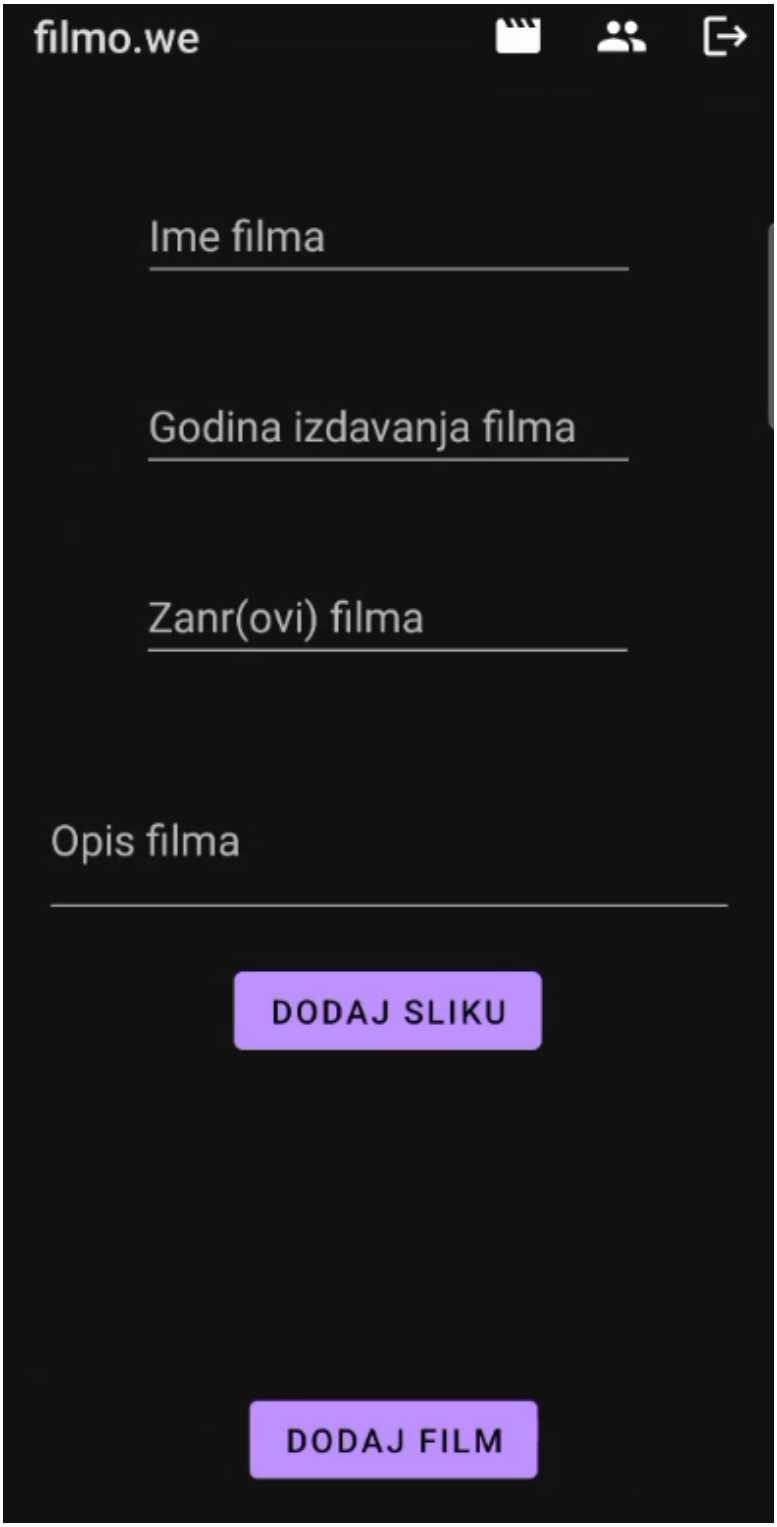


Ukoliko je film sačuvan, biće prikazana opcija za brisanje iz omiljenih filmova.

4.4. Dizajn stranica administratorskog interfejsa

Stranica administratorskog interfejsa moraju biti jednostavne za korišćenje kako bi se olakšao posao administratoru.

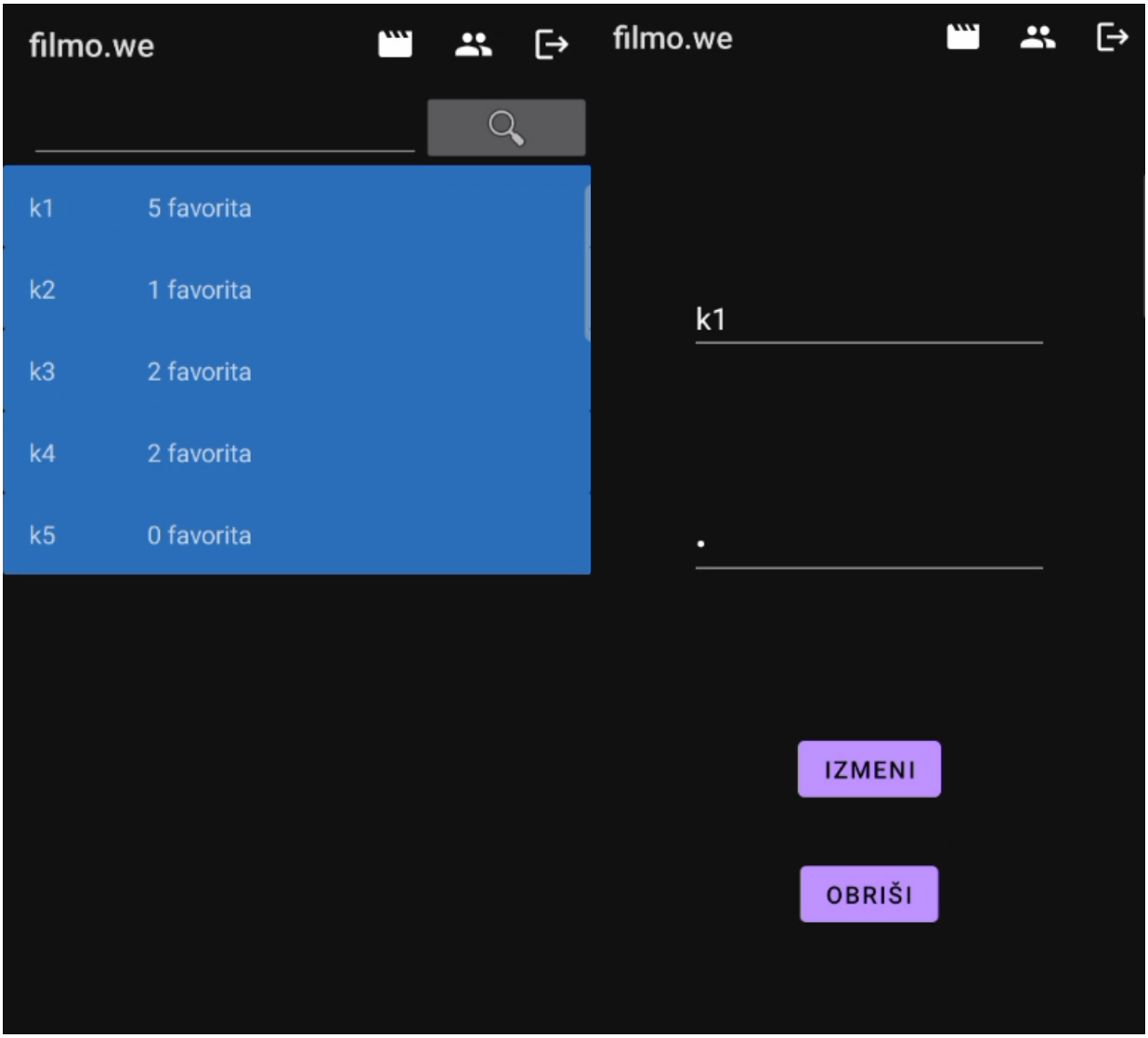
Slika ispod daje uvid administratoru o svim postojećim filmovima, klikom na „+“ dugme, administrator može dodati novi film.



Prikaz stranice za dodavanje novog filma.

Opcija „Dodaj sliku“ pokreće filtriranu galeriju koja priakzuje samo PNG i JPG slike iz koje korisnik može odabrati sliku.

Administrator klikom na drugu opciju u meniju biva redirektovan na stranicu koja prikazuje listu svih korisnika sa mogućnošću pretraživanja. Administrator klikom na neki red liste dobija mogućnost uređivanja ili brisanja korisnika čiji username se nalazi u prvoj koloni reda na koji je kliknuto.



5. Literatura

https://developer.android.com/develop/ui/views/theming/look-and-feel

Dizajn aplikacije

https://developer.android.com/reference/androidx/recyclerview/widget/RecyclerView.Adapter

Prikaz stavki

https://developer.android.com/develop/ui/views/graphics/vector-drawable-resources

Ikonice

https://developer.android.com/docs

Sveukupna literatura