SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Kristijan Počta

NAPREDNA BAZA PODATAKA ZA ADMINISTRACIJU KORISNIČKIH RAČUNA

DIPLOMSKI RAD

Varaždin, 2016.

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Kristijan Počta
Redoviti student
Broj indeksa: 43561/14-R
Smjer: Organizacija poslovnih sustava
Diplomski studij

NAPREDNA BAZA PODATAKA ZA ADMINISTRACIJU KORISNIČKIH RAČUNA

DIPLOMSKI RAD

Mentor:

Doc. dr. sc. Markus Schatten

Varaždin, rujan 2016.

Sadržaj

1. UVOD

Čovjek je oduvijek imao potrebu za pohranom informacija. Prije računalnog „boom“-a oblici pohrana su bili uglavnom papirnatog oblika. Samim time pohrana tih zapisa je zahtijevala dosta prostora, a pretraživanje podataka dosta vremena. U današnjem, računalnom dobu, za to koristimo baze podataka koje čine pohranu i pretragu podataka nemjerljivo bržom i jednostavnijom. Kroz ovaj rad razradit će se aktivne, temporalne, poopćene te objektno-relacijske baze podataka koje će se koristiti uz tradicionalne relacijske baze podataka. U sklopu ovog rada izradit će se baza podataka te aplikacija koja će tu bazu podataka koristiti. Kao sustav za upravljanje bazom koristit će se PostgreSQL. PostgreSQL je moćan sustav otvorenog koda za upravljanje objektno-relacijskim bazama podataka (PostgreSQL, 2016). Aplikacija naziva „RentalVideo“ biti će namijenjena za videoteke te će, koristeći navedene baze podataka, omogućiti lakše upravljanje videotekom te korisnicima koji koriste usluge te videoteke.

2. TEMPORALNE BAZE PODATAKA

Temporalnim bazama podataka nazivaju se tradicionalne baze podataka koje imaju temporalnu odnosno vremensku komponentu. Temporalni podaci pohranjeni u takvim bazama podataka različiti su od podataka pohranjenih u netemporalnim bazama podataka po tome što je vremenski zapis vezan za podatak te govori kada je taj podatak bio važeći ili pohranjen u bazi podataka (TimeConsult, 2015).

Temporalne baze podataka omogućuju reprezentaciju povijesti objekata te se temporalni relacijski model sastoji od temporalnih relacija, temporalne relacijske algebra i temporalnih uvjeta integriteta (Maleković, 2008, str. 73).

Drugim riječima, temporalne baze podataka daju odgovor kada je podatak bio važeći u nekom vremenu. Temporalne baze podataka imaju nebrojeno primjena, a neke od njih su primjerice informacije o osobama o kojima države, županije, općine i drugi vode evidencije. Tu spadaju datumi rođenja, datumi ženidbe, datum položenog vozačkog ispita, datum valjanosti osobnih dokumenata, kod bankarskih sustava datumi isteka kartica, vremenski plan otplate kredita, povijesno stanje bankovnih računa i još mnoge druge. Bez temporalnih baza podataka danas poslovanje te cjelokupno funkcioniranje sustava ne bi bilo moguće stoga one imaju jako bitnu ulogu u današnjem “digitalnom” svijetu.

2.1 Primjeri rada s temporalnim bazama podataka

U sklopu ovog rada koristit će se temporalne baze podataka pa će u nastavku ovog dijela biti objašnjene neke operacije nad temporalnim komponentama.

Dva glavna vremenska tipa u PostgreSQL-u su “date” (hrv. Datum) i “timestamp” (hrv. Vremenski pečat). Postoji još vremenskih tipova ali su oni varijacije nad spomenuta dva osnovna. “Date” tip sadrži datum (bez vremena) te je sljedećeg formata: “gggg-mm-dd”. “Timestamp” je vremenski tip koji sadrži i vrijeme uz datum. Može sadržavati i vremensku zonu, no osnovni oblik je: “gggg-mm-dd hh:mm:ss”.

U nastavku će biti prikazane neke operacije nad zanimljivijim funkcijama i operatorima u PostgreSQL-u u obliku tablice radi jednostavnosti. Kada se radi s vremenskom dimenzijom i operatorima kod “date” tipa koristi se ključna riječ “integer”, a kod “timestamp” tipa ključna riječ “interval” gdje se definira vremenski iznos (“hours”, “minutes”, “days”…). Rezultat

oduzimanja “timestamp”-a je uvijek interval dok je kod “date”-a rezultat cijeli broj (eng. *Integer*).

Tako (*PostgreSQL, 2015) navodi sljedeće primjere:

Tablica 1. Vremenski operatori

Operator	Timestamp	Rezultat	Date	Rezultat
+	timestamp '2001-09-28 01:00' + interval '23 hours'	timestamp '2001-09-29 00:00:00'	date '2001-09-28' + integer '7'	date '2001-10-05'
-	timestamp '2001-09-29 03:00' - timestamp '2001-09-27 12:00'	interval '1 day 15:00:00'	date '2001-10-01' - date '2001-09-28'	integer '3'
*	900 * interval '1 second'	interval '00:15:00'	-	-
/	interval '1 hour' / double precision '1.5'	interval '00:40:00'		

Izvor: https://www.postgresql.org/docs/8.1/static/functions-datetime.html, 03.08.2016., prilagođeno

U tablici 1. navedeni su primjeri rada s operatorima, dok je u sljedećoj tablici prikazan primjer rada s nekim funkcijama za rad s vremenskim tipovima (*PostgreSQL, 2015):

Tablica 2. Funkcije za rad s vremenskim tipovima

FUNKCIJA	Primjer	Rezultat
date_part(text, timestamp)	date_part('month', interval '2 years 3 months')	3
extract(field from timestamp)	extract(month from interval '2 years 3 months')	3
justify_hours(interval)	justify_hours(interval '24 hours')	1 dan
justify_days(interval)	justify_days(interval '30 days')	1 mjesec
now()	now()	Trenutni vremenski pečat

Izvor: https://www.postgresql.org/docs/8.1/static/functions-datetime.html, 03.08.2016., prilagođeno

Važno je napomenuti da ovo nisu jedine funkcije za rad s vremenskim tipovima. One su izabrane iz razloga što su se autoru ovog rada činile najzanimljivijima.

3. POOPĆENE BAZE PODATAKA

Poopćene relacijske baze podataka su one čija se shema sastoji i od složenih atributa, odnosno atribut A je složen ako njegova domena sadrži barem jedan složeni objekt. Aplikacije zahtijevaju takve relacije (Maleković, 2008, str. 123).

Prema (Schatten, 2016) poopćene-relacijske baze podataka omogućuju korištenje kompleksnijih objekata:

- BLOB[1]
- Polja (1D, 2D)
- Pobrojene vrijednosti (ENUM[2])
- Složeni tipovi

3.1 BLOB

BLOB-ovi se koriste za pohranu raznih binarnih podataka poput različitih dokumenata, slika, videa itd. (Schatten, 2016).

Takve binarne datoteke se spremaju u posebne objekte koji se zovu OID[3] (hrv. Identifikatori objekta).

Prilikom rada s OID-ima na raspolaganju je nekoliko funkcija:

- „lo_import“ funkciju – Koristi se kako bi se „uvezla“ binarna datoteka s klijentskog računala u samu bazu podataka. Sintaksa funkcije je: lo_import(lokacija_datoteke).
- „lo_export“ funkciju – Koristi se kako bi se „izvezla“ binarna datoteka iz OID-a u orginalnu datoteku. Sintaksa funkcije je lo_export(OID, ime_datoteke_s_ektenzijom).
- „lo_unlink“ funkciju – Koristi se za brisanje veze s objektom, drugim riječima njom brišemo sami OID iz baze podataka. Sintaksa funkcije je lo_unlink(OID).

Prednosti ovog načina pohrane datoteka su u tome što je u slučaju migracije podataka dovoljno napraviti „izvoz“ svih OID-a u bazi te se tako ponovno posjeduju sve datoteke koje su pohranjene. Još jedna od pogodnosti je kod datoteka koje se s vremenom mijenjaju. Ako se uvede i temporalna komponenta mogu se imati različita stanja datoteka kroz vrijeme. Korisniku

[1] Binary Large Object (hrv. Binarna datoteka)
[2] Enumerated (hrv. Pobrojenja)
[3] Object Identifiers (hrv. Identifikatori objekata)

se mogu prikazati samo najnovije datoteke, a za povijesne potrebe, napravi se „izvoz“ ukoliko je potreban.

3.2 Polja

Polja su kod poopćeno-relacijskih baza podataka ili jednodimenzionalna ili dvodimenzionalna. Kako bi se kreiralo polje dovoljno je navesti tip podatka te u uglatim zagradama definirati dimenzije polja. Primjerice, kod tablice „popusti“ je definirano kako kvartalno postoji dozvoljen popust na određenu uslugu (vrsta popusta nije bitna, ovdje je samo kako bi se dao primjer rada s poljima). Pomoću polja će se definirati broj iskorištenih popusta:

```
CREATE TABLE "popusti" (
    "ime" varchar(32) NOT NULL,

    "iskoristen" bool[4]
);
```

Nakon što se unese slog:

```
INSERT INTO popusti VALUES ('20% gratis', '{false,false,false,false}');
```

Zavisno od kvartala (u ovom slučaju 1. kvartal) postavi se popust ili na iskorišten (true) ili na neiskorišten (false).

```
UPDATE popusti SET iskoristen [1] = '{true}' WHERE ime = '20% gratis';
```

3.3 Pobrojene vrijednosti

Tip atributa koji je definiran s pobrojenom vrijednošću može poprimiti samo vrijednosti iz domene pobrojene vrijednosti. S obzirom na to, vrijednosti kojima je pobrojena vrijednost definirana, može se gledati kao konstante pošto se one ne mogu mijenjati.

Primjer pobrojenja je sljedeći:

```
CREATE TYPE spol AS ENUM ('M', 'Ž');
```

Ovdje atribut definiran s pobrojenom vrijednosti „spol“ može poprimiti samo vrijednosti ili „M“ ili „Ž“ gdje su vrijednosti osjetljive na velika i mala slova.

3.4 Složeni tipovi

Složeni tipovi su tipovi koji se grade od više jednostavnih tipova. Primjer za njih je sljedeći:

```
CREATE TYPE identitet AS (
    "ime" varchar(30),
    "prezime" varchar(30),
    "datum_rodenja" date
);
```

Ovdje je tip „identitet“ sastavljen od 3 jednostavna tipa (varchar, varchar, date). Neka u tablici „A“ postoji atribut „B“ koji je definiran složenim tipom „identitet“, sintaksa za unos vrijednosti atributa „B“ je sljedeća:

```
INSERT INTO A VALUES (ROW('Marko', 'Maric', '1992-11-03'));
```

Vrijednosti se unose uz pomoć ključne riječi „ROW“.

4. OBJEKTNO-RELACIJSKE BAZE PODATAKA

Baze temeljene na poopćenom relacijskom modelu podataka su objektno-relacijske baze. Nadalje, objektno relacijski sustav je sustav za upravljanje objektno relacijskom bazom podataka (Maleković, 2008., str. 16).

Objektno-relacijski sustav ili prošireni relacijski sustav je pokušaj spajanja najboljeg iz relacijskog i objektno-orijentiranog pristupa (Napredni modeli i baze podataka, 2014).

Ideja je bila da se u baze podataka uvedu objektno orijentirani koncepti iz drugih programskih jezika poput C++, PHP, C#, Java i sl.

Pa prema tome objektno relacijske baze podataka imaju sljedeće karakteristike (Napredni modeli i baze podataka, 2014):

- Temeljene su na relacijskom modelu podataka
- sačuvane su relacijske karakteristike
- Zadržavaju kompatibilnost s postojećim relacijskim jezicima
- Omogućuju nasljeđivanje tipova i tablica
- Posjeduju ugniježdene relacije
- ...

Koncept na koji će se ovaj rad, što se tiče objektno-relacijskih baza podataka, fokusirati je nasljeđivanje (generalizacija). Princip nasljeđivanja je gotovo identičan drugim programskim jezicima samo što se ovdje koriste tablice umjesto klasa. Definicija će biti objašnjena na sljedećem primjeru:

Tablica „vozilo“ ima sljedeće atribute:

```
 CREATE TABLE "vozilo" (

    "broj_mjesta_za_sjedenje" int4 NOT NULl,
    "boja" varchar(20) NOT NULl
);
```

Uz tablicu „vozilo“ postoji i tablica „bicikl“ koja ima sljedeće atribute:

```
CREATE TABLE "bicikl" (
    "tip_rame" varchar(15) NOT NULl,
    "elektricni_pogon" bool NOT NULl
)
INHERITS ("vozilo");
```

Ključna riječ „INHERITS“ koristi se kako bi naslijedili atribute neke tablice. U prethodnom slučaju tablica „bicikl“ je naslijedila sve atribute tablice „vozilo“ tako pri kreiranju tablice „bicikl“ postoje 4 atributa (2 iz tablice „vozilo“ te 2 iz tablice „bicikl“). Ovime se eliminira redundantnost koda i izbjegava mogućnost da neki atribut u tablici nije definiran. Za popunjavanje tablice „bicikl“ mogao bi se koristiti sljedeći upit:

```
INSERT INTO bicikl VALUES (1, 'crna', 'ženska', false);
```

Ukoliko se želi popis svih vozila u bazi umjesto upita nad svakom tablicom, koja se odnosi na neko vozilo, koristimo upit:

```
SELECT * FROM vozilo;
```

U ovom radu će se nasljeđivanje koristiti kako bi definirali prava pristupa korisnicima unutar aplikacije.

5. AKTIVNE BAZE PODATAKA

„Aktivne baze podataka su baze podataka koje prate određene točke od interesa i kada se one dogode aktiviraju odgovarajući odgovor unutar određenog vremena. Željeno ponašanje je definirano u pravilima (pravila događaja – uvjeta – reakcije) koja su definirana i pohranjena u bazi podataka“ (Dayal, Hanson, Widom, 1994, str. 2).

Drugim riječima aktivne baze podataka temelje se na okidačima (eng. *Trigger*) koji imaju određenu reakciju na definirani događaj ako su uvjeti zadovoljeni. U odnosu na aktivne, tradicionalne baze podataka su pasivne, odnosno samo izvršavaju upite ili transakcije na zahtjev korisnika ili određene aplikacije (Dayal, Hanson, Widom, 1994, str. 2). Kao što je navedeno u tekstu iznad aktivne baze podataka su vezane za samu bazu podataka te se one izvršavaju neovisno od korisnika ili aplikacija koje koriste bazu podataka, iako se u dosta slučajeva okidači koriste da izvrše određenu radnju nakon što je korisnik izvršio neku radnju (određeni upit).

Prema (**PostgreSQL, 2016) sintaksa za kreiranje okidača je sljedeća:

```
CREATE [ CONSTRAINT ] TRIGGER name { BEFORE | AFTER | INSTEAD OF } { event [
OR ... ] }
    ON table
    [ FROM referenced_table_name ]
    [ NOT DEFERRABLE | [ DEFERRABLE ] { INITIALLY IMMEDIATE | INITIALLY
DEFERRED } ]
    [ FOR [ EACH ] { ROW | STATEMENT } ]
    [ WHEN ( condition ) ]
    EXECUTE PROCEDURE function_name ( arguments )
```

Gdje je:

- „name“ – Ime okidača
- „ON“ – Definira za koju tablicu je okidač vezan
- „BEFORE“, „AFTER“ i „INSTEAD OF“ označavaju kada se okidač aktivira: prije, poslije ili umjesto određene akcije and tablicom/pogledom (uvjet „INSTEAD OF“ se može definirati samo na pogledu)
- „FOR EACH ROW | STATEMENT“ – Ukoliko je odabran uvjet „FOR EACH ROW“ okidač pokreče proceduru za svaki izmijenjeni, obrisani ili umetnuti slog u tablici. Ako je odabran uvjet „FOR EACH PROCEDURE“ okidač pokreće proceduru za samo prvi izmijenjeni, obrisani ili umetnuti slog.
- „EXECUTE PROCEDURE“ – Definira koja će se procedura (funkcija) izvršiti aktiviranjem okidača

5.1 Funkcije okidača

Pod pojmom procedure kod okidača u PostgreSQL-u se ustvari misli na funkcije okidača pošto PostgreSQL poznaje samo funkcije. Stoga kada se govori o proceduri okidača misli se na funkciju okidača.

Na njih se može gledati kao na ponovno iskoristivi dio koda, ideja nije ništa drugačija u odnosu na ostale programske jezike razlika je samo u sintaksi.

U nastavku je definiran primjer funkcije koja će se zvati „zapisi_obrisanog_korisnika_u_log“. Sintaksa je sljedeća:

```
CREATE FUNCTION zapisi_obrisanog_korisnika_u_log()
RETURNS TRIGGER AS $$
BEGIN
INSERT INTO log VALUES (DEFAULT,'Korisnik korisničkog imena "'
|| OLD.username||'" je obrisan iz baze');
RETURN NEW;
END;
$$
LANGUAGE plpgsql;
```

„CREATE FUNCTION“ – Služi PostgreSQL-u da kreira funkciju imena „zapisi_obrisanog_korisnika_u_log“

„RETURNS trigger“ – Povratni tip funkcije je okidač

„BEGIN“ i „END“ – Označavaju početak, odnosno završetak tijela funkcije.

„RETURN NEW“ – Znači vraćanje nove vrijednosti sloga

„LANGUAGE plpgsql“ – Govori da se radi o PostgreSQL jeziku

5.2. Objekti OLD i NEW

U tijelu funkciji je navedena ključnu riječ „OLD“ kod definiranja upita. „OLD“ je objekt koji sadrži staru vrijednost sloga (stare vrijednosti koje se brišu ili ažuriraju). Uz „OLD“ postoji i „NEW“ te on sadrži novu vrijednost sloga (nove vrijednosti koje se unose ili ažuriraju). „OLD“ i „NEW“ su dostupni zavisno od akcije koja je aktivirala okidač. Još jedan operator je operator || koji služi za spajanje tekstualnih vrijednosti. U nastavku je prikazana tablica koja prikazuje dostupnost navedenih objekata.

Tablica 3. Dostupnost „OLD“ i „NEW“ objekata

AKCIJA	DOSTUPNI OBJEKTI	OBJAŠNJENJE
INSERT	NEW	S obzirom da se radi o umetanju sloga taj slog prethodno nije postojao pa stoga niti ne postoje konkretne vrijednosti osim NULL (nepoznatih) vrijednosti. Stoga ako se koristi OLD objekt svi atributi bi bili null jer taj slog nije postojao do tada.
UPDATE	NEW i OLD	Prilikom ažuriranja ažurira se **postojeći** slog stoga je dostupan objekt OLD (vrijednosti već postoje) i objekt NEW (nove vrijednosti koje se ažuriraju).
DELETE	OLD	S obzirom da se kod brisanja briše postojeći slog iz baze dostupan je objekt OLD (vrijednosti su već postojale), a kod objekta NEW su sve vrijednosti NULL jer je slog obrisan i više ne postoji.

Iako će se u nastavku rada opisati i definirati svi okidači i procedure u bazi podataka, ovdje je prikazan primjer jednostavnog okidača. Dakle postoji jednostavna tablica „korisnici“ gdje se nalaze svi korisnici aplikacije, a ima sljedeću sintaksu:

```
CREATE TABLE "korisnici" (
    "korisnicki_id" SERIAL NOT NULL,
    "korisnicko_ime" varchar(20) NOT NULL,
    "lozinka" char(32) NOT NULL,
    PRIMARY KEY("korisnicki_id")
);
```

Želi se definirati okidač koji će izvršiti proceduru „zapisi_obrisanog_korisnika_u_log“ koja je već unaprijed definirana, a zapisuje u tablicu „log“ podatke o obrisanom korisniku. Sintaksa tablice „log“ je sljedeća:

```
CREATE TABLE "log" (
    "id_log" SERIAL NOT NULL,
    "opis" text NOT NULL,
    PRIMARY KEY("id_log")
);
```

Nakon toga, definira se okidač kojem je sintaksa sljedeća:

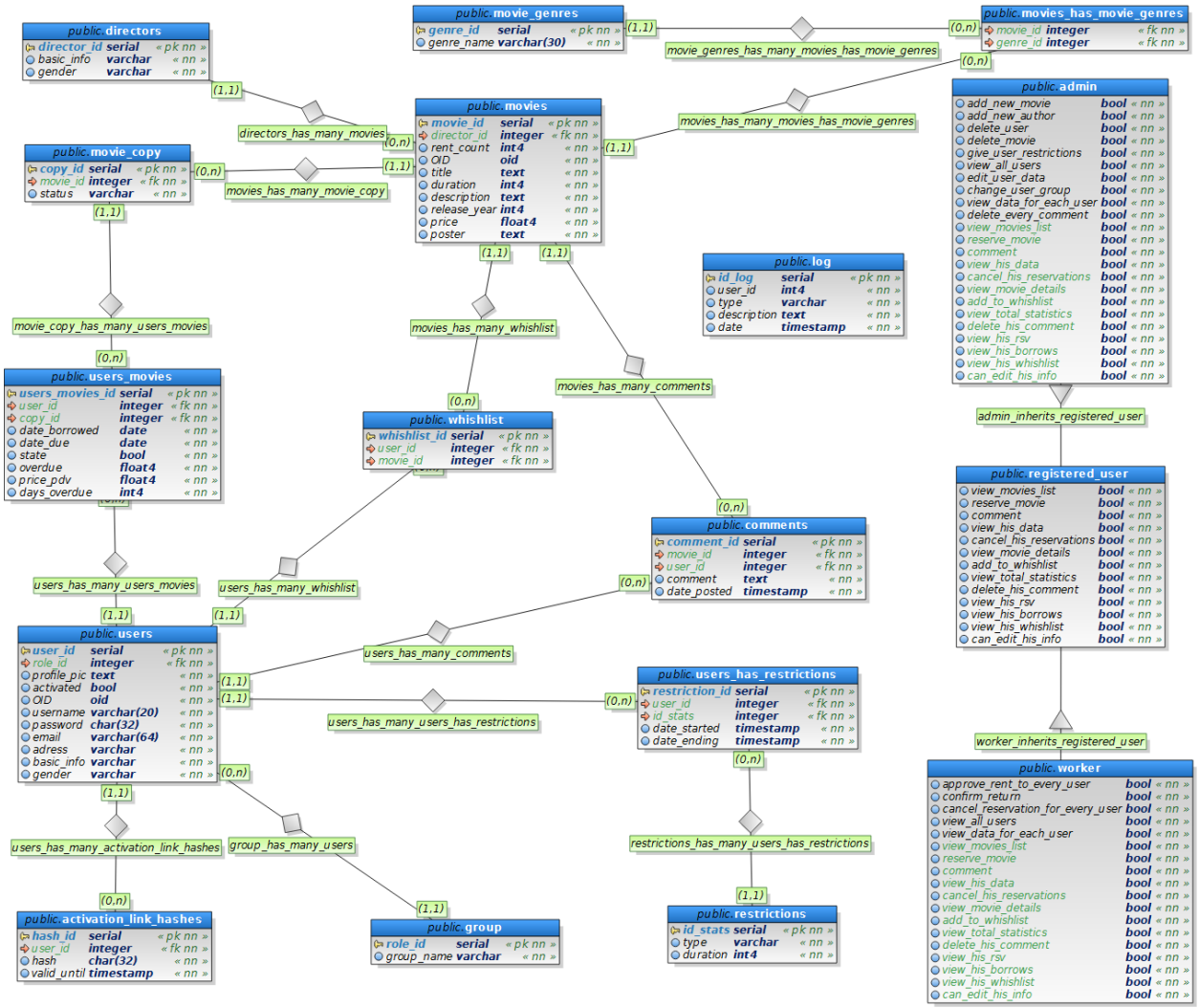
```
CREATE TRIGGER korisnik_obrisan
AFTER DELETE ON korisnici
FOR EACH ROW
EXECUTE PROCEDURE zapisi_obrisanog_korisnika_u_log();
```

Ovdje se koristi uvjet „AFTER DELETE“ jer se želi da se okidač aktivira samo ukoliko je korisnik obrisan. Ne želi se zapisivati u tablicu „log“ ukoliko brisanje nije uspješno prošlo.

6. ERA MODEL

Slika 6.1. prikazuje model entiteta i veza za bazu podataka. Model je rađen uz pomoć alata pgModeler te se sastoji od 17 tablica:

- "users_has_restrictions" – hrv. Korisnik ima zabrane
- "group" – hrv. Grupe korisnika
- "directors" – hrv. Redatelji
- "registered_user" – hrv. Registrirani korisnik
- "admin" – hrv. Administratori
- "worker" – hrv. Radnik
- "movie_genres" – hrv. Žanrovi filmova
- "movies_has_movie_genres" – hrv. Film ima žanrove
- "users_movies" – hrv. Korisnici Filmovi
- "restrictions" – hrv. Zabrane
- "activation_link_hashes" – hrv. Aktivacijske šifre
- "movie_copy" – hrv. Kopija filma
- "log" – hrv. Zapisnik
- "users" – hrv. Korisnik
- "movies" – hrv. Filmovi
- "whishlist" – hrv. Lista želja
- "comments" – hrv. Komentari



Slika 6.1. ERA model

Izvor: autor

7. APLIKACIJSKA DOMENA

Aplikacija „RentalVideo“ izrađena kao praktični dio ovog diplomskog rada je aplikacija koja služi videotekama za upravljanje zalihama filmova te korisnicima za online rezervacije i posudbe filmova. Aplikacija kao sustav za upravljanje bazom podataka koristi PostgreSQL. Kao bi ostvarili sve zamišljene funkcionalnosti korištene su temporalne baze podataka, kao temelj relacijske baze podataka, aktivne baze podataka, objektno-relacijske baze podataka te poopćene baze podataka. Aplikaciju pokreće „Apache“ server, a programski jezik na strani poslužitelja je PHP. Za izradu korisničkog sučelja korišten je HTML za strukturu podataka te CSS za definiranje stilova korisničkog sučelja. Aplikacija sadrži tri razine (grupe) korisnika :

- Registrirane korisnike
- Administratore sustava
- Radnike u videoteci

S obzirom na to da se radi o videoteci uz navedene razine korisnika još jedan važan entitet u našoj bazi su filmovi.

U nastavku ovog rada će se detaljno objasniti sve navedeno.

7.1 Razine korisnika

NAPOMENA 1: Razine prava su definirane kao istinosne vrijednosti jer je sam problem korisničkih prava riješen uz pomoć objektno-relacijskih baza podataka stoga je to prvenstveno bilo najlogičnije rješenje. U nastavku rada bit će detaljnijeg govora o spomenutome.

NAPOMENA 2: Navedena prava su specifična samo za grupu korisnika o kojoj je riječ. U nastavku će biti govora o nasljeđivanju prava pa će određena razina korisnika „naslijediti“ prava drugih korisnika.

7.1.1 Registrirani korisnici

Pod pojom registriranog korisnika misli se na običnog korisnika koji nije niti administrator niti radnik u videoteci stoga će se pojmovi „registrirani korisnik“ te „obični korisnik“ u nastavku teksta odnositi na jedno te isto. Prilikom registracije svaki korisnik je

prema zadanim postavkama obični korisnik. Obični korisnici su najveći korisnici aplikacije. Oni unutar aplikacije imaju sljedeća prava:

- "view_movies_list" - hrv. Pregled svih filmova
- "reserve_movie"- hrv. Rezervacija (rezervacija filmova)
- "comment" - hrv. Komentiranje
- "view_his_data" - hrv. Pregled svojih podataka
- "cancel_his_reservation" - hrv. Otkazivanje svoje rezervacije
- "view_movie_details" - hrv. Pregled pojedinačnog filma
- "add_to_wishlist" - hrv. Dodavanje na listu želja
- "view_total_statistics" - hrv. Pregled sveukupne statistike
- "delete_his_comment" - hrv. Brisanje svojih komentara
- "view_his_rsv" - hrv. Pregled svojih rezervacija
- "view_his_borrows" - hrv. Pregled svojih posudbi
- "view_his_whishlist" - hrv. Pregled svoje liste želja
- "can_edit_his_info" - hrv. Editiranje svojih informacija

7.1.2 Radnik videoteke

Radnik videoteke je osoba koja je fizički prisutna u videoteci. Ona se brine o fizičkoj predaji filma korisniku te vodi evidenciju u sustav kada je korisnik preuzeo tj. posudio film i kod povratka filma u videoteku. Također bilježi zaprimanje filma te evidenciju istog u sustav. Radnik u aplikaciji ima sljedeća prava specifična za njegovu razinu:

- "approve_rent_to_every_user" - hrv. Odobravanje rezervacije svakom korisniku (to uključuje i samog radnika)
- "confirm_return" - hrv. Zaprimanje filma (fizički povratak filma u videoteku i evidencija u sustav)
- "cancel_reservation_for_every_user" - hrv. Otkazivanje rezervacije za svakog korisnika
- "view_all_users" - hrv. Pregled svih korisnika
- "view_data_for_each_user" - hrv. Pregled podataka svih korisnika

7.1.3 Administrator

Administrator je osoba koja se brine o održavanju stranice, odnosno u ovom slučaju dodavanje novih filmova, blokiranje korisnika, izmjena korisničkih grupa i druge mogućnosti.

U našem sustavu administrator ima sljedeća prava:

- "add_new_movie" - hrv. Dodavanje novog filma
- "add_new_author" - hrv. Dodavanje novog redatelja
- "delete_user" - hrv. Brisanje korisnika
- "delete_movie" - hrv. Brisanje filma
- "give_user_restrictions" - hrv. Davanje zabrana korisniku
- "view_all_users" - hrv. Pregled svih korisnika
- "change_user_group" - hrv. Izmjena korisničke grupe
- "delete_every_comment" - hrv. Brisanje svakog komentara
- "view_data_for_each_user" - hrv. Pregled podataka svih korisnika

7.1.4 Filmovi

Filmovi su temelj poslovanja u videoteci. U bazi podataka film je jedinka koja se može posuditi nego se može posuditi kopija filma. Stoga se u bazi pohranjuju informacije o samom filmu, a posuđuju se kopije filmova. Svaki film može imati N kopija. Informacije koje se pohranjuju o svakom filmu su sljedeće:

- Tko je redatelj filma
- Naziv filma
- Trajanje
- Opis
- Godina izdavanja
- Cijena
- Poster
- Broj posuđivanja

8. IMPLEMENTACIJA BAZE PODATAKA

8.1 Složeni tipovi

U bazi podataka postoji 7 složenih tipova (adresa, spol, grupe, statusi filmova...). Oni služe kako bi vrijednosti atributa poprimile točno određene vrijednosti, odnosno kako bi se jasno definirale vrijednosti pojedinih atributa. U nastavku teksta svaki će dio biti detaljnije objašnjen.

8.1.1 Adress_info

Kod za kreiranje složenog tipa „adress_info“ (hrv. Adresa) je sljedeći:

```
CREATE TYPE adress_info AS (
    city_name varchar(64),
    street_name varchar(64),
    postal_number int4,
    house_number int4
);
```

On služi kako bi se točno odredilo što se od podataka mora unijeti kako bi se definirala nečija adresa.

Sastoji se od sljedećih atributa:

- „city_name“ hrv. Ime grada
- „street_name“ hrv. Ime ulice
- „postal_number“ hrv. Poštanski broj
- „house_number“ hrv. Kućni broj

8.1.2 Basic_info

Kod za kreiranje složenog tipa „basic_info“ (hrv. Osnovne informacije) je sljedeći:

```
CREATE TYPE basic_info AS (
    "fname" varchar(30),
    "lname" varchar(30),
    "date_of_birth" date
);
```

Složeni tip se koristi kako bi definirali osnovne informacije koje svaki korisnik videoteke mora imati kako bi se deifinirao njegov identitet. Ovdje se misli o pravom identitetu, a ne identitetu na web stranici (korisničko ime i sl.).

On se sastoji od sljedećih atributa:

- "fname" hrv. Ime

- "lname" hrv. Prezime
- "date_of_birth" hrv. Datum rođenja

8.1.3 Gender_info

Kod za kreiranje pobrojene vrijednosti „gender_info“ (hrv. Spol) je sljedeći:

```
CREATE TYPE gender_info AS ENUM ('M', 'F');
```

Kod za kreiranje ove pobrojene vrijednosti je prilično samoobjašnjavajući. Koristi se da definiranje spola korisnika te može poprimiti samo dvije vrijednosti:

- „M“ hrv. Muško
- „F“ hrv. Žensko

8.1.4 Restriction_type

Kod za kreiranje pobrojene vrijednosti „restriction_type“ (hrv. Tip zabrane) je sljedeći:

```
CREATE TYPE restriction_type AS ENUM ('comments', 'borrow', 'rent');
```

Služi kako bi se definiralo koje zabrane korisnik može imati. One mogu biti:

- „comments“ hrv. Komentiranja
- „borrow“ hrv. Posudba
- „rent“ hrv. Reservacija

8.1.5 Role_info

Kod za kreiranje pobrojene vrijednosti „role_info“ (hrv. Uloga):

```
CREATE TYPE role_info AS ENUM ('registered_user','admin','worker');
```

Služi za definiranje tipa korisnika. U sustavu postoje 3 razine korisnika:

- „registered_user“ hrv. Registrirani korisnik
- „admin“ hrv. Administrator
- „worker“ hrv. Radnik

8.1.6 Status_info

Kod za kreiranje pobrojene vrijednosti „status_info“ (hrv. Status):

```
CREATE TYPE status_info AS ENUM('available','reserved','borrowed');
```

Služi kako bi definirali u kojem je stanju film. U videoteci status filma može poprimiti sljedeće vrijednosti:

- „available“ hrv. Dostupan
- „reserved“ hrv. Rezerviran
- „borrowed“ hrv. Posuđen

8.1.7 Log_t

Kod za kreiranje pobrojene vrijednosti „log_t“ (hrv. Log) je sljedeći:

```
CREATE TYPE log_t AS ENUM('comments', 'borrow', 'rent', 'registration',
'activation', 'deletion', 'deletionUser', 'group');
```

Ova pobrojena vrijednost je jako važna jer se koristi prilikom zapisivanja u tablicu „log“. Nju se može smatrati kao šifrom koju će se koristiti za filtriranje, i naposljetku grupiranje, određenih vrsta zapisa u tablici „log“. Tipovi zapisa su sljedeći:

- „comments“ hrv. Komentari (za korisnike koji su dobili zabranu komentiranja)
- „borrow“ hrv. Posudbe (za korisnike koji su dobili zabranu posudbi)
- „rent“ hrv. Rezervacije (za korisnike koji su dobili zabranu rezervacija)
- „registration“ hrv. Registracije (za zapis korisnika koji su se registrirali)
- „activation“ hrv. Aktivacija (za zapis korisnika koji su aktivirali račun)
- „deletion“ hrv. Brisanje (za zapis obrisanih filmova)
- „deletionUser“ hrv. Brisanje korisnika (za zapis obrisanih korisnika)
- „group“ hrv. Grupa (za zapis promjene korisničke grupe)

8.2 Tablice

Kod za kreiranje tablice „log“je sljedeći:

```
CREATE TABLE "log" (
    "id_log" SERIAL NOT NULL,
    "user_id" int4 NOT NULL,
    "type" log_t NOT NULL,
```

```
    "description" text NOT NULL,
    "date" timestamp NOT NULL,
    PRIMARY KEY("id_log")
);
```

Tablica „log“ (hrv. Dnevnik) koristi se kako bi se vodila evidencija o određenim akcijama koje se događaju u bazi. Evidenciju se vodi o 7 tipova događaja (tip atributa je „log_t“) koje okidači zapisuju u tablicu. Postoji 5 atributa u tablici, a to su:

- "id_log" - Jedinstveni identifikator tablice
- "user_id" – identifikator koji poprima vrijednosti > 0 ukoliko se radi o korisničkom logu, a vrijednosti -1 u slučaju filma
- "type" – tip događaja koji se dogodio (složeni pobrojeni tip)
- "description" – opis događaja koji se dogodio
- "date" – vrijeme kada se događaj dogodio

Kod za kreiranje tablice „users“je sljedeći:

```
CREATE TABLE "users" (
    "user_id" SERIAL NOT NULL,
    "role_id" int4 NOT NULL,
    "username" varchar(20) NOT NULL,
    "password" char(32) NOT NULL,
    "email" varchar(64) NOT NULL,
    "adress" adress_info NOT NULL,
    "basic_info" basic_info NOT NULL,
    "gender" gender_info NOT NULL,
    "profile_pic" text NOT NULL,
    "activated" bool NOT NULL DEFAULT False,
    "OID" oid NOT NULL,
    PRIMARY KEY("user_id")
);
```

Tablica „users“ (hrv. Korisnici) je jedna od najbitnijih tablica u samoj bazi. U nju se pohranjuju informacije o korisnicima aplikacije. Atributi u tablici su sljedeći:

- "user_id" – Jedinstveni korisnički identifikator
- "role_id" – Vanjski ključ na tablicu „group“ koja sadrži korisničke uloge tj. razine
- "username" – Korisničko ime
- "password" - Lozinka
- "email" - Email
- "adress" – Složeni tip „adress_info“ kojim se definira adresu korisnika
- "basic_info" – Složeni tip kojim se definiraju osnovne informacije o korisniku
- "gender" - Pobrojena vrijednosti kojom se definira spol korisnika
- "profile_pic" - Atribut koji kao vrijednost ima punu putanju do profilne slike.
- "activated" – hrv. Aktiviran kojim se određuje da li je korisnik aktivirao svoj račun

- "OID" – BLOB objekt kojim se fizički pohranjuje profilna sliku u bazu podataka

Kod za kreiranje tablice „directors“je sljedeći:

```
CREATE TABLE "directors" (
    "director_id" SERIAL NOT NULL,
    "basic_info" basic_info NOT NULL,
    "gender" gender_info NOT NULL,
    PRIMARY KEY("director_id")
);
```

Tablica „directors“ (hrv. Redatelji) služi za pohranu informacija o redateljima filmova u bazi podataka. Ona je definirana sljedećim atributima:

- "director_id" - Jedinstveni identifikator redatelja
- "basic_info" - Složeni tip kojim se definiraju osnovne informacije o redatelju
- "gender" - Pobrojena vrijednosti kojom se definira spol redatelja

Kod za kreiranje tablice „movies“je sljedeći:

```
CREATE TABLE "movies" (
    "movie_id" SERIAL NOT NULL,
    "director_id" int4 NOT NULL,
    "title" text NOT NULL,
    "duration" int4 NOT NULL,
    "description" text NOT NULL,
    "release_year" int4 NOT NULL,
    "price" float4 NOT NULL,
    "poster" text NOT NULL,
    "rent_count" int4 NOT NULL DEFAULT 0,
    "OID" oid,
    PRIMARY KEY("movie_id")
);
```

Tablica „movies“ (hrv. Filmovi) se koristi za pohranu filmova te informacija o samom filmu. Ona sadrži sljedeće atribute:

- "movie_id" - Jedinstveni identifikator filma
- "director_id" – Vanjski ključ na tablicu „directors“ koja sadrži popis redatelja
- "title" – Naziv filma
- "duration" – Trajanje filma
- "description" – Radnja filma
- "release_year" – Godina izdavanja filma
- "price" – Cijena filma
- "poster" - Atribut koji kao vrijednost ima punu putanju do postera filma.
- "rent_count" – Broj posudbi filma
- "OID" - BLOB objekt kojim se fizički pohranjuje poster filma u našu bazu podataka

Kod za kreiranje tablice „movie_genres“je sljedeći:

```
CREATE TABLE "movie_genres" (
    "genre_id" SERIAL NOT NULL,
    "genre_name" varchar(30) NOT NULL,
    PRIMARY KEY("genre_id")
);
```

Tablica „movie_genres“ (hrv. Filmski žanrovi) služi za pohranu žanrova filmova. Sastoji se od sljedećih atributa:

- "genre_id" – Jedinstveni identifikator žanra
- "genre_name" – Naziv žanra

Kod za kreiranje tablice „group“je sljedeći:

```
CREATE TABLE "group" (
    "role_id" SERIAL NOT NULL,
    "group_name" role_info DEFAULT 'registered_user' NOT NULL,
    PRIMARY KEY("role_id")
);
```

Tablica „group“ (hrv. Grupa) je kreirana kako bi se mogla pohraniti imena grupa tj. različitih uloga u sustavu te tako postoje sljedeći atribute kojima je ova tablica definirana:

- „role_id“ – Jedinstveni identifikator grupe
- „group_name“ – Ime grupe koje je tipa pobrojene vrijednosti te može poprimiti jednu od 3 vrijednosti koje su ranije spomenute.

Kod za kreiranje tablice „movies_has_movie_genres“je sljedeći:

```
CREATE TABLE "movies_has_movie_genres" (
    "movie_id" int4 NOT NULL,
    "genre_id" int4 NOT NULL
);
```

Tablica „movies_has_movie_genres“ (hrv. Film ima žanrove) je kreirana kako bi se riješio problem veze više-više između tablica „movies“ i „movie_genres“ pošto svaki film može imati više žanrova, a pojedini žanr može pripadati više filmova. Ona je definirana sljedećim atributima:

- „movie_id“ – Vanjski ključ na tablicu „movies“
- „genre_id“ – Vanjski ključ na tablicu „movie_genres“

Kod za kreiranje tablice „users_movies“ je sljedeći:

```
CREATE TABLE "users_movies" (
    "users_movies_id" SERIAL NOT NULL,
    "copy_id" int4 NOT NULL,
    "user_id" int4 NOT NULL,
    "date_borrowed" date NOT NULL,
    "date_due" date NOT NULL,
    "state" bool NOT NULL DEFAULT False,
    "overdue" float4 NOT NULL DEFAULT 0,
    "price_pdv" float4 NOT NULL DEFAULT 0,
    "days_overdue" int4 NOT NULL DEFAULT 0,
    PRIMARY KEY("users_movies_id")
);
```

Tablica „users_movies“ (hrv. Korisnici Filmovi) je tablica koja pohranjuje informacije o trenutnim rezervacijama te o posuđenim filmovima (vraćenim i trenutno posuđenim). Informacije koje pohranjuje su sljedeće:

- "users_movies_id" – Jedinstveni identifikator tablice „users_movies“
- "copy_id" – Vanjski ključ na tablicu „movies“
- "user_id" - Vanjski ključ na tablicu „users“
- "date_borrowed" – Datum (posudbe ili rezervacije)
- "date_due" – Datum do kada vrijedi (posudba ili rezervacija)
- "state" – Stanje (odnosi se na posuđeni film tj. da li je vraćen ili još uvijek posuđen. Za rezervacije ovaj atribut ima uvijek vrijednost „false“ (hrv. Lažno))
- "overdue" – Iznos zakasnine (samo kod posuđenih filmova)
- "price_pdv" – Cijena filma s pdv-om
- "days_overdue" – Broj dana za koliko korisnik kasni s povratkom filma (samo za posuđene filmove)

Kod za kreiranje tablice „movie_copy“ je sljedeći:

```
CREATE TABLE "movie_copy" (
    "copy_id" SERIAL NOT NULL,
    "movie_id" int4 NOT NULL,
    "status" status_info NOT NULL DEFAULT 'available',
    PRIMARY KEY("copy_id")
);
```

Tablica „movie_copy“ (hrv. Kopija filma) služi za pohranu informacija o broju kopija te statusu pojedine kopije kako bi se znalo koliko još dostupnih kopija ima, kako bi ih korisnik mogao posuditi. Tablica ima sljedeće atribute:

- „copy_id“ - jedinstveni indentifikator tablice „movie_copy“
- „movie_id“ - Vanjski ključ na tablicu „movies“

- „status“ – atribut tipa pobrojene vrijednosti „stauts_info“ te može poprimiti 3 vrste vrijednosti koje su ranije spomenute, a koje označavaju status filma

Kod za kreiranje tablice „comments“je sljedeći:

```
CREATE TABLE "comments" (
    "comment_id" SERIAL NOT NULL,
    "user_id" int4 NOT NULL,
    "movie_id" int4 NOT NULL,
    "comment" text NOT NULL,
    "date_posted" timestamp NOT NULL,
    PRIMARY KEY("comment_id")
);
```

Tablica „comments“ (hrv. Komentari), kao što joj i samo ime govori, je tablica u koju se pohranjuju komentari korisnika na određeni film. Podatci koji se prate su:

- „comment_id“ – Jedinstveni identifikator komentara
- „user_id“ – Vanjski ključ na tablicu „users“
- „movie_id“ – Vanjski ključ na tablicu „movies“
- „comment“ – Sadržaj komentara
- „date_posted“ – Datum i vrijeme objave komentara

Kod za kreiranje tablice „restrictions“je sljedeći:

```
CREATE TABLE "restrictions" (
    "id_stats" SERIAL NOT NULL,
    "type" restriction_type NOT NULL,
    "duration" int4 NOT NULL DEFAULT 1,
    PRIMARY KEY("id_stats")
);
```

Tablica „restrictions“ (hrv. Zabrane) je tablica koja sadrži imena svih mogućih zabrana koje se mogu dobiti unutar baze podataka. Atributi kojima je tablica definirana su:

- „id_stats“ – Jedinstveni identifikator zabrane
- „type“ – Tip zabrane koje korisnik može dobiti. Atribut je tipa „restriction_type“ za kojeg je prethodno navedeno koje vrijednosti može poprimiti.
- „duration“ – Trajanje pojedine zabrane

Kod za kreiranje tablice „users_has_restrictions“je sljedeći:

```
CREATE TABLE "users_has_restrictions" (
    "restriction_id" SERIAL NOT NULL,
    "user_id" int4 NOT NULL,
    "id_stats" int4 NOT NULL,
    "date_started" timestamp NOT NULL,
    "date_ending" timestamp NOT NULL,
    PRIMARY KEY("restriction_id")
);
```

Tablica je kreirana kako bi se riješio problem veze više-više između tablica „users“ i „restrictions“ pošto korisnik može imati više zabrana, a jedna zabrana može pripadati više korisnika.

Ona ima sljedeće atribute:

- „restriction_id“ – Jedinstveni identifikator zabrane koje korisnik ima
- „user_id“ – Vanjski ključ na tablicu „users“
- „id_stats“ – Vanjski ključ na tablicu „restrictions“
- „date_started“ – Vremenski zapis kada je zabrana počela
- „date_ending“ – Vremenski zapis kada zabrana završava

Kod za kreiranje tablice „activation_link_hashes“je sljedeći:

```
CREATE TABLE "activation_link_hashes" (
    "hash_id" SERIAL NOT NULL,
    "user_id" int4 NOT NULL,
    "hash" char(32) NOT NULL,
    "valid_until" timestamp NOT NULL,
    PRIMARY KEY("hash_id")
);
```

Tablica „activation_link_hashes“ (hrv. Aktivacijski ključevi) sadrži aktivacijske ključeve korisnika koji im se dodijeljuju prilikom registracije (i na zahtjev korisnika), a služe za aktivaciju korisničkih računa. Ona sadrži sljedeće atribute:

- „hash_id“ – Jedinstveni identifikator ključa
- „user_id“ – Vanjski ključ na tablicu „users“
- „hash“ – Aktivacijski ključ
- „valid_until“ – Vremenski zapis do kada aktivacijski ključ vrijedi

Kod za kreiranje tablice „whishlist“je sljedeći:

```
CREATE TABLE "whishlist" (
    "whishlist_id" SERIAL NOT NULL,
    "user_id" int4 NOT NULL,
    "movie_id" int4 NOT NULL,
    PRIMARY KEY("whishlist_id")
);
```

Tablica „whishlist“ (hrv. Lista Želja) je tablica u koju se spremaju filmovi koje bi korisnik htio pogledati, a iz nekog razloga trenutno ne može. Atributi kojima je tablica definirana su:

- „whishlist_id“ – Jedinstveni identifikator tablice „whishlist“
- „user_id“ - Vanjski ključ na tablicu „users“
- „movie_id“ - Vanjski ključ na tablicu „movies“

8.2.1 Kreiranje korisničkih prava

Preostale tri tablice u bazi su „registered_user“ (hrv. Registrirani korisnik), „admin“ (hrv. Administrator), worker (hrv. Radnik). One služe kako bi definirali prava korisnika odnosno koje ovlasti unutar aplikacije pojedina razina korisnika ima. U nastavku su vidljivi kod te objašnjenja prava za svaku pojedinu tablicu.

Kod za kreiranje tablice „registered_user“ je sljedeći:

```
CREATE TABLE "registered_user" (
    "view_movies_list" bool NOT NULL DEFAULT True,
    "reserve_movie" bool NOT NULL DEFAULT True,
    "comment" bool NOT NULL DEFAULT True,
    "view_his_data" bool NOT NULL DEFAULT True,
    "cancel_his_reservation" bool NOT NULL DEFAULT True,
    "view_movie_details" bool NOT NULL DEFAULT True,
    "add_to_wishlist" bool NOT NULL DEFAULT True,
    "view_total_statistics" bool NOT NULL DEFAULT True,
    "delete_his_comment" bool NOT NULL DEFAULT True,
    "view_his_rsv" bool NOT NULL DEFAULT True,
    "view_his_borrows" bool NOT NULL DEFAULT True,
    "view_his_whishlist" bool NOT NULL DEFAULT True,
    "can_edit_his_info" bool NOT NULL DEFAULT True,
);
```

Tablica „registered_user“ (hrv. Registrirani korisnik) je tablica koja sadrži sva prava koje posjeduje obični korisnik. Svi atributi su tipa **bool** odnosno imaju stanja true (hrv. Točno) ili false (hrv. Lažno). Drugim riječima korisnik nešto može ili ne može. Dohvat podataka te pristup samom djelu aplikacije na koje se pravo odnosi se vrši uz pomoć PHP-a te je baza samo zadužena za pohranu tih podataka. Prava, specifična za registriranog korisnika ima su sljedeća:

- "view_movies_list" - hrv. Pregled svih filmova
- "reserve_movie"- hrv. Rezervacija (rezervacija filmova)
- "comment" - hrv. Komentiranje
- "view_his_data" - hrv. Pregled svojih podataka
- "cancel_his_reservation" - hrv. Otkazivanje svoje rezervacije
- "view_movie_details" - hrv. Pregled pojedinačnog filma
- "add_to_wishlist" - hrv. Dodavanje na listu želja
- "view_total_statistics" - hrv. Pregled sveukupne statistike
- "delete_his_comment" - hrv. Brisanje svojih komentara
- "view_his_rsv" - hrv. Pregled svojih rezervacija
- "view_his_borrows" - hrv. Pregled svojih posudbi

- "view_his_whishlist" - hrv. Pregled svoje liste želja
- "can_edit_his_info" - hrv. Editiranje svojih informacija

Kod za kreiranje tablice „admin“je sljedeći:

```
CREATE TABLE "admin" (
    "add_new_movie" bool NOT NULL DEFAULT True,
    "add_new_author" bool NOT NULL DEFAULT True,
    "delete_user" bool NOT NULL DEFAULT True,
    "delete_movie" bool NOT NULL DEFAULT True,
    "give_user_restrictions" bool NOT NULL DEFAULT True,
    "view_all_users" bool NOT NULL DEFAULT True,
    "edit_user_data" bool NOT NULL DEFAULT True,
    "change_user_group" bool NOT NULL DEFAULT True,
    "delete_every_comment" bool NOT NULL DEFAULT True,
    "view_data_for_each_user" bool NOT NULL DEFAULT True


)
INHERITS ("registered_user");
```

Tablica „admin“ (hrv. Administrator) je tablica koja sadrži sva prava koje posjeduje administrator. Svi atributi su također tipa **bool** odnosno imaju stanja true (hrv. Točno) ili false (hrv. Lažno).

U kodu postoji dio „INHERITS(„registered_user“) koji omogućava tablici „admin“ da naslijedi sve atribute tablice „registered_user“. To je ključna riječ rezervirana za PostgreSQL i omogućava prethodno objašnjeni koncept nasljeđivanja tj. generalizacije podataka. Time tablica administrator posjeduje atribute definirane u svojoj tablici zajedno sa svim atributima definiranima u tablici „registered_user“.
Prava,specifična za administratora, su sljedeća::

- "add_new_movie" – hrv. Dodavanje novog filma
- "add_new_author" – hrv. Dodavanje novog redatelja
- "delete_user" – hrv. Brisanje korisnika
- "delete_movie" – hrv. Brisanje filma
- "give_user_restrictions" – hrv. Davanje ograničenja korisnicima (posudbe/rezervacije i komentiranja)
- "view_all_users" – hrv. Pregled svih korisnika
- "change_user_group" – hrv. Promjena korisničke grupe
- "delete_every_comment" – hrv. Brisanje svakog komentara
- "view_data_for_each_user" - hrv. Pregled podataka svih korisnika

Kod za kreiranje tablice „worker“je sljedeći:

```
CREATE TABLE "worker" (
    "approve_rent_to_every_user" bool NOT NULL DEFAULT True,
    "confirm_return" bool NOT NULL DEFAULT True,
    "cancel_reservation_for_every_user" bool NOT NULL DEFAULT True,
    "view_all_users" bool NOT NULL DEFAULT True,
    "view_data_for_each_user" bool NOT NULL DEFAULT True,

)
INHERITS ("registered_user");
```

Tablica „worker“ (hrv. Radnik) je tablica koja sadrži sva prava koje posjeduje Radnik. Svi atributi su također tipa **bool** odnosno imaju stanja true (hrv. Točno) ili false (hrv. Lažno).
Kod tablice „worker“ se također javlja nasljeđivanje svih prava koje ima regisrirani korisnik kao i u tablici „admin“.
Prava,specifična za radnika, su sljedeća:

- "approve_rent_to_every_user" - hrv. Odobravanje rezervacije svakom korisniku (to uključuje i samog radnika)
- "confirm_return" - hrv. Zaprimanje filma (fizički povratak filma u videoteku i evidencija u sustav)
- "cancel_reservation_for_every_user" - hrv. Otkazivanje rezervacije za svakog korisnika
- "view_all_users" - hrv. Pregled svih korisnika
- "view_data_for_each_user" - hrv. Pregled podataka svih korisnika

8.2.2 Kreiranje veza između tablica

U nastavku je prikazan kod za kreiranje veza odnosno vanjskih ključeva između tablica.

```
ALTER TABLE "users" ADD CONSTRAINT "Ref_users_to_roles" FOREIGN KEY
("role_id")
    REFERENCES "group"("role_id")
    MATCH SIMPLE
    ON DELETE RESTRICT
    ON UPDATE CASCADE
    NOT DEFERRABLE;

ALTER TABLE "movies" ADD CONSTRAINT "Ref_movies_to_directors" FOREIGN KEY
("director_id")
    REFERENCES "directors"("director_id")
    MATCH SIMPLE
    ON DELETE RESTRICT
    ON UPDATE CASCADE
    NOT DEFERRABLE;

ALTER TABLE "movies_has_movie_genres" ADD CONSTRAINT
"Ref_movies_has_movie_genres_to_movies" FOREIGN KEY ("movie_id")
    REFERENCES "movies"("movie_id")
```

```
    MATCH SIMPLE
    ON DELETE CASCADE
    ON UPDATE CASCADE
    NOT DEFERRABLE;

ALTER TABLE "movies_has_movie_genres" ADD CONSTRAINT
"Ref_movies_has_movie_genres_to_movie_genres" FOREIGN KEY ("genre_id")
    REFERENCES "movie_genres"("genre_id")
    MATCH SIMPLE
    ON DELETE RESTRICT
    ON UPDATE CASCADE
    NOT DEFERRABLE;

ALTER TABLE "users_movies" ADD CONSTRAINT "Ref_movies_has_users_to_users"
FOREIGN KEY ("user_id")
    REFERENCES "users"("user_id")
    MATCH SIMPLE
    ON DELETE RESTRICT
    ON UPDATE CASCADE
    NOT DEFERRABLE;

ALTER TABLE "users_movies" ADD CONSTRAINT "Ref_users_movies_to_movie_copy"
FOREIGN KEY ("copy_id")
    REFERENCES "movie_copy"("copy_id")
    MATCH SIMPLE
    ON DELETE RESTRICT
    ON UPDATE CASCADE
    NOT DEFERRABLE;

ALTER TABLE "movie_copy" ADD CONSTRAINT "Ref_movie_copy_to_movies" FOREIGN
KEY ("movie_id")
    REFERENCES "movies"("movie_id")
    MATCH SIMPLE
    ON DELETE CASCADE
    ON UPDATE CASCADE
    NOT DEFERRABLE;


ALTER TABLE "users_has_restrictions" ADD CONSTRAINT
"Ref_users_has_restrictions_to_users" FOREIGN KEY ("user_id")
    REFERENCES "users"("user_id")
    MATCH SIMPLE
    ON DELETE NO CASCADE
    ON UPDATE NO CASCADE
    NOT DEFERRABLE;

ALTER TABLE "users_has_restrictions" ADD CONSTRAINT
"Ref_users_has_restrictions_to_restrictions" FOREIGN KEY ("id_stats")
    REFERENCES "restrictions"("id_stats")
    MATCH SIMPLE
    ON DELETE NO CASCADE
    ON UPDATE NO CASCADE
    NOT DEFERRABLE;

ALTER TABLE "activation_link_hashes" ADD CONSTRAINT
"Ref_activation_link_hashes_to_users" FOREIGN KEY ("user_id")
    REFERENCES "users"("user_id")
    MATCH SIMPLE
    ON DELETE CASCADE
    ON UPDATE CASCADE
    NOT DEFERRABLE;
```

```
ALTER TABLE "whishlist" ADD CONSTRAINT "Ref_users_has_movies_to_users"
FOREIGN KEY ("user_id")
    REFERENCES "users"("user_id")
    MATCH SIMPLE
    ON DELETE NO CASCADE
    ON UPDATE NO CASCADE
    NOT DEFERRABLE;

ALTER TABLE "whishlist" ADD CONSTRAINT "Ref_users_has_movies_to_movies"
FOREIGN KEY ("movie_id")
    REFERENCES "movies"("movie_id")
    MATCH SIMPLE
    ON DELETE NO CASCADE
    ON UPDATE NO CASCADE
    NOT DEFERRABLE;

ALTER TABLE "overdues" ADD CONSTRAINT "Ref_overdues_to_users" FOREIGN KEY
("user_id")
    REFERENCES "users"("user_id")
    MATCH SIMPLE
    ON DELETE RESTRICT
    ON UPDATE CASCADE
    NOT DEFERRABLE;

ALTER TABLE "overdues" ADD CONSTRAINT "Ref_overdues_to_movie_copy" FOREIGN
KEY ("copy_id")
    REFERENCES "movie_copy"("copy_id")
    MATCH SIMPLE
    ON DELETE RESTRICT
    ON UPDATE CASCADE
    NOT DEFERRABLE;


ALTER TABLE "comments" ADD CONSTRAINT "Ref_comments_to_users" FOREIGN KEY
("user_id")
    REFERENCES "users"("user_id")
    MATCH SIMPLE
    ON DELETE CASCADE
    ON UPDATE CASCADE
    NOT DEFERRABLE;

ALTER TABLE "comments" ADD CONSTRAINT "Ref_comments_to_movies" FOREIGN KEY
("movie_id")
    REFERENCES "movies"("movie_id")
    MATCH SIMPLE
    ON DELETE CASCADE
    ON UPDATE CASCADE
    NOT DEFERRABLE;

ALTER TABLE "users_has_restrictions" ADD CONSTRAINT
"Ref_users_has_restrictions_to_users" FOREIGN KEY ("user_id")
        REFERENCES "users"("user_id")
    MATCH SIMPLE
    ON DELETE CASCADE
    ON UPDATE CASCADE
    NOT DEFERRABLE;
```

8.3 Okidači

Kako bi se smanjio broj upita prema bazi podataka te automatizirale neke radnje korišteni su okidači (eng. trigger) kako bi aktivirali određenu pohranjenu proceduru u određenom trenutku. U nastavku slijedi prikaz korištenih triggera te njima pripadajućih pohranjenih procedura od kojih će svaki/e biti detaljno pojašnjen.

NAPOMENA: S obzirom na to da je postupak tj. sintaksa kreiranja okidača prethodno objašnjena, ona se ovdje neće ponovno pojašnjavati nego će se samo objasniti svrha pojedinog triggera.

```
CREATE TRIGGER user_registered_hash
AFTER INSERT ON users
FOR EACH ROW EXECUTE PROCEDURE generate_user_hash();
```

Navedeni trigger "user_registered_hash" se brine o tome da nakon umetanja novog sloga[4] u tablicu "users" tj. nakon registracije korisnika za svaki red pokrene procedure "generate_user_hash". Ovdje je ključna riječ **nakon** jer se ne želi unositi podatke za nepostojećeg korisnika ukoliko umetanje odnosno proces registracije ne uspije.

```
CREATE TRIGGER user_request_hash
BEFORE INSERT ON activation_link_hashes
FOR EACH ROW EXECUTE PROCEDURE generate_user_hash_by_request();
```

Trigger "user_request_hash" se aktivira prije umetanja sloga u tablicu "activation_link_hashes" te se poziva procedura "generate_user_hash_by_request" za svaki slog koji planiramo unjeti.

```
CREATE TRIGGER log_registration
AFTER INSERT ON users
FOR EACH ROW
EXECUTE PROCEDURE log_user_registration();
```

Trigger „log_registration" se aktivira nakon umetanja sloga u tablici „users" te se za svaki novoumetnuti red poziva procedura „log_user_registration".

```
CREATE TRIGGER acc_activation
```

[4] Pod pojmom "slog" podrazumijeva se red u tablici baze podataka

```
AFTER UPDATE ON users
FOR EACH ROW
EXECUTE PROCEDURE log_acc_activation();
```

Trigger „acc_activation“ se aktivira nakon ažuriranja sloga u tablici „users“ te se za svaki novoumetnuti slog poziva procedura „log_acc_activation“.

```
CREATE TRIGGER permission_change
AFTER UPDATE ON users
FOR EACH ROW
EXECUTE PROCEDURE log_acc_permission_change();
```

Trigger „permission_change“ se aktivira nakon ažuriranja sloga u tablici „users“ te se za svaki novoumetnuti slog poziva procedura „log_acc_permission_change“.

```
CREATE TRIGGER user_deleted
AFTER DELETE ON users
FOR EACH ROW
EXECUTE PROCEDURE log_user_deletion();
```

Trigger „user_deleted“ se aktivira nakon brisanja sloga u tablici „users“ te se za svaki obrisani slog poziva procedura „log_user_deletion“.

```
CREATE TRIGGER movie_deleted
AFTER DELETE ON movies
FOR EACH ROW
EXECUTE PROCEDURE log_movie_deletion();
```

Trigger „movie_deleted“ se aktivira nakon brisanja sloga u tablici „movies“ te se za svaki obrisani slog poziva procedura „log_movie_deletion“.

```
CREATE TRIGGER log_restriction
AFTER INSERT ON users_has_restrictions
FOR EACH ROW
EXECUTE PROCEDURE log_user_restriction();
```

Trigger „log_restriction“ se aktivira nakon umetanja sloga u tablicu „users_has_restrictions“ te se za svaki umetnuti slog poziva procedura „log_user_restriction“.

```
CREATE TRIGGER log_restriction_delete
AFTER DELETE ON users_has_restrictions
FOR EACH ROW
EXECUTE PROCEDURE log_user_restriction_delete();
```

Trigger „log_restriction_delete“ se aktivira nakon brisanja sloga iz tablice „users_has_restrictions“ te se za svaki obrisani slog poziva prodcedura „log_user_restriction_delete“.

```
CREATE TRIGGER user_got_restriction
BEFORE INSERT ON users_has_restrictions
FOR EACH ROW
EXECUTE PROCEDURE define_end_date_restriction();
```

Trigger „user_got_restriction“ se aktivira prije umetanja sloga u tablicu „users_has_restrictions“ te se za svaki umetnuti slog poziva procedura „define_end_date_restriction“.

```
CREATE TRIGGER user_rented_borrowed
BEFORE INSERT ON users_movies
FOR EACH ROW
EXECUTE PROCEDURE define_end_date_borrow_rent();
```

Trigger „user_rented_borrowed“ se aktivira prije umetanja sloga u tablicu „users_movies“ te se za svaki umetnuti slog poziva procedura „define_end_date_borrow_rent“.

```
CREATE TRIGGER user_activated_acc
AFTER UPDATE ON users
FOR EACH ROW
EXECUTE PROCEDURE delete_activation_link();
```

Trigger „user_activated_acc“ se aktivira poslije ažuriranja sloga u tablici „users“ te se za svaki ažurirani slog poziva procedura „define_end_date_borrow_rent“.

```
CREATE TRIGGER movie_borrowed
AFTER UPDATE ON movie_copy
FOR EACH ROW
EXECUTE PROCEDURE increase_total_borrow_count();
```

Trigger „movie_borrowed“ se aktivira poslije ažuriranja sloga u tablici „movie_copy“ te se za svaki ažurirani slog poziva procedura „increase_total_borrow_count“.

```
CREATE TRIGGER movie_rented_pdv
BEFORE INSERT ON users_movies
FOR EACH ROW
EXECUTE PROCEDURE calculate_price_with_PDV();
```

Trigger „movie_rented_pdv“ se aktivira poslije umetanja sloga u tablicu „users_movies“ te se za svaki umetnuti slog poziva procedura „calculate_price_with_PDV“.

```
CREATE TRIGGER movie_rented_set_status
AFTER INSERT ON users_movies
FOR EACH ROW
EXECUTE PROCEDURE set_movie_copy_status_rented();
```

Trigger „movie_rented_set_status“ se aktivira poslije umetanja sloga u tablicu „users_movies“ te se za svaki umetnuti slog poziva procedura „set_movie_copy_status_rented“.

```
CREATE TRIGGER movie_freed
AFTER UPDATE OR DELETE ON users_movies
FOR EACH ROW EXECUTE PROCEDURE free_up_movie_copy();
```

Trigger „movie_freed“ se aktivira poslije ažuriranja sloga u tablici „users_movies“ te se za svaki ažurirani slog poziva procedura „free_up_movie_copy“.

```
CREATE TRIGGER overdue
BEFORE UPDATE ON users_movies
FOR EACH ROW
EXECUTE PROCEDURE set_overdues();
```

Trigger „overdue“ se aktivira prije ažuriranja sloga u tablici „users_movies“ te se za svaki slog prije ažurianja poziva procedura „set_overdues“.

```
CREATE TRIGGER verify_user_registration
BEFORE INSERT ON users
FOR EACH ROW
EXECUTE PROCEDURE verify_user_data();
```

Trigger „verify_user_registration“ se aktivira prije umetanja sloga u tablicu „users“ te se za svaki slog prije umetanja poziva procedura „verify_user_data“.

```
CREATE TRIGGER a_verify_rented_borrow_valid
BEFORE INSERT ON users_movies
FOR EACH ROW
EXECUTE PROCEDURE verify_if_borrow_rent_valid();
```

Trigger „a_verify_rented_borrow_valid“ se aktivira prije umetanja sloga u tablicu „users_movies“ te se za svaki slog prije umetanja poziva procedura „verify_if_borrow_rent_valid“.

```
CREATE TRIGGER can_user_comments
BEFORE INSERT ON comments
FOR EACH ROW
EXECUTE PROCEDURE verify_if_commenting_valid();
```

Trigger „can_user_comments“ se aktivira prije umetanja sloga u tablicu „comments“ te se za svaki slog prije umetanja poziva procedura „verify_if_commenting_valid“.

```
CREATE TRIGGER can_user_be_deleted
BEFORE DELETE ON users
FOR EACH ROW
EXECUTE PROCEDURE verify_if_user_can_be_deleted();
```

Trigger „can_user_be_deleted“ se aktivira prije brisanja sloga iz tablice „users“ te se za svaki slog prije brisanja poziva procedura „verify_if_user_can_be_deleted“.

```
CREATE TRIGGER can_movie_be_deleted
BEFORE DELETE ON movies
FOR EACH ROW
EXECUTE PROCEDURE verify_if_movie_can_be_deleted();
```

Trigger „can_movie_be_deleted“ se aktivira prije brisanja sloga iz tablice „movies“ te se za svaki slog prije brisanja poziva procedura „verify_if_movie_can_be_deleted“.

8.4 Funkcije – pohranjene procedure

Generate_user_hash

```
CREATE OR REPLACE FUNCTION generate_user_hash()
RETURNS TRIGGER AS $$
DECLARE u_id integer;
DECLARE hash char(32);
DECLARE valid timestamp;
DECLARE durationR integer;
BEGIN
hash := md5(NOW()::char);
durationR := 3;
u_id := NEW.user_id;
valid := NOW() + ((durationR || ' days')::interval);
INSERT INTO activation_link_hashes VALUES (DEFAULT, u_id, hash, valid);
RETURN NEW;
END;
$$
LANGUAGE plpgsql;
```

Pohranjena procedura „generate_user_hash“ koristi se prilikom umetanja novog sloga u tablicu „users“ odnosno registracije novog korisnika. Ona kreira md5[5] hash na temelju trenutnog vremena s kojim korisnik aktivira svoj račun te vrijeme do kojeg je šifra valjana (u ovom slučaju 3 dana). Također u varijablu „u_id“ pohranjuje jedinstveni identifikator korisnika pomoću ključne riječi **NEW** dobiven iz upita koji je aktivirao trigger. Nakon što su sve vrijednosti pohranjene pomoću upita, unosi korisnički identifikator i md5 hash u tablicu „activation_link_hashes“.

```
CREATE OR REPLACE FUNCTION generate_user_hash_by_request()
RETURNS TRIGGER AS $$
DECLARE durationR integer;
BEGIN
durationR := 3;
NEW.hash := md5(NOW()::char);
NEW.valid_until := NOW() + ((durationR || ' days')::interval);
```

[5] „Potpis“ koji se može primjeniti na neki sadržaj kako bi se osigurala njegova autentičnost (Izvor: https://www.quora.com/What-is-MD5-in-laymans-terms)

```
RETURN NEW;
END;
$$
LANGUAGE plpgsql;
```

Kako bi se smanjila mogućnost manipulacije s aktivacijskim šiframa aktiviranjem triggera „user_request_hash“ funkcija „generate_user_hash_by_request“ generira aktivacijsku šifru i vrijeme do kad aktivacijska šifra vrijedi (u ovom slučaju vrijedi 3 dana) te ih uz pomoć ključnih riječi **NEW.hash** i **NEW.valid_until** zamjenjuje s trenutnom vrijednosti upita prije nego što se isti izvrši.

```
CREATE OR REPLACE FUNCTION log_user_registration()
RETURNS TRIGGER AS $$
BEGIN

INSERT INTO log VALUES (DEFAULT, NEW.user_id, 'registration', 'Korisnik
"'||NEW.username|| '"se registrirao', NOW());
RETURN NEW;
END;

$$
LANGUAGE plpgsql;
```

Funkcija „log_user_registration“ se pokreće nakon umetanja sloga u tablicu „users“ odnosno nakon što je registracija uspješno prošla te zapisuje u tablicu „log“ podatke o novoregistriranom korisniku (username, datum registracije).

```
CREATE OR REPLACE FUNCTION log_acc_activation()
RETURNS TRIGGER AS $$
DECLARE activated bool;
BEGIN
activated := NEW.activated;
IF activated AND OLD.activated = false THEN
INSERT INTO log VALUES (DEFAULT, NEW.user_id, 'activation', 'Korisnik
"'||NEW.username|| '" aktivirao račun', NOW());
END IF;
RETURN NEW;
END;
$$
LANGUAGE plpgsql;
```

Funkcija „log_user_registration“ se pokreće nakon izmjene sloga u tablici „users“ odnosno nakon što se atribut „activated“ promjeni iz starog stanja „false“ (OLD.activated) u novo stanje true (NEW.activated) što govori da je korisnik aktivirao račun. Nakon uspješne aktivacije funkcija zapisuje u tablicu „log“ podatke o aktivaciji korisnika (username, datum aktivacije).

```
CREATE OR REPLACE FUNCTION log_acc_permission_change()
RETURNS TRIGGER AS $$
DECLARE newpermission text;
DECLARE oldpermission text;

BEGIN

IF OLD.role_id <> NEW.role_id THEN
newpermission := (SELECT group_name FROM "group" WHERE role_id =
NEW.role_id);
oldpermission := (SELECT group_name FROM "group" WHERE role_id =
OLD.role_id);

INSERT INTO log VALUES (DEFAULT, NEW.user_id, 'group', 'Korisnik
"'||NEW.username|| '" prešao iz razine "'||oldpermission ||'" u
"'||newpermission||'" razinu korisnika', NOW());
END IF;
RETURN NEW;
END;
$$
LANGUAGE plpgsql;
```

Funckija „log_acc_permission_change“ zapisuje u tablicu „log“ podatke o promjeni korisničkih prava. Funkcija prvo provjerava da li su se prava promjenila te ako jesu zapisuje ime nove grupe u varijablu „newpermission“. Kako bi evidencija bila potpuna zapisuje se i ime stare grupe i to u varijablu „oldpermission“ te se nakon toga unose podaci u tablicu „log“.

```
CREATE OR REPLACE FUNCTION log_user_deletion()
RETURNS TRIGGER AS $$
BEGIN
INSERT INTO log VALUES (DEFAULT, OLD.user_id, 'deletionUser','Korisnik
korisničkog imena "'
|| OLD.username||'" je obrisan iz baze', NOW());
RETURN NEW;
END;
$$
LANGUAGE plpgsql;
```

Funkcija „log_user_deletion“ se aktivira triggerom nakon brisanja sloga iz tablice „users“ te u tablicu „log“ zapisuje podatke o obrisanom korisniku (njegov stari korisnički identifikator, username, te vrijeme brisanja).

```
CREATE OR REPLACE FUNCTION log_movie_deletion()
RETURNS TRIGGER AS $$
BEGIN
INSERT INTO log VALUES (DEFAULT, -1, 'deletion', 'Film "'||OLD.title ||'" je
obrisan iz baze', NOW());
RETURN NEW;
END;
$$
LANGUAGE plpgsql;
```

Funkcija „log_movie_deletion“ se aktivira triggerom nakon brisanja sloga iz tablice „movies“ te u tablicu „log“ zapisuje podatke o obrisanom filmu (-1 koji označava da je to film, title, te vrijeme brisanja).

```
CREATE OR REPLACE FUNCTION log_user_restriction()
RETURNS TRIGGER AS $$
DECLARE restriction text;
DECLARE username text;
BEGIN
restriction := (SELECT type
FROM restrictions AS r
JOIN users_has_restrictions AS ur
ON r.id_stats = ur.id_stats WHERE user_id = NEW.user_id AND ur.id_stats =
NEW.id_stats AND ur.id_stats = NEW.id_stats
);
username := (SELECT users.username FROM users WHERE user_id = NEW.user_id);
IF restriction = 'comments' THEN
INSERT INTO log VALUES (DEFAULT, NEW.user_id, 'comments', 'Korisnik
"'||username|| '" dobio zabranu komentiranja', NOW());
ELSE
INSERT INTO log VALUES (DEFAULT, NEW.user_id, 'borrow', 'Korisnik
"'||username|| '" dobio zabranu posudbe', NOW());
END IF;
RETURN NEW;
END;
$$
LANGUAGE plpgsql;
```

Funkcija koja se pokreće aktivacijom triggera poslije unosa sloga u tablicu „users_has_restrictions“. Ona u tablicu „log“, zapisuje koju vrstu zabrane je korisnik dobio te vrijeme dobivanja. Prvo na temelju korisničkog identifikatora te identifikatora zabrane, koji su unešeni u tablicu „users_has_restrictions“ dohvaća ime zabrane i na temelju imena zabrane upisuje odgovarajuće vrijednosti u tablicu „log“.

```
CREATE OR REPLACE FUNCTION log_user_restriction_delete()
RETURNS TRIGGER AS $$
DECLARE restriction text;
DECLARE username text;
BEGIN
IF EXISTS(SELECT * FROM users WHERE user_id = OLD.user_id) THEN
restriction := (SELECT type
FROM restrictions AS r
JOIN users_has_restrictions AS ur
ON r.id_stats = ur.id_stats WHERE user_id = OLD.user_id AND ur.id_stats =
OLD.id_stats
);
username := (SELECT users.username FROM users WHERE user_id = OLD.user_id);

IF restriction = 'comments' THEN
INSERT INTO log VALUES (DEFAULT, OLD.user_id, 'comments', 'Korisniku
"'||username|| '" istekla zabrana komentiranja', NOW());
ELSE
INSERT INTO log VALUES (DEFAULT, OLD.user_id, 'borrow', 'Korisniku
"'||username|| '" istekla zabrana posudbe', NOW());
```

```
END IF;
END IF;
RETURN NEW;
END;
$$
LANGUAGE plpgsql;
```

Funkcija se pokreće aktivacijom triggera poslije brisanja sloga iz tablice „users_has_restrictions“. Prije svega provjerava da li korisnik postoji (jer se može dogoditi da se briše korisnik pa se zabrane također brišu). Nakon toga u tablicu „log“, zapisuje koja je vrsta zabrane ukinuta odnosno koja je zabrana korisniku istekla. Funkcija prvo na temelju korisničkog identifikatora, te identifikatora zabrane, koji su izbrisani iz tablice „users_has_restrictions“ dohvaća ime obrisane zabrane te nakon toga upisuje odgovarajuće vrijednosti u tablicu „log“.

```
CREATE OR REPLACE FUNCTION define_end_date_restriction()
RETURNS TRIGGER AS $$
DECLARE durationR integer;
DECLARE has_restriction bool;
BEGIN
has_restriction := (EXISTS(SELECT * FROM users_has_restrictions WHERE user_id
= NEW.user_id AND id_stats = NEW.id_stats));
if has_restriction THEN
RAISE EXCEPTION 'Korisnik već ima tekuću zabranu ovog tipa!'
      USING HINT = '';
END IF;
durationR := (SELECT duration FROM restrictions WHERE id_stats =
NEW.id_stats);
NEW.date_ending := NOW() + ((durationR || ' days')::interval);
RETURN NEW;
END;
$$
LANGUAGE plpgsql;
```

Funkcija „define_end_date_restriction“ prvo provjerava da li prilikom zadavanja zabrane korisniku korisnik već ima zabranu tog tipa. Ukoliko ima podiže odgovarajuću iznimku te onemogućuje dodavanje zabrane. Ukoliko zabrana ne postoji definira datum do kada zabrana traje koristeći već definirane podatke unutar baze te na trenutno vrijeme nadodaje trajanje zabrane. Zatim vrijednosti postavlja u upit koji je aktivirao okidač koji je naposlijetku pozvao ovu funkciju koristeći objekt „NEW“ (sadrži nove vrijednosti sloga).

```
CREATE OR REPLACE FUNCTION define_end_date_rent()
RETURNS TRIGGER AS $$
DECLARE durationR integer;
BEGIN
durationR := 3;
NEW.date_due := NOW() + ((durationR || ' days')::interval);
RETURN NEW;
END;
$$
LANGUAGE plpgsql;
```

Funkcija „define_end_date_rent“ definira krajnje vrijeme valjanosti rezervacije. Aktivira se prilikom unosa sloga u tablicu „user_movies“. Inkrementira trenutno vrijeme za 3 dana te tim datumom zamjenjuje datum dobiven u upitu uz pomoć objekta „NEW“.

```
CREATE OR REPLACE FUNCTION define_end_date_borrow()
RETURNS TRIGGER AS $$
DECLARE durationR integer;
DECLARE date_duee timestamp
BEGIN
durationR := 3;
IF OLD.status = 'reserved' AND NEW.status = 'borrowed' THEN
date_duee := NOW() + ((durationR || ' days')::interval);

UPDATE users_movies SET date_due = date_duee, date_borrowed = NOW() WHERE
copy_id = NEW.copy_id;

END IF;
RETURN NEW;
END;
$$
LANGUAGE plpgsql;
```

Funkcija „define_end_date_borrow“ definira krajnje vrijeme valjanosti posudbe. Aktivira se prilikom ažuriranja sloga u tablici „movie_copy“. Provjerava da li je nova vrijednost atributa „status“ jednaka „borrowed“, a stara „reserved“ što znači da je kopija prešla iz rezervirane u posuđenu. Ukoliko je inkrementira trenutno vrijeme za 3 dana te to definirano vrijeme koristi kako bi ažurirala upitom odgovarajući slog u tablici „users_movies“ i postavila krajnji datum posudbe („date_due“) na to vrijeme i datum posudbe („date_borrowed“) na trenutno vrijeme.

```
CREATE OR REPLACE FUNCTION delete_activation_link()
RETURNS TRIGGER AS $$
DECLARE activated_acc bool;

BEGIN
IF OLD.activated = false AND NEW.activated = true THEN
DELETE FROM activation_link_hashes WHERE activation_link_hashes.user_id =
OLD.user_id;
END IF;
RETURN NEW;
END
$$

LANGUAGE plpgsql;
```

Funkcija koja se aktivira nakon ažuriranja sloga u tablici „users“. Funkcija prvo provjerava da li je ažuriran atribut „activated“ unutar tablice „users“. Pošto je atribut standardno postavljan na vrijednost „false“ on će jedino biti izmijenjen kada korisnik aktivira račun. Ukoliko je to slučaj ,odnosno da je stara vrijednost atributa „activated“ lažna, a nova vrijednost istinita, iz tablice

„activation_log_hashes“ brišu se sve aktivacijske šifre koje pripadaju korisniku koji je aktivirao račun.

```
CREATE OR REPLACE FUNCTION increase_total_borrow_count()
RETURNS TRIGGER AS $$
DECLARE movie integer;
DECLARE rentCount integer;

BEGIN

IF OLD.status = 'reserved' AND NEW.status = 'borrowed' THEN
movie := (SELECT movie_id FROM movie_copy WHERE copy_id = NEW.copy_id );

rentCount := (SELECT rent_count FROM movies WHERE movie_id = movie);
rentCount := rentCount + 1;
UPDATE movies SET rent_count = rentCount WHERE movie_id = movie;
END IF;
RETURN NEW;
END;
$$
LANGUAGE plpgsql;
```

Funkcija „increase_total_borrow_count“ se izvršava aktivacijom triggera nakon ažuriranja sloga u tablici „movie_copy“. Funkcija provjerava da li je stara vrijednost atributa „status“ bila jednaka vrijednosti „reserved“ (film je bio rezerviran), a nova vrijednost je jednaka vrijednosti „borrowed“ (film je posuđen). Ukoliko je uvjet zadovoljen dohvaća se trenutni iznos vrijednosti atributa „rent_count“ u tablici „movies“, za odgovarajući film, te se ona poveća za 1 i ažurira.

```
CREATE OR REPLACE FUNCTION calculate_price_with_PDV()
RETURNS TRIGGER AS $$
DECLARE price_noPDV real;
DECLARE movie integer;
BEGIN

movie := (SELECT movie_id FROM movie_copy WHERE copy_id = NEW.copy_id );
price_noPDV := (SELECT price FROM movies WHERE movie_id = movie);

NEW.price_pdv = round((price_noPDV * ((25.00/100.00)+ 1))::numeric,2);
RETURN NEW;

END;
$$

LANGUAGE plpgsql;
```

Pošto se tablicu „movies“ pohranjuju informacije o cijeni koja je bez PDV-a prilikom posudbe filma bitno je da je ta cijena cijena s PDV-om. Stoga je kreirana funkcija koja se aktivira prije unosa sloga u tablicu „users_movies“. Funkcija dohvaća cijenu filma bez PDV-a te tu cijenu množi sa stopom PDV-a od 25%. Nakon izračuna cijena definirana u upitu se zamjenjuje s novoizračunatom cijenom te se slog unosi u tablicu.

```
CREATE OR REPLACE FUNCTION set_movie_copy_status_rented()
RETURNS TRIGGER AS $$
BEGIN

UPDATE movie_copy SET status = 'reserved' WHERE copy_id = NEW.copy_id;

RETURN NEW;

END;
$$

LANGUAGE plpgsql;
```

Funkcija „set_movie_copy_status_rented“ se brine o tome da postavi kopiju filma na status reservirane („reserved“) jer unos u tablicu „users_movies“ podrazumijeva da se kopija rezervirala jer je postupak taj da se kopija filma prvo rezervira, a zatim posudi. Funkcija se pokreće aktivacijom triggera „movie_rented_set_status“.

```
CREATE OR REPLACE FUNCTION free_up_movie_copy()
RETURNS TRIGGER AS $$
BEGIN
IF TG_OP = 'DELETE' THEN
UPDATE movie_copy SET status = 'available' WHERE copy_id = OLD.copy_id;

ELSIF TG_OP = 'UPDATE' THEN
IF OLD.state = false AND NEW.state = TRUE THEN
UPDATE movie_copy SET status = 'available' WHERE copy_id = OLD.copy_id;
END IF;
END IF;
RETURN NEW;

END;
$$
LANGUAGE plpgsql;
```

Funkcija „free_up_movie_copy“ ima za zadatak da postavi kopiju filma na slobodno ukoliko se dogodi da se slog u tablici „users_movies“ ažurira ili obriše. Ukoliko se radi o ažuriranju funkcija provjerava da li je staro stanje atributa „state“ bilo „false“, a novo „true“ (što znači da je film tj. kopija vraćena u videoteku). Ako je, kopija s odgovarajućim identifikatorom postavi se na slobodnu odnosno vrijednost atributa „status“ na „available“. Ukoliko se radi o brisanju nema provjere nego se odgovarajućoj kopiji samo postavi atribut „status“ na „available“.

```
CREATE OR REPLACE FUNCTION set_overdues()
RETURNS TRIGGER AS $$
DECLARE daysOverdue integer;
DECLARE isBorrow bool;

BEGIN
```

```
isBorrow:= (SELECT status = 'borrowed' FROM movie_copy WHERE copy_id =
OLD.copy_id);
IF isBorrow AND NEW.state = false THEN
daysOverdue := (SELECT date_part('days', NOW() - OLD.date_due::timestamp));

IF daysOverdue > 0 THEN

NEW.overdue := daysOverdue * 2.00;
NEW.days_overdue := daysOverdue;

ELSE
NEW.overdue := 0.00;
NEW.days_overdue := 0;
END IF;
END IF;
RETURN NEW;
END;
$$

LANGUAGE plpgsql;
```

Funkcija „set_overdues“ se brine o zakasninama. Ona se koristi kao glavni mehanizam za definiranje zakasnina koje korisnik dobije nepovratkom filma do definiranog datuma. Funkcija prvo provjerava da li se radi o posudbi (jer za rezervacije nema zakasnina, one se automatski brišu). Ukoliko se radi o posudbi provjerava se da li je broj koji se dobije oduzimanjem trenutnog vremena od krajnjeg vremena povratka filma veći od nule. Ako je veći od 0 to je broj dana koliko je korisnik zakasnio s povratkom. Taj se broj tada množi s fiktivnim iznosom zakasnine po danu. Ovdje je to definirano kao decimalni broj 2.00 (2 kune po zakašnjelom danu). Nakon izračuna vrijednosti stupaca „overdue“ i „days_overdue“ zamjenjuje se s izračunatom zakasninom odnosno brojem zakašnjelih dana uz pomoć objekta „NEW“. Ukoliko korisnik nije zakasnio s posudbom odnosno vrijednost dana je 0, vrijednosti stupaca „overdue“ i „days_overdue“ se postavljaju na 0,00.

```
CREATE OR REPLACE FUNCTION verify_user_data()
RETURNS TRIGGER AS $$
DECLARE userExists bool;
DECLARE emailExists bool;

BEGIN

userExists:= (EXISTS(SELECT * FROM users WHERE username = NEW.username));
emailExists:= (EXISTS(SELECT * FROM users WHERE email = NEW.email));
IF userExists THEN
RAISE EXCEPTION 'Korisničko ime zauzeto!'
      USING HINT = 'Odaberite drugo korisničko ime!';
END IF;
IF emailExists THEN
RAISE EXCEPTION 'Email je zauzet!'
      USING HINT = 'Odaberite drugi email!';
END IF;
IF NOT NEW.email LIKE '%@%.%' THEN
RAISE EXCEPTION 'Email nije ispravnog formata!'
```

```
      USING HINT = 'Format je tipa nesto@negdje.negdje';
END IF;
RETURN NEW;
END;
$$

LANGUAGE plpgsql;
```

Funkcija „verify_user_data“ se koristi prije nego se umetne slog u tablicu „users“ odnosno prije nego se završi postupak registracije. Ona provjerava sljedeće:

- Da li se korisnik pokušava registrirati s već postojećim korisničkim imenom. U slučaju potvrde, funkcija podiže iznimku i zabranjuje registraciju. To je vrlo bitno jer ukoliko postoje 2 korisnika s istim korisničkim imenom prilikom prijave neće se znati koji je korisnik ispravan te će prijava biti onemogućena.
- Drugi uvjet koji se provjerava je taj da li je adresa elektroničke pošte zauzeta. Svaki korisnik mora imati različitu adresu elektroničke pošte zbog slanja aktivacijskih ključeva.
- Treći uvjet je taj da je adresa elektroničke pošte ispravnog formata. U slučaju da nije formata %@%.% funkcija zabranjuje registraciju. Kao i prethodni uvjet, taj je također važan zbog slanja aktivacijskih ključeva.

```
CREATE OR REPLACE FUNCTION verify_if_borrow_rent_valid()
RETURNS TRIGGER AS $$

DECLARE exists_movie bool;
DECLARE exists_restriction bool;
BEGIN
exists_restriction := (SELECT EXISTS(SELECT * FROM users_has_restrictions AS
ur JOIN restrictions AS R ON
ur.id_stats = r.id_stats WHERE ur.user_id = NEW.user_id AND type = 'borrow'
));
IF exists_restriction THEN
RAISE EXCEPTION 'Zabrana posudbi i rezervacija! Posudba nemoguća!'
      USING HINT = 'Korisnik ima zabranu posudbe!';
END IF;
exists_movie := (SELECT EXISTS(SELECT * FROM users_movies as um JOIN
movie_copy AS mc ON um.copy_id = mc.copy_id JOIN movies AS m on mc.movie_id =
m.movie_id  WHERE user_id= NEW.user_id  AND m.movie_id = (SELECT movie_id
FROM movie_copy WHERE copy_id = NEW.copy_id) AND state = false));
IF exists_movie THEN
RAISE EXCEPTION 'Već ste rezervirali/posudili ovaj film!'
      USING HINT = '';
END IF;

RETURN NEW;
END;
$$

LANGUAGE plpgsql;
```

Funkcija „verify_if_borrow_rent_valid“ provjerava da li korisnik može posuditi/rezervirati film/kopiju. Provjeravaju se 2 uvjeta:

- Ukoliko korisnik ima zabranu posudbe funkcija zabranjuje posudbu podizanjem iznimke
- Ukoliko korisnik ima već posuđen/rezerviran film/kopiju funkcija mu zabranjuje rezervaciju/posudbu

```
CREATE OR REPLACE FUNCTION verify_if_commenting_valid()
RETURNS TRIGGER AS $$

DECLARE exists_restriction bool;
BEGIN
exists_restriction := (SELECT EXISTS(SELECT * FROM users_has_restrictions AS
ur JOIN restrictions AS R ON
ur.id_stats = r.id_stats WHERE ur.user_id = NEW.user_id AND type = 'comments'
));
IF exists_restriction THEN
RAISE EXCEPTION 'Zabrana komentiranja! Ne možete komentirati!'
      USING HINT = 'Korisnik ima zabranu komentiranja!';
END IF;
RETURN NEW;
END;
$$

LANGUAGE plpgsql;
```

Funkcija koja se pokreće aktivacijom triggera prije nego što dođe do umetanja sloga u tablicu „comments“ odnosno prije komentiranja filma od strane korisnika. Ona provjerava da li korisnik ima zabranu komentiranja tako što provjerava da li za korisnika koji želi objaviti komentar postoji slog u tablici „users_has_restrictions“ gdje je „type“ jednak vrijednosti „comments“. Ako postoji, funkcija podiže iznimku s odgovarajućom porukom te onemogućuje komentiranje korisniku.

```
CREATE OR REPLACE FUNCTION verify_if_user_can_be_deleted()
RETURNS TRIGGER AS $$

DECLARE exists_movie bool;
BEGIN
exists_movie := (SELECT EXISTS(SELECT * FROM users_movies WHERE user_id=
OLD.user_id  AND state = false));
IF exists_movie THEN
RAISE EXCEPTION 'Korisnika se ne može obrisati dok god ima rezerviran/posuden
Film!'
      USING HINT = '';
END IF;

RETURN OLD;
END;
$$

LANGUAGE plpgsql;
```

Funkcija „log_movie_deletion“ se pokreće aktivacijom triggera prije samog brisanja korisnika iz baze podataka. Ona provjerava da li korisnik ima posuđenih filmova te podiže iznimku ako ima te tako odgovarajućom porukom obavještava korisnika sustava zašto taj korisnik ne može biti obrisan.

```
CREATE OR REPLACE FUNCTION verify_if_movie_can_be_deleted()
RETURNS TRIGGER AS $$

DECLARE exists_movie bool;
BEGIN
exists_movie := (SELECT EXISTS(SELECT * FROM movie_copy WHERE movie_id=
OLD.movie_id  AND status <> 'available'));
IF exists_movie THEN
RAISE EXCEPTION 'Film se ne moze obrisati dok god ima rezerviranih/posudenih
kopija!'
      USING HINT = '';
END IF;

RETURN OLD;
END;
$$

LANGUAGE plpgsql;
```

Kako bi korisnika koji ima mogućnost brisanja filmova iz baze podataka obavijestili o nemogućnosti brisanja koristi se funkcija koja pokreće trigger prije samog brisanja filma. Ona provjerava da li trenutno ima posuđenih kopija filma, ako ima podiže iznimku koja korisnika obavještava zašto ne može obrisati odgovarajući film.

9. pgAgent

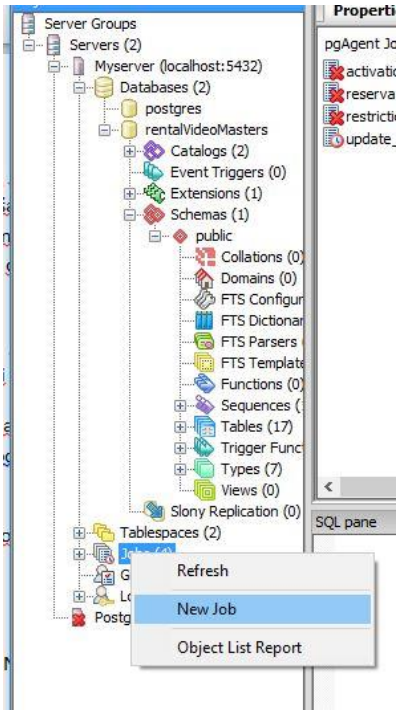
Alat “pgAgent” je agent za zakazivanje zadataka za PostgreSQL, sposoban za pokretanje batch/shell te SQL skripti koje se sastoje od više koraka uz složeni raspored (vrijeme izvršavanja) (pgAdmin, 2016). Drugim riječima, “pgAgent” omogućuje zadavanje određenih zadataka, bilo to SQL naredbi ili batch/shell skripti, koji će se izvršiti u određeno vrijeme, a da je sve to automatizirano.

U ovom slučaju je bilo više problema koje bi bez “pgAgenta” bilo jako teško ili nemoguće riješiti. U nastavku slijedi detaljni opis svakog pojedinog problem te kako su oni riješeni uz pomoć “ pgAgenta”.

No, prikazati će se samo postavke pojedinog zadatka/rasporeda bez objašnjavanja postupka kreiranja istog. Kreiranje zadatka i rasporeda je vrlo lako. Najveći izazov je dobro definirati korake izvršavanja u zadatku.

9.1 Kreiranje zadataka

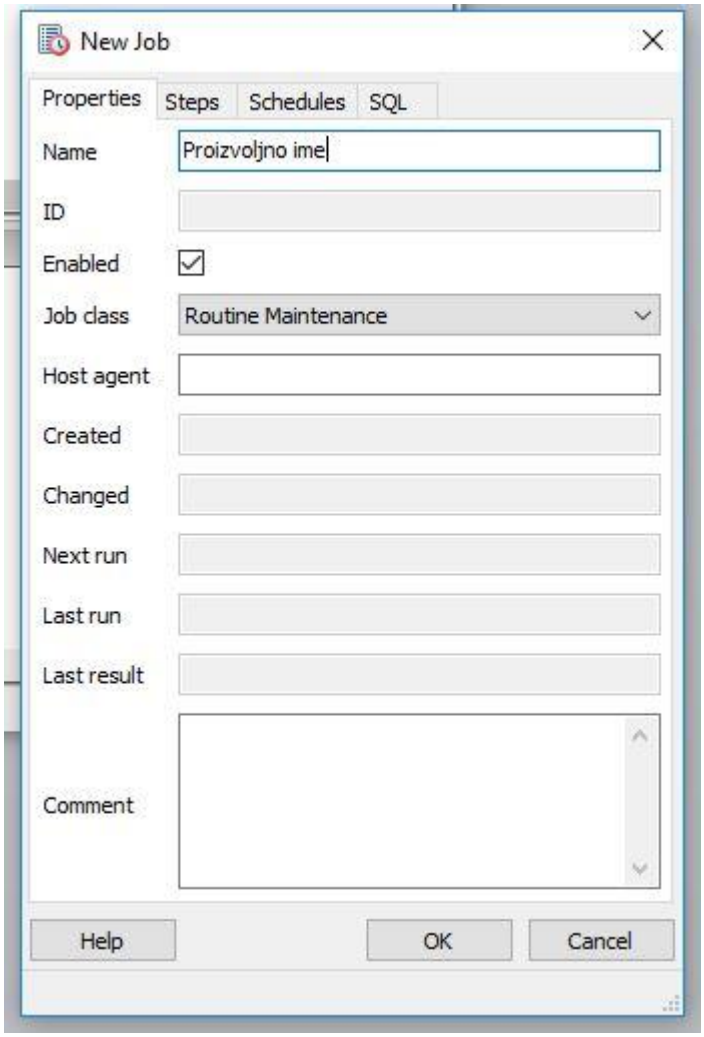
Unutar pgAgenta kreira se zadatak desnim klikom na “Jobs” te se odabere “New Job”.



Slika 9.1.1. Kreiranje zadatka

Izvor: autor

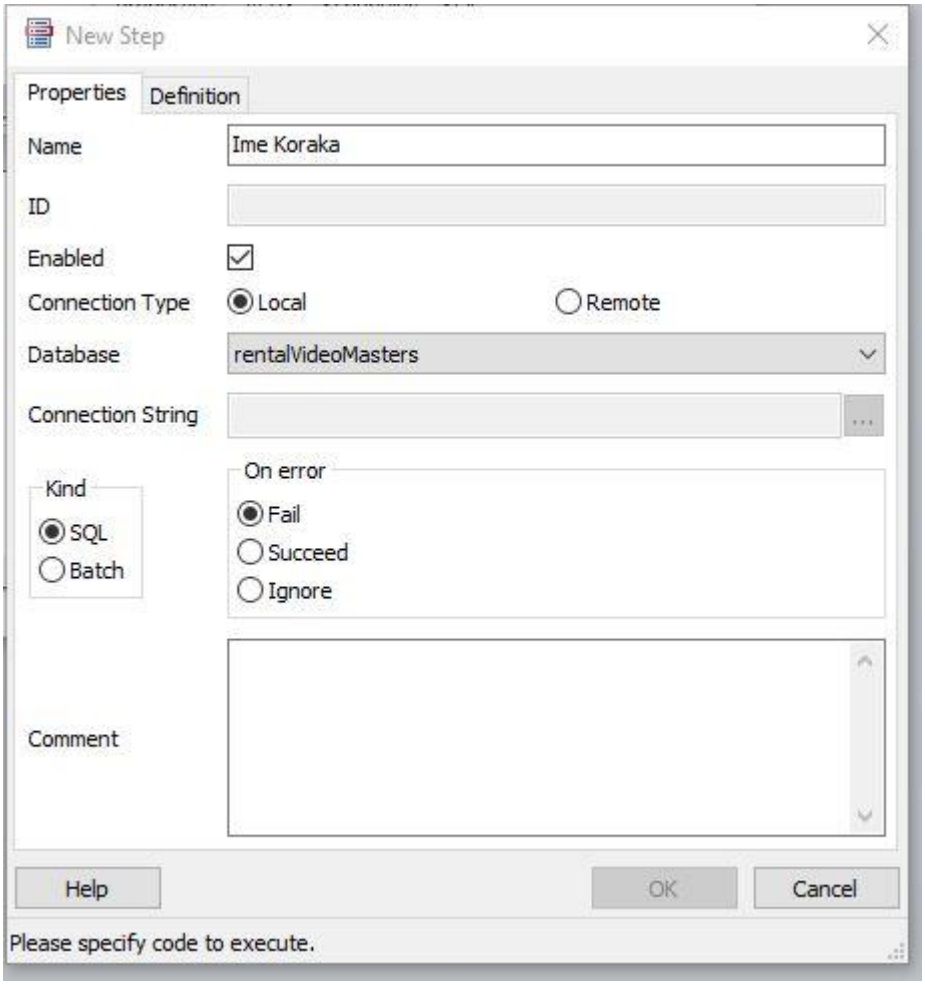
Nakon toga upiše se proizvoljno ime zadatka u polje “Name” unutar taba “Properties” što je prikazano na slici 9.1.2.



Slika 9.1.2. Dodjeljivanje imena zadatku

Izvor: autor

Zatim se odabire tab “Steps” i klikne se na “Add” gdje se otvara prozor kao na slici 9.1.3. te se ovdje proizvoljno upisuje ime koraka.

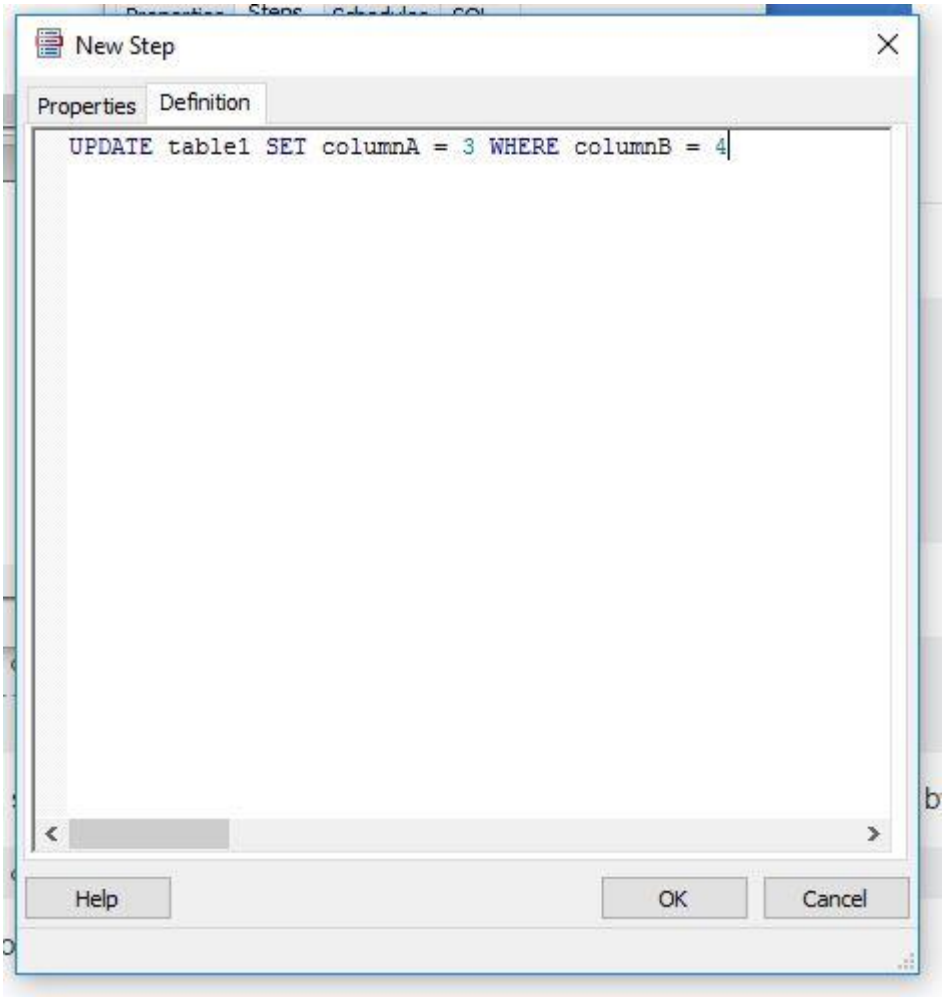


Slika 9.1.3. Dodjeljivanje imena koraku

Izvor: autor

Ako se radi o bazi na lokalnom serveru (kao u ovom slučaju) postavlja se "Connection Type" na "Local" te se odabere baza gdje se primjenjuje ovaj korak. Inače se odabire "Remote" i upisuju potrebni podaci za spajanje na bazu kao što su korisničko ime, lozinka, IP adresa, ime baze i drugo. Ako se izvršavaju SQL upiti i naredbe "Kind" opcija se postavlja na SQL odnosno na "Batch" ukoliko se izvršavaju neke skripte.

Nakon upisanih podataka pod tabom "Definition" upisuje se najbitnija stvar, a to su sami koraci koji će se izvršiti/izvršavati, odnosno definicija samog koraka. Tako npr. za korak se može postaviti slijedeća SQL naredba prikazana na slici 9.1.4. Klikne se na "OK" te je time korak definiran.

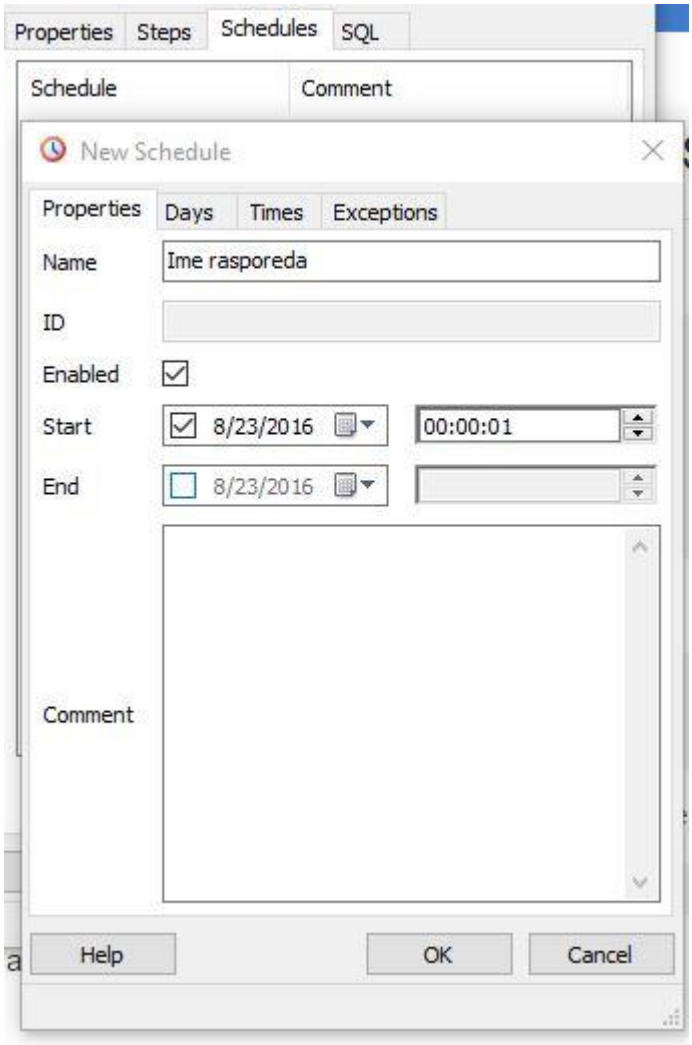


Slika 9.1.4. Definiranje koraka

Izvor: autor

Još je preostalo definirati raspored izvršavanja (tab “Schedules”).

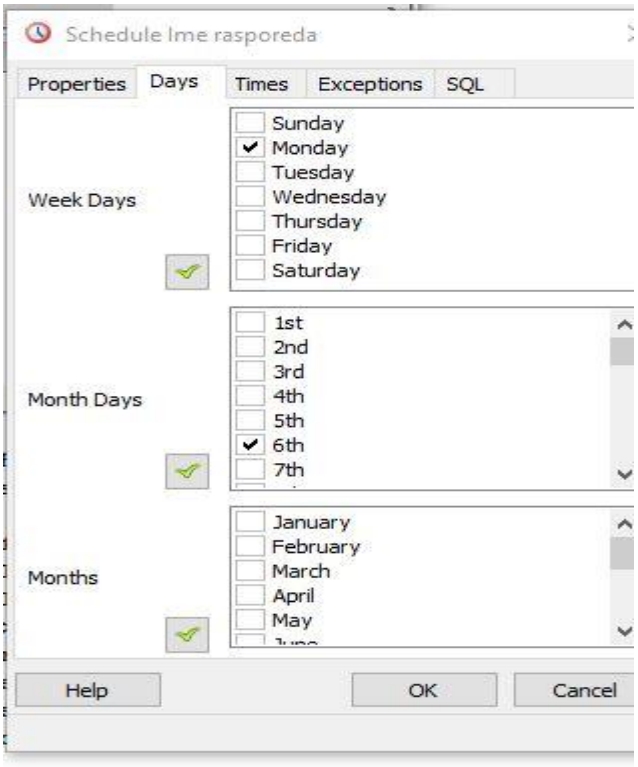
Odabirom “Add” se otvara prozor na slici 9.1.5. gdje se upisuje željeno ime rasporeda izvršavanja te se nude opcije za početno i završno vrijeme (ukoliko se ne unese završno vrijeme izvršavanja, ono je beskonačno).



Slika 9.1.5. Definiranje rasporeda izvršavanja

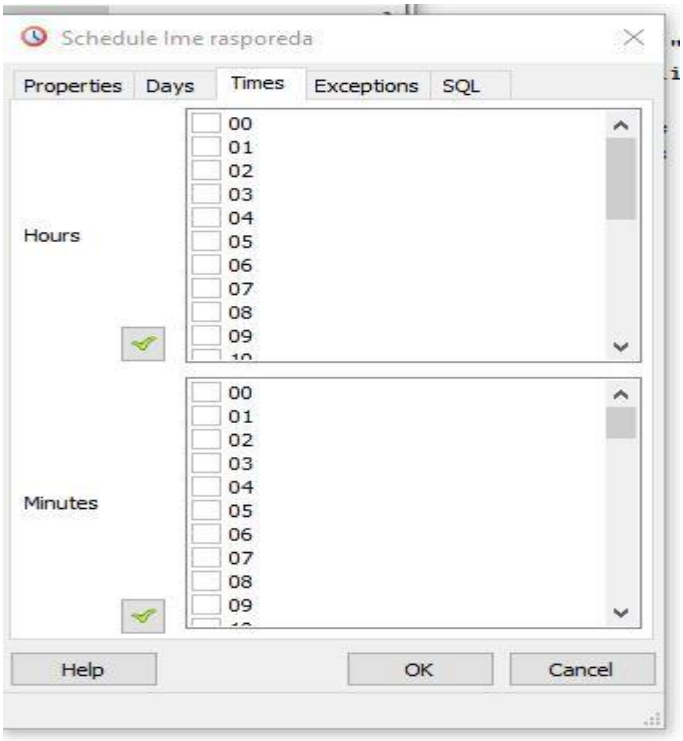
Izvor: autor

Unutar tabova “Days” i “Times” odabiru se dani izvršavanja odnosno vrijeme izvršavanja. Ako se ne odabere, odnosno ne označi “kvačicom” niti jedno polje u grupi podrazumijeva se da su sva označena. Tako se podrazumijeva izvršavanje samo ponedjeljkom ako je to šesti dan u svakom mjesecu što je prikazano na slici 9.1.6., dok je na slici 9.1.7. vidljivo da se podrazumijeva izvršavanje svake minute (najmanja vremenska jedinica je jedna minuta) svakog sata.



Slika 9.1.6. Definiranje dana u rasporedu

Izvor: autor



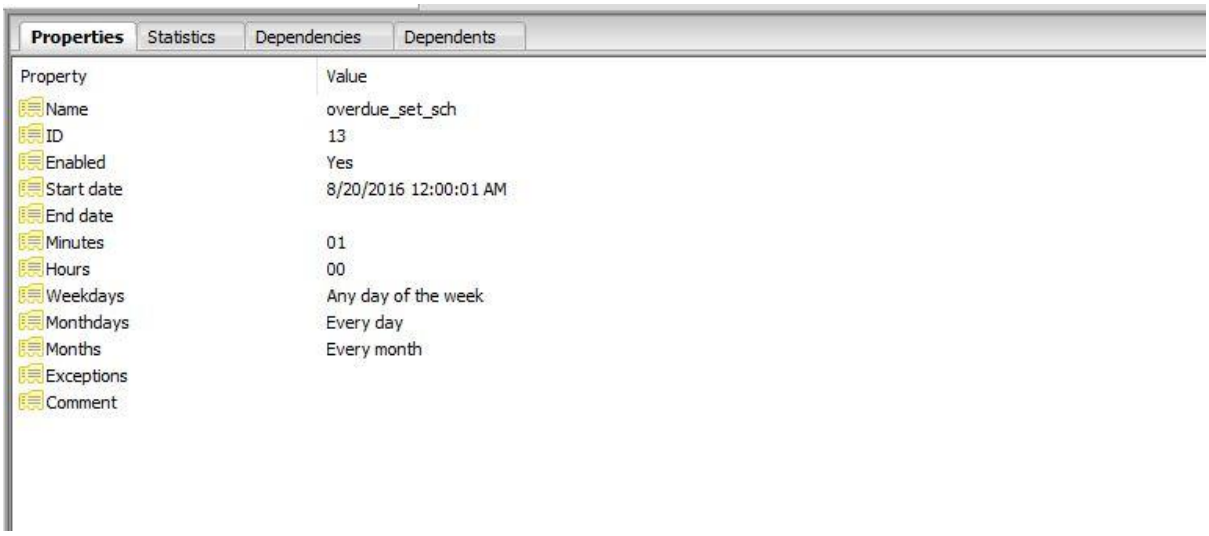
Slika 9.1.7. Definiranje vremena u rasporedu

Izvor: autor

Potvrdom svega unesenoga, zadatak je spreman za izvršavanje. Međutim, ovaj zadatak neće ništa raditi jer je upit definiran nad tablicama koje ne postoje. U nastavku će se objasniti realni problemi u bazi podataka. Prikazat će se samo postavke, a ne i kreiranje zadatka pošto je sam postupak kreiranja objašnjen u tekstu ranije.

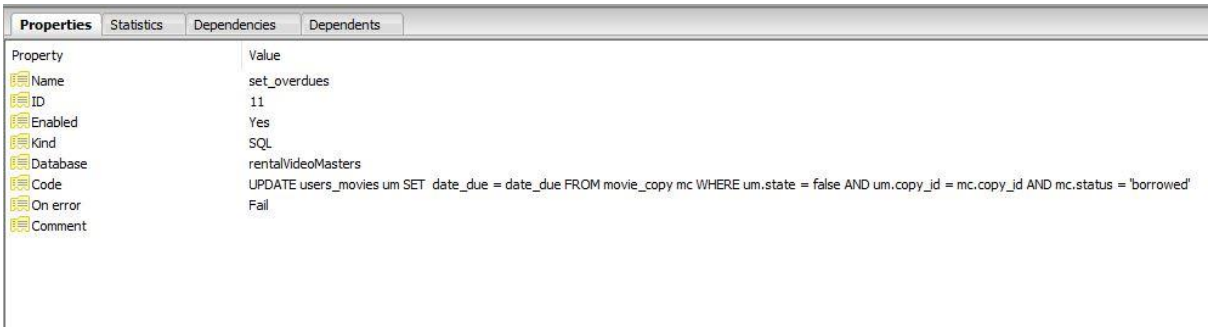
9.2 Problem zakasnina

Kao i kod svake videoteke postoje zakasnine koje se računaju temeljem broja dana nakon kojeg korisnik nije vratio film, od zadanog roka, množeći se s određenim iznosom. Ta funkcionalnost je implementirana i u ovoj aplikaciji, no problem je kako izračunati zakasninu. Malo je nelogično da se prilikom prijave korisnika računa zakasnina jer se može dogoditi da se više nikada ne prijavi na stranicu te bi tako imao 0 kn zakasnine, a i radnici videoteke ne bi imali uvid u realno stanje zakasnina. Upravo zbog te činjenice problem je riješen uz pomoć ponavljajućih zadataka iz pgAgenta. Pošto se zakasnine računaju na dnevnoj bazi dovoljno je bilo u raspored izvršavanja označiti nulti sat i nultu minutu (00:00). Drugim riječima svaku ponoć se računaju zakasnine (prelazak u novi dan) ukoliko je korisnik prekoračio dozvoljeni rok posudbe. Za ovaj ponavljajući zadatak definiran je vremenski raspored kao na slici 9.2.1. te korake kao na slici 9.2.2.



Slika 9.2.1. Vremenski raspored zakasnina

Izvor: autor

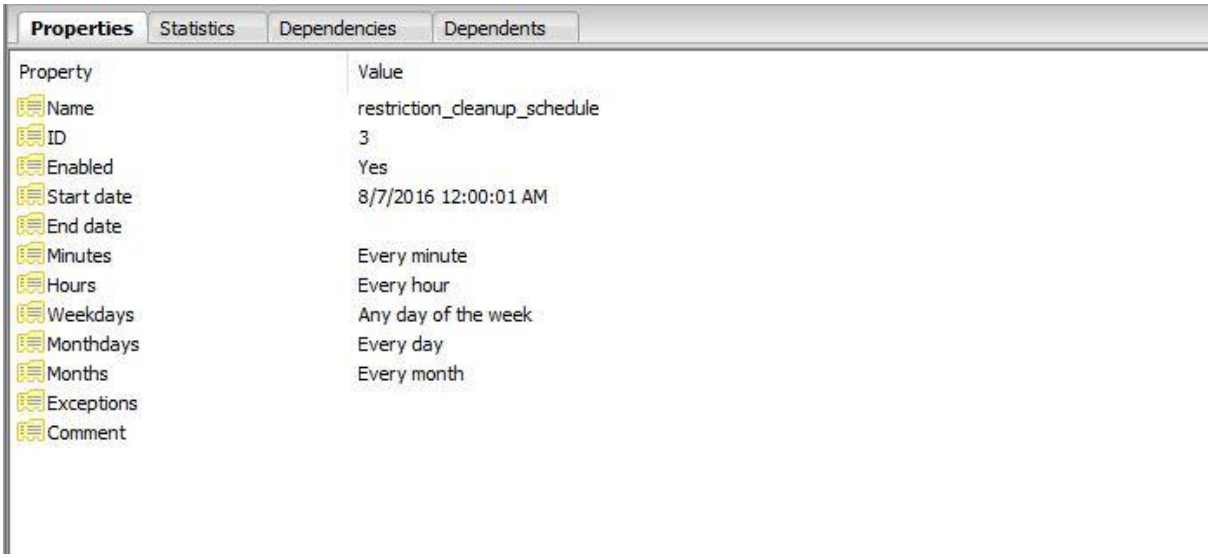


Slika 9.2.2. Definirane zakasnine

Izvor: autor

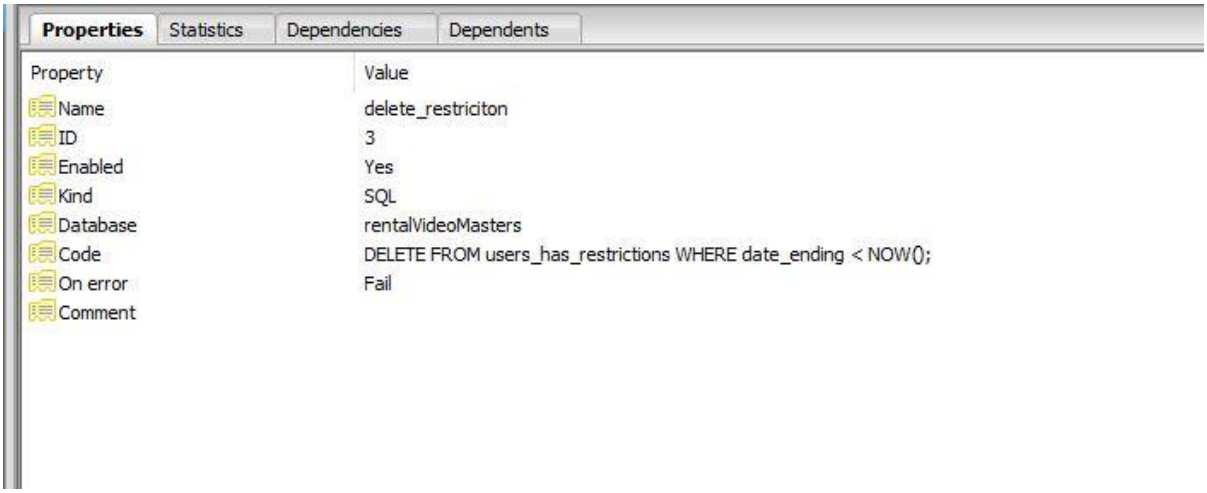
9.3 Problem isteklih zaliha

Još jedan od problema koji je riješen zadacima tj. pgAgentom su problemi isteklih zabrana. Korisnik može dobiti zabranu komentiranja ili zabranu posudbe/rezervacije koja vrijedi određeno vrijeme. Isto kao i kod problema zakasnina nelogično je da se provjeravaju i zatim brišu samo dok se korisnik prijavi jer opet ne bi postojao uvid u realnu situaciju. Stoga je implementiran zadatak sa slijedećim postavkama za korake 9.3.1. te vremenom izvršavanja 9.3.2.



Slika 9.3.1. Vremenske postavke za istekle zalihe

Izvor: autor



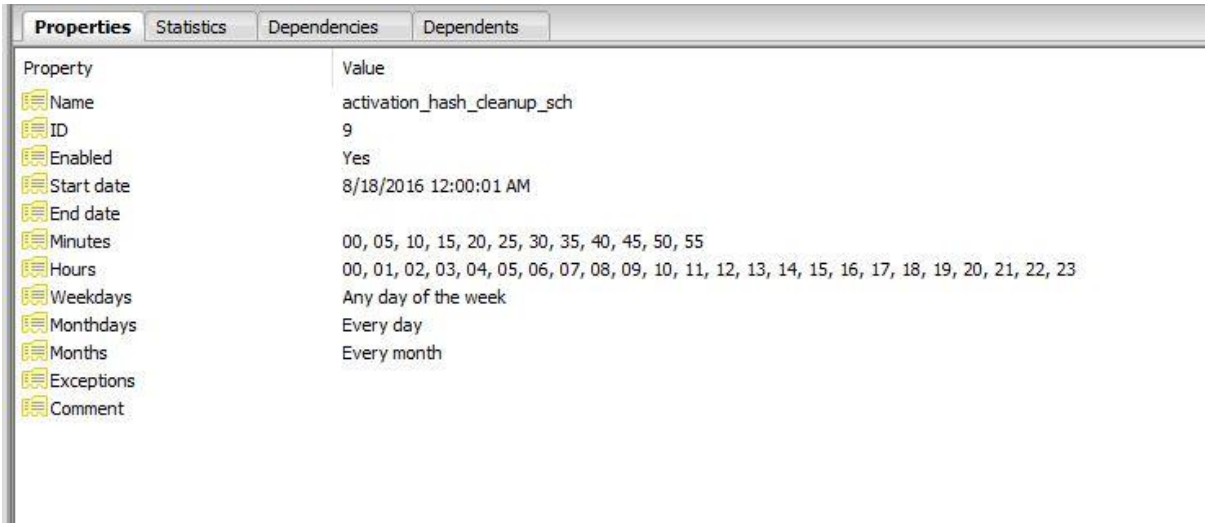
Slika 9.3.2. Postavke za istekle zalihe

Izvor: autor

Ovdje je vrijeme izvršavanja nešto drugačije jer se istek zabrana ne računa na dnevnoj bazi nego u minutama kao najnižoj mjernoj jedinici u alatu. Stoga se odabire da se definicija zadatka izvršava svakog mjeseca, svakog dana u mjesecu i tjednu i to svakog sata i svake minute (s obzirom na to da je minuta najmanja vremenska jedinica u pgAgentu. Provjerava se da li je definirano vrijeme isteka manje od trenutnog vremena tj. ako je zabrana istekla. U tom slučaju zabrana se briše i korisnik ju više nema.

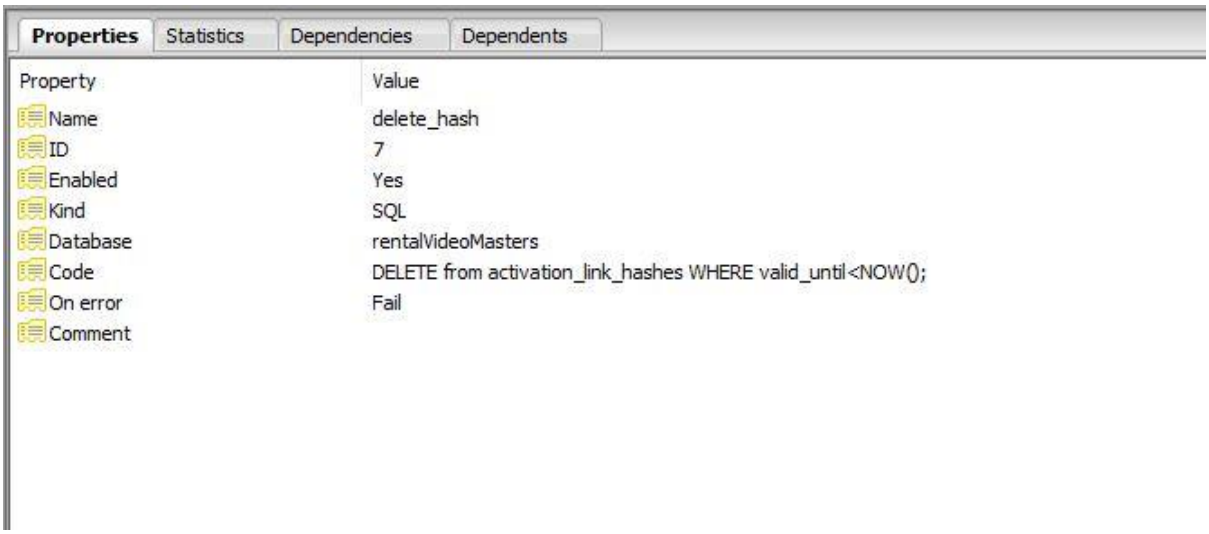
9.4 Problem aktivacijskih šifri

Prilikom registracije ili na vlastiti zahtjev svakom korisniku se dodjeli aktivacijska šifra kako bi mogao koristiti aplikaciju. Svaka ta šifra ima svoj vijek trajanja (u ovom slučaju 3 dana). Pošto se ta šifra nakon isteka valjanosti smatra nevažećom rješenje je bilo ili svaki puta prilikom aktivacije provjeravati da li je šifra istekla pa nju ili brisati/ignorirati ili osmisliti neki drugi način rješavanja ovog problema. Pošto su navedene opcije izvedive, ali dosta ne praktične, ovo je bilo idealno riješiti ponavljajućim zadacima. Oni bi u točno određeno vrijeme provjeravali da li je trenutno vrijeme „novije“ od vremena isteka šifre te ako je su šifru obrisali. Samim time se ne pohranjuju nepotrebni podaci u bazi jer su šifre samo važne za aktivaciju računa. Stoga se u tu svrhu definira vremenski raspored izvršavanja kao što je prikazano na slici 9.4.1. zajedno sa zadanim koracima na slici 9.4.2.



Slika 9.4.1. Vremenski raspored izvršavanja – aktivacijske šifre

Izvor: autor



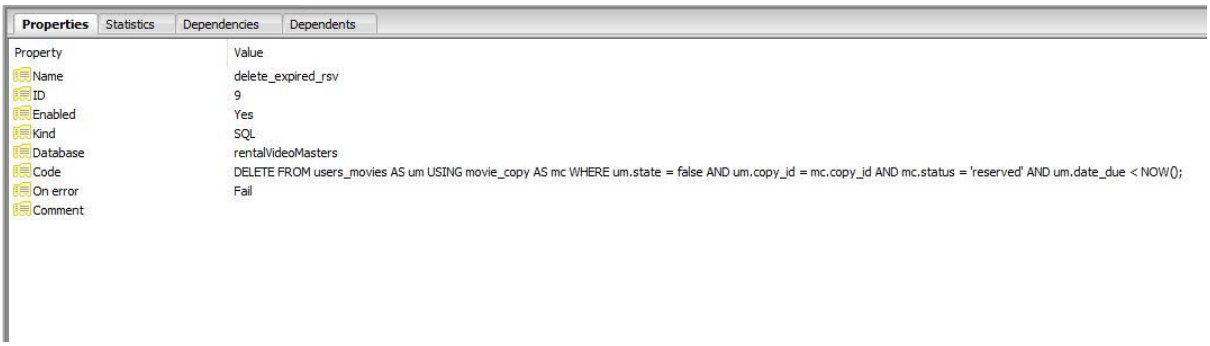
Slika 9.4.2. Definiranje koraka – aktivacijske šifre

Izvor: autor

9.5 Problem isteklih rezervacija

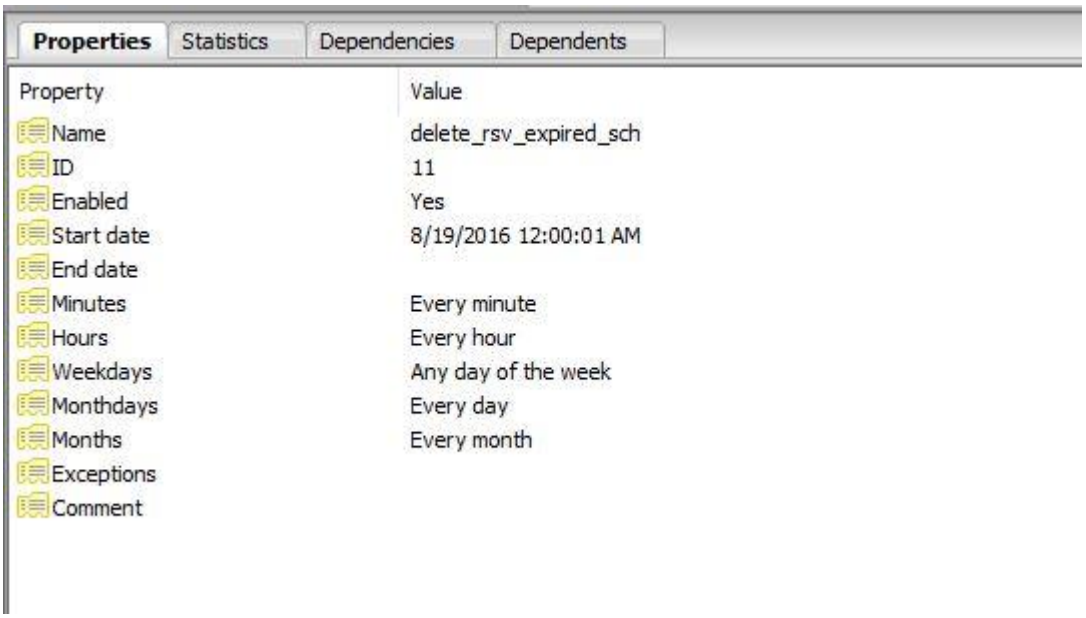
U sklopu aplikacije korisnik može rezervirati određeni film (njegovu kopiju) na određeno vrijeme, koje je postavljeno na 3 dana te taj film/kopiju posuditi. Pošto rezerviranu kopiju za vrijeme rezervacije nitko ne može posuditi kako korisnik ne bi ostao bez željenog filma problem se javlja u tome što ako korisnik nikada ne posudi rezervirani film/kopiju. Kopija bi tako vječno ostala zauzeta i nedostupna za daljnje posuđivanje. Problem je riješen uz kombinaciju okidača i ponavljajućih zadataka. Ponavljajući zadaci su korišteni kako bi obrisali rezervaciju ukoliko je

trenutno vrijeme novije od krajnjeg vremena rezervacije odnosno roka valjanosti rezervacije i ako se stvarno radi o rezervaciji. Nakon brisanja aktivirao bi se okidač koji bi postavio tu kopiju filma na slobodnu i tako ju učinio dostupnom ostalim korisnicima. Koraci su definirani kao na slici 9.5.1. i vremenski raspored izvršavanja prikazan na slici 9.5.2.



Slika 9.5.1. Definirani koraci za istekle rezervacije

Izvor: autor



Slika 9.5.2. Definirano vrijeme za istekle rezervacije

Izvor: autor

10. PHPMailer

Kako aplikacija omogućuje slanje elektroničke pošte s aktivacijskim šiframa na mail korisnika potrebno je bilo implementirati tu funkcionalnost. Jedna od opcija za implementaciju slanja elektroničke pošte je metoda "mail()" unutar PHP programskog jezika koja omogućuje slanje elektroničke pošte. Ona ovdje neće biti korištena jer posjeduje neka ograničenja kao npr. to što zahtijeva lokalni mail poslužitelj za slanje elektroničke pošte. Zbog tih razloga koristila se biblioteka pod nazivom "PHPMailer". Ova biblioteka je dostupna na GitHub-u te distribuirana pod LGPL 2.1 licencom pa je samim time besplatna za korištenje i modificiranje.

Prema (PHMailer, 2016.) "PHPMailer" karakteristike su:

- Vjerojatno najpopularniji kod za slanje elektroničke pošte za PHP
- Koristi se od strane raznih projekata otvorenog koda: WordPress, Drupal, 1CRM, SugarCRM, Yii, Joomla! te mnogih drugih
- Podrška za SMTP protokol – slanje elektroničke pošte bez lokalnog mail poslužitelja
- Pogreške na 47 različitih jezika
- Kompatibilnost s PHP 5.0 I kasnijim
- …

Ova biblioteka podržava SMTP protokol te zbog toga nije potrebno posjedovati predinstalirani mail poslužitelj. Time je omogućeno da se koristi Gmail račun kao račun s kojeg će se slati elektronička pošta. Za ovu potrebu kreiran je Gmail račun pod adresom "noguk114@gmail.com". On će se koristiti za slanje elektroničke pošte korisnicima. Prije slanja potrebno je postaviti odgovarajuće postavke unutar samog koda tj. postaviti odgovarajući SMTP host te SMTP port. Sve navedene postavke Google nudi na svojim stranicama te su one sljedeće:

- Host: smtp.gmail.com
- Port: 587

Sam postupak postavljanja biblioteke je jednostavan. Sa Github repozitorija se preuzme kompresirana datoteka te se ona otpakira u folder web stranice na serveru nakon stoga nju je samo potrebno uključiti u PHP skriptu u kojoj će se koristiti. Nakon postavljanja host-a i port-a, kompletirani kod koji se koristi za slanje elektroničke pošte korisnicima je sljedeći (lozinka je iz jasnih razloga sakrivena):

```
require_once './phpmailer/PHPMailerAutoload.php';

class SendMail {

    private $hash;
    private $username;
    private $hash_valid_until;
    private $email;
    private $fname;
    private $lname;

    public function __construct($hash, $username, $hash_valid_until, $email,
$fname, $lname) {

        $this->hash = $hash;
        $this->username = $username;
        $this->hash_valid_until = $hash_valid_until;
        $this->email = $email;
        $this->fname = $fname;
        $this->lname = $lname;
    }

    public function sendMail() {
        $message = "Dragi Korisniče " . $this->username . ", na temelju Vaše
registracije poslan vam je aktivacijski link: " . $this->hash . " koji
vrijedi do " . $this->hash_valid_until . ""
                . ". Molimo unesite aktivacijski link prilikom logina kako bi
aktivirali Vaš Račun!";
        $name = $this->fname . " " . $this->lname;

        date_default_timezone_set('Etc/UTC');

        $mail = new PHPMailer;
//pošto naš lokalni server nema potpisani (valjani) certifikat od ovlaštene
organizacije morali smo koristiti sljedeći kod kako bi ipak mogli poslati
mail
        $mail->SMTPOptions = array(
            'ssl' => array(
                'verify_peer' => false,
                'verify_peer_name' => false,
                'allow_self_signed' => true
            )
        );
        //protokol je smtp
        $mail->isSMTP();


        //$mail->SMTPDebug = 2;

        $mail->Debugoutput = 'html';

        $mail->Host = 'smtp.gmail.com';
        //broj porta
        $mail->Port = 587;
        //način enkripcije
        $mail->SMTPSecure = 'tls';
        //dio koji govori da je potrebna autentifikacija prije slanja pošte
        $mail->SMTPAuth = true;

        $mail->Username = "noguk114@gmail.com";

        $mail->Password = "**********";
        //od
```

```
        $mail->setFrom('noguk114@gmail.com', 'rentalVideo Videoteka');
         //za
        $mail->addAddress($this->email, $name);
        //predmet
        $mail->Subject = 'Aktivacijski Hash';


        //tijelo poruke
        $mail->Body = $message;


        if (!$mail->send()) {
            echo "Poruka nije poslana [Greška]: " . $mail->ErrorInfo;
        } else {
            echo "Poruka Poslana!";
        }
    }

}
```

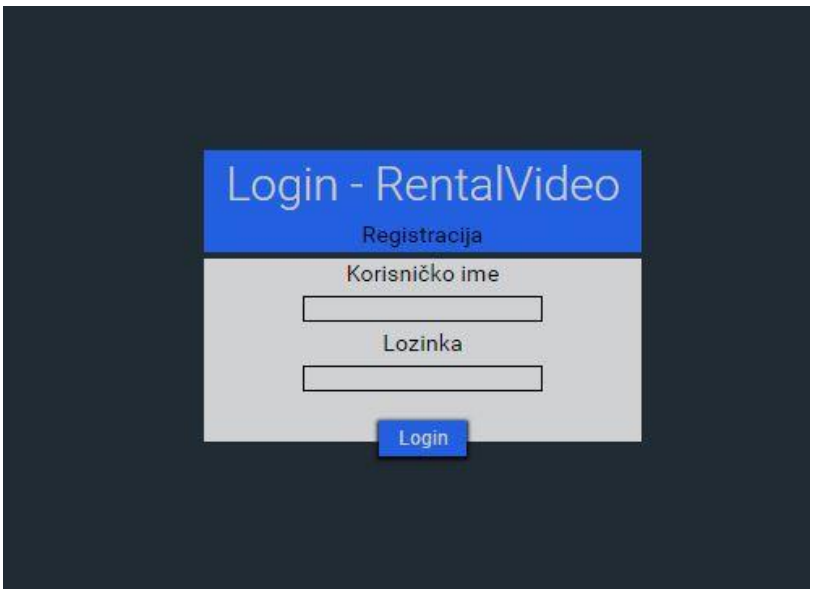
Konstruktor prima vrijednosti koje se koriste u slanju maila te se te vrijednosti prosljeđuju odgovarajućim funkcijama. Ukoliko je elektronička pošta poslana, odnosno nije poslana ispisuje se odgovarajuća poruka.

11. PRIMJERI KORIŠTENJA

U ovom poglavlju bit će prikazan konačan proizvod odnosno funkcionalna aplikacija te prikazano kako aplikacija izgleda uz njene funkcionalnosti, s obzirom na razine korisnika koje postoje.

11.1 Login

Prilikom posjeta web stranice “RentalVideo” videoteke prvo što svi korisnici vide, ukoliko nisu prijavljeni, je prozor za prijavu prikazan na slici 11.1.1. On je standardnog oblika s poljima za unos korisničkog imena te lozinke. Korisnik ima i opciju registracije ukoliko je prvi puta na stranici i želi koristiti naše usluge. Ukoliko se radi samo o loginu korisnik unese potrebne podatke i pristiskom na gumb “Login” sustav vrši provjeru korisnika (da li su ispravni podatci te da li je aktiviran korisnički račun aktiviran).



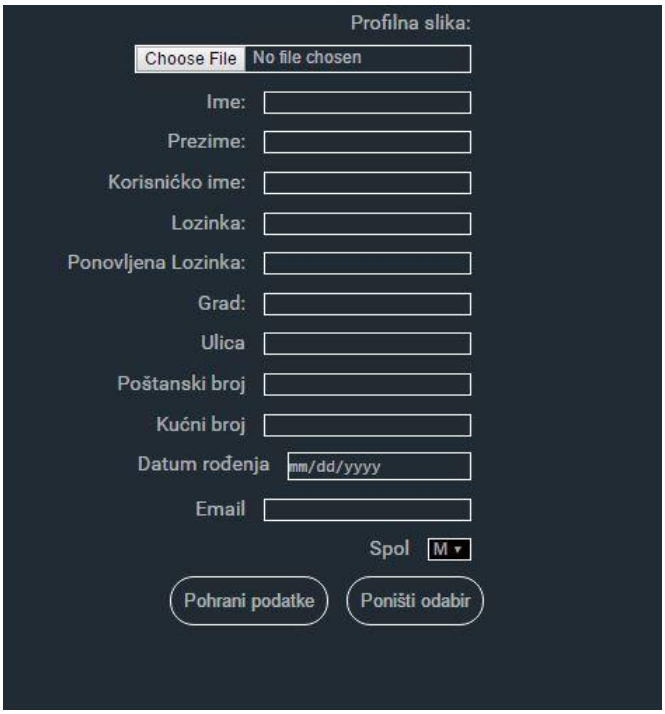
Slika 11.1.1. Login forma aplikacije

Izvor: autor

11.2 Registracija

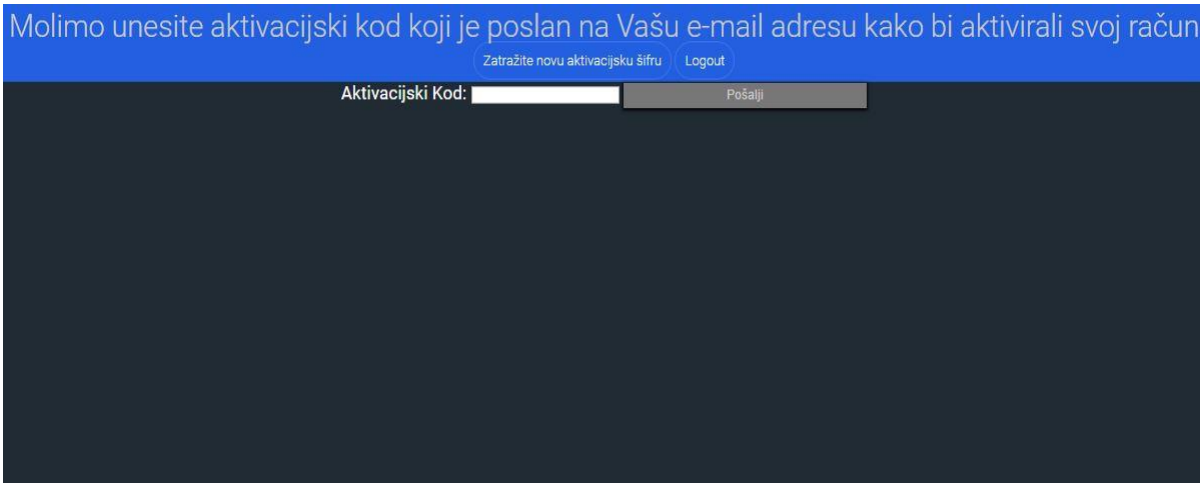
Ukoliko se korisnik odluči registrirati prikazat će mu se prozor za registraciju na slici 11.2.1. Ovdje korisnik unosi svoje podatke i odabire profilnu sliku te ukoliko su svi podaci ispravni aplikacija ga pohranjuje u našu bazu podataka. Pošto se korisnik tek registrirao korisnički račun mu nije aktivan te ga mora aktivirati aktivacijskom šifrom koju je dobio kao

poruku u svojoj elektroničkoj pošti. Svakog korisnika koji nije aktivirao svoj račun aplikacija preusmjerava na formu u kojoj unosi šifru te tako aktivira svoj račun ukoliko je ona ispravna (Slika 11.2.2). Korisnik unutar ove forme može zatražiti i novu šifru ukoliko mu je ista istekla ili ju je izgubio.



Slika 11.2.1. Forma za registraciju

Izvor: autor

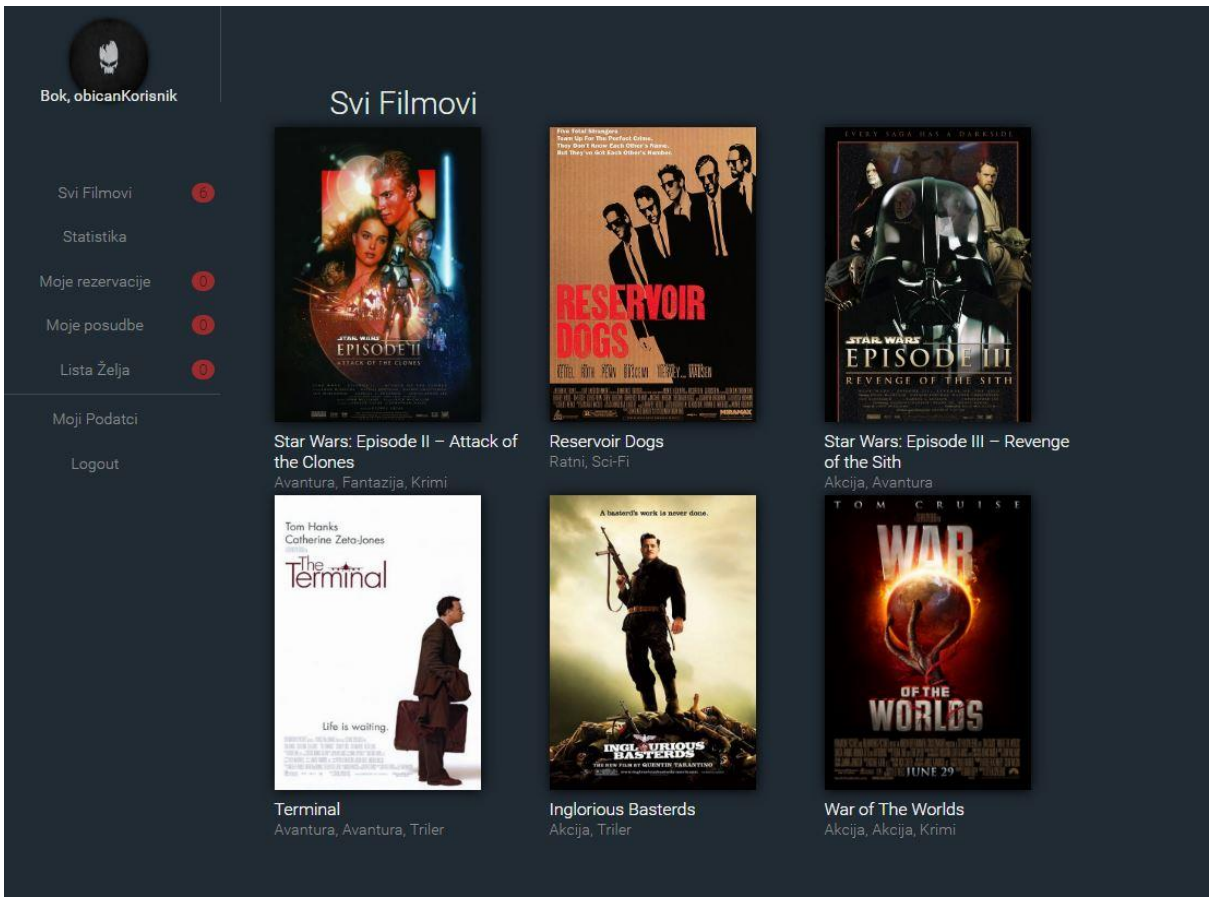


Slika 11.2.2. Unos aktivacijskog koda

Izvor: autor

11.3 Početna stranica

Početna stranica prikazuje sve filmove u videoteci te je ona vidljiva na slici 11.3.1. Korisnika se klikom na određeni film preusmjerava na stranicu s detaljima filma te dostupnim akcijama ovisno od razine ovlasti. Početna stranica je za sve korisnike ista jedino u čemu se razlikuje su dostupne opcije prikazane unutar navigacijskog izbornika. Sve razine korisnika naslijeđuju prava običnog korisnika stoga su im dostupne sve opcije kao i običnom korisniku.

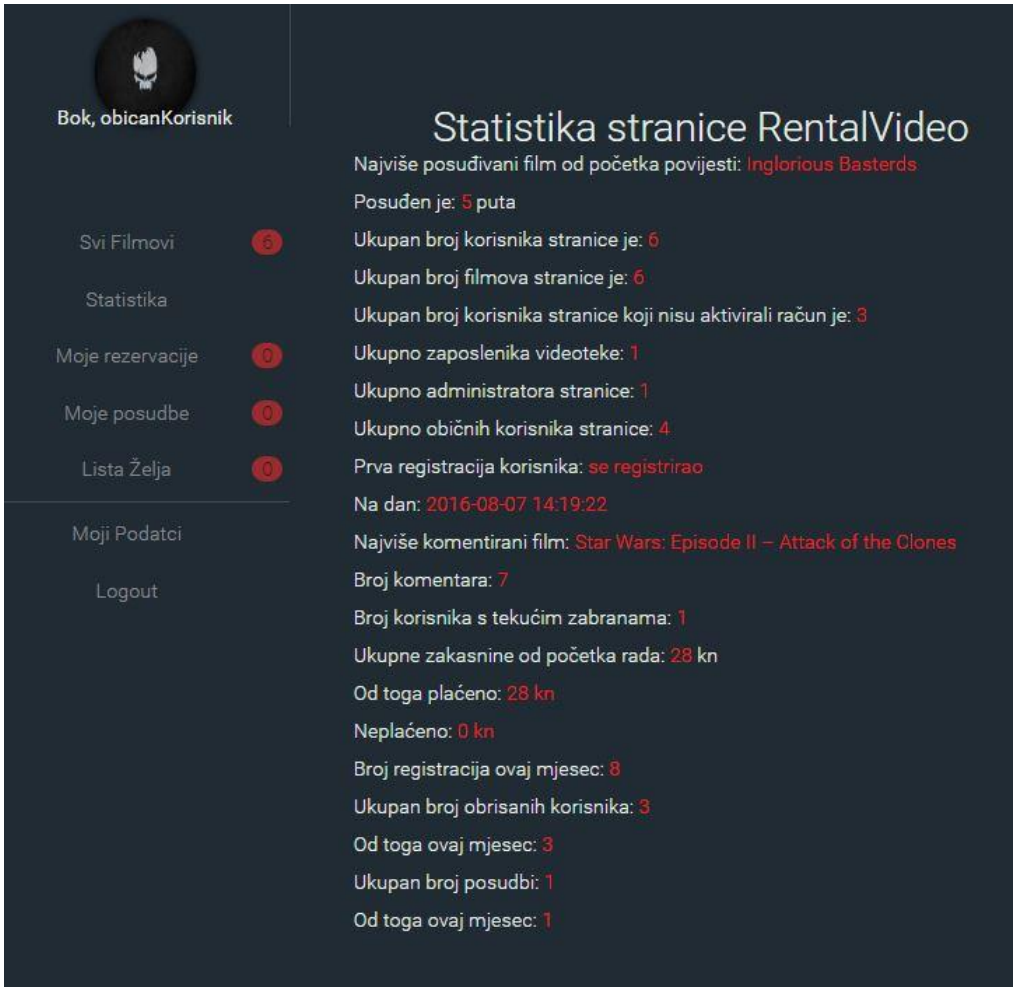


Slika 11.3.1. Početna stranica aplikacije

Izvor: autor

11.4 Statistika stranice

Svaka razina korisnika ima mogućnost pregleda statistike stranice (vidljivo na slici 11.4.1). i to klikom na gumb “Statistika” iz navigacijskog izbornika. Ovdje korisnik može vidjeti razne korisne informacije poput (prve registracije, datuma prve registracije, ukupne zakasnine, plaćene i neplaćene zakasnine, broj ukupnih korisnika, broj korisnika koji nisu aktivirali račun I drugo). Korisnik ove podatke može samo čitati te se oni ni na koji način ne mogu mjenjati.

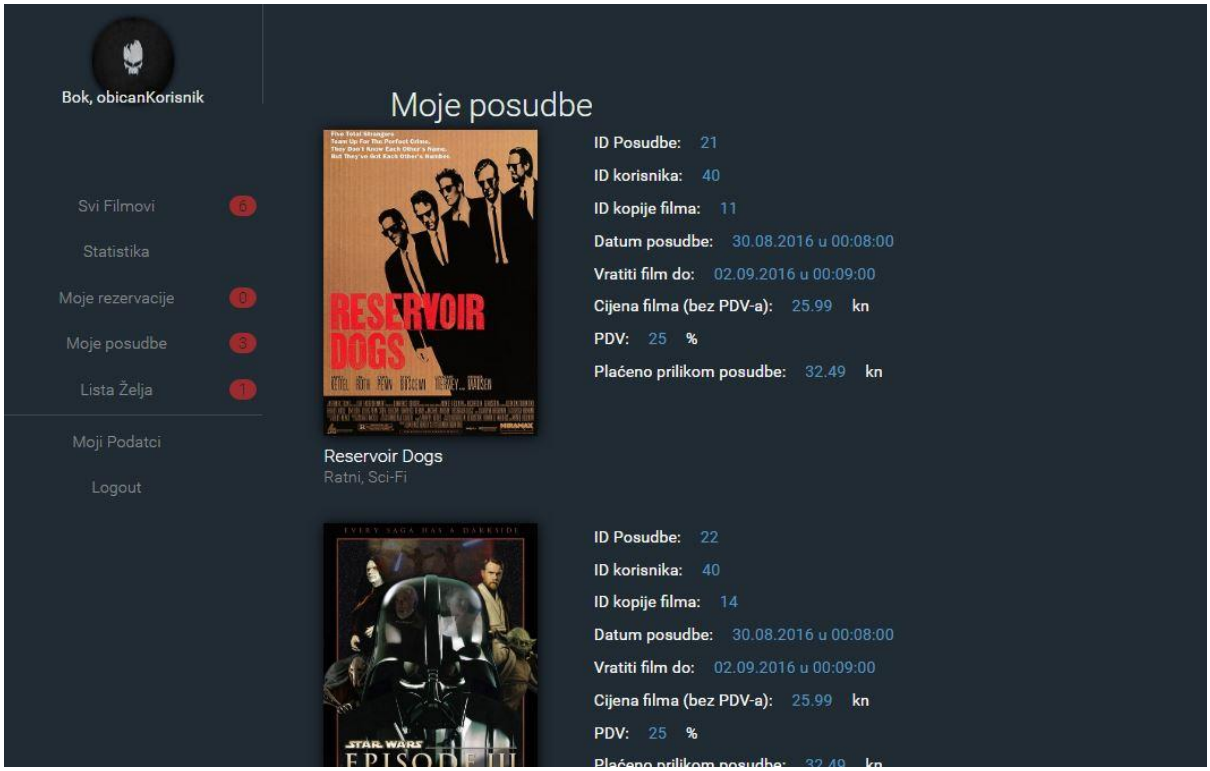


Slika 11.4.1. Statistika stranice

Izvor: autor

11.5 Moje posudbe

Stranici "Moje posudbe" pristupa se klikom na gumb "Moje posudbe", unutar navigacijskog izbornika. Ovdje korisnik vidi popis svih posudbi odnosno posuđenih filmova koje je podigao i koji su fizički kod njega (slila 11.5.1). S lijeve strane glavnog sadržaja nalazi se poster te ime i žanrovi filma. S desne strane korisnik vidi informacije o pojedinoj posudbi kao što su: id kopije, id filma, id korisnika, kada je film posuđen, do kada posudba vrijedi, cijenu filma, stopu PDV-a na posudbu te konačnu cijenu koju je korisnik platio prilikom posudbe. Korisnici na ovoj stranici nemaju nikakva prava izmjene podataka osim pregleda podataka.

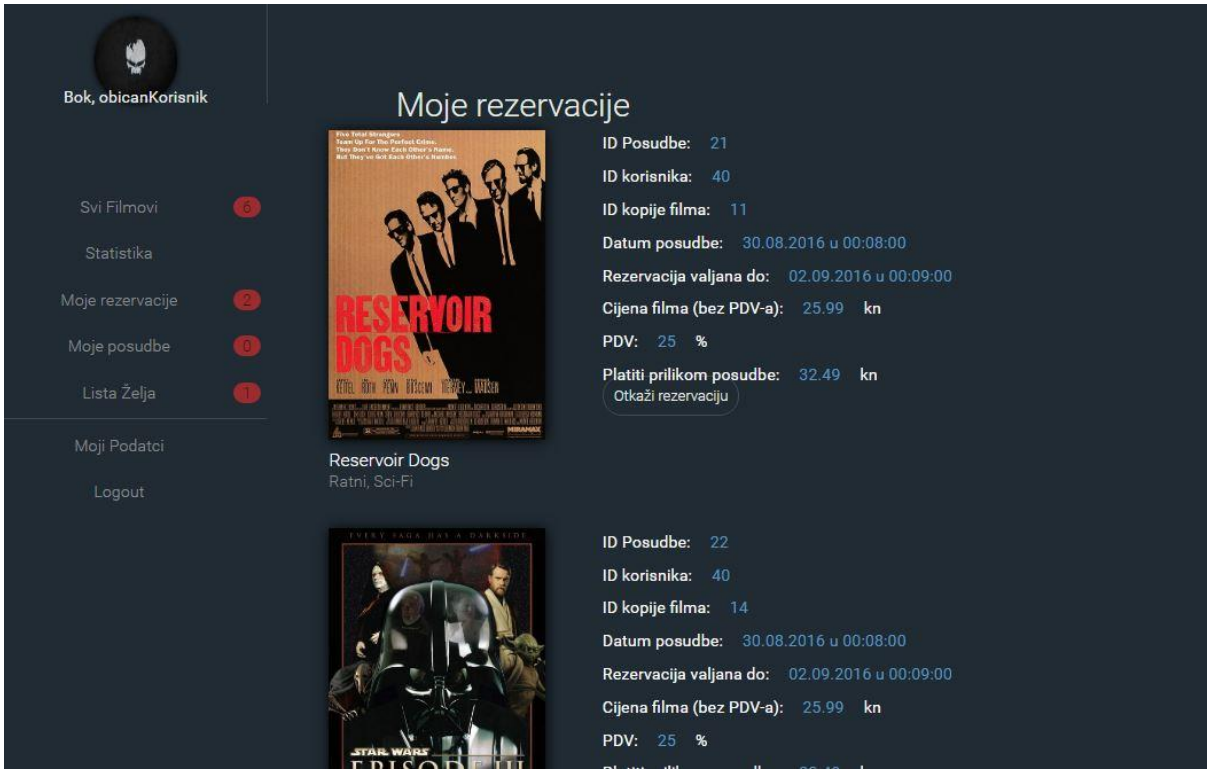


Slika 11.5.1. Prikaz korisnikovih posudbi filmova

Izvor: autor

11.6 Moje rezervacije

Stranici “Moje rezervacije” pristupa se klikom na gumb “Moje rezervacije”, unutar navigacijskog izbornika. Ovdje korisnik vidi popis svih rezervacija koje su trenutno važeće i koje pripadaju samo njemu. Prikaz je vidljiv na slici u nastavku. S lijeve strane glavnog sadržaja nalazi se poster te ime i žanrovi filma kao i na prethodnom prikazu posudbi filmova. S desne strane korisnik vidi informacije o pojedinoj rezervaciji kao što su: id kopije, id filma, id korisnika, kada je film rezerviran, do kada rezervacija vrijedi, cijenu filma, stopu PDV-a na posudbu te konačnu cijenu koju mora platiti kako bi posudio film. Korisnik može otkazati svoju rezervaciju u svakom trenutku (te tako odustati od posudbe) klikom na gumb “Otkaži rezervaciju”.

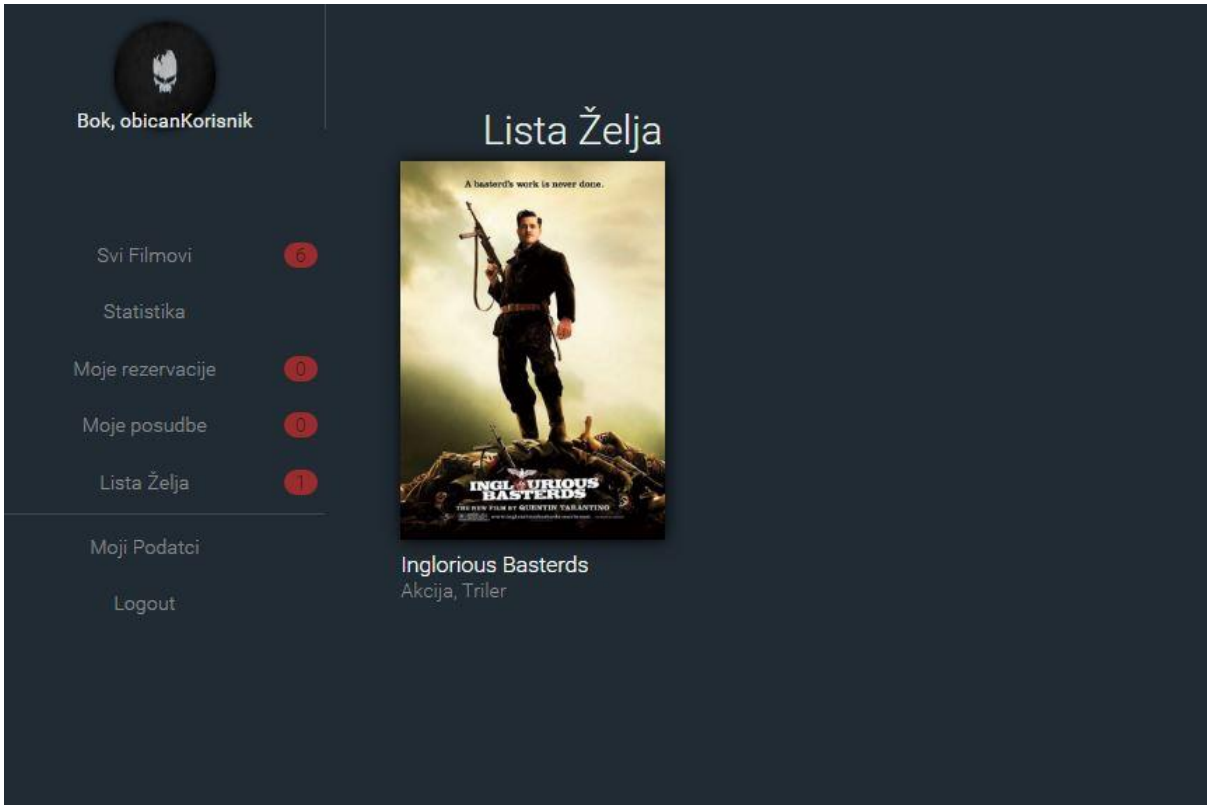


Slika 11.6.1. Prikaz korisnikovih rezervacija filmova

Izvor: autor

11.7 Lista želja

Lista želja prikazana na slici 11.7.1. je posebna vrsta stranice gdje se nalaze svi filmovi koje je korisnik označio da ih želi, ali iz nekog razloga ih trenutno nije u mogućnosti posuditi. Do nje se dolazi klikom na “Whishlist” (hrv. Lista želja) iz navigacijskog izbornika. Lista želja je tu kako bi se korisnik mogao podsjetiti koje filmove još nije pogledao, a hoće pogledati. Prikaz filmova je jednostavan s posterom te imenom i žanrovima. Klikom na pojedini film korisnika se odvodi na detalje samog filma gdje može ukloniti fillm sa liste.



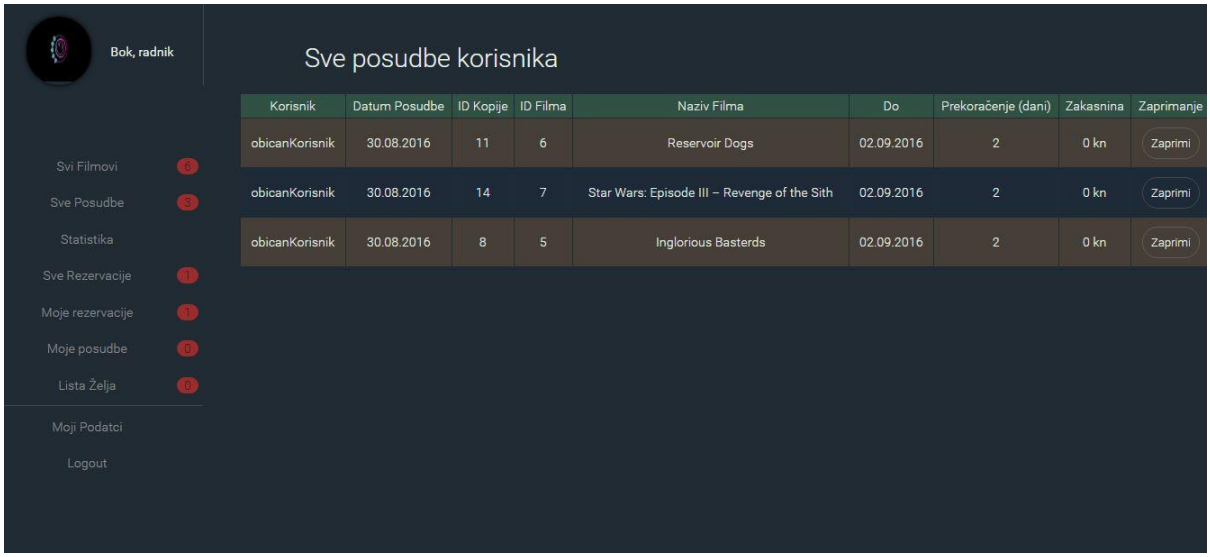
Slika 11.7.1. Prikaz liste želja

Izvor: autor

U nastavku će biti prikazane individualne mogućnosti aplikacije s obzirom na to da su prikazane aktivnosti od točaka 11.1. do 11.7. bile jednake za sve vrste korisnika. Nadalje će biti prikazane razlike između preostalih funkcionalnosti za sve vrste korisnika.

11.8 Sve posudbe

Do stranice “Sve posudbe korisnika” na slici 11.8.1 dolazi se klikom na gumb “Sve Posudbe” unutar navigacijskog izbornika. Ovdje radnik videoteke ima uvid u sve filmove koji su trenutno posuđeni odnosno fizički su kod korisnika. Stranica nudi informacije o tome tko je posudio film, kada je film posuđen, do kada posudba vrijedi, ime filma, id filma i kopije, koliko dana korisnik kasni s povratkom filma te odgovarajuću zakasninu na temelju tih dana. Radniku videoteke je dostupna opcije “Zaprimi” što označava da je korisnik platio zakasninu, ukoliko ju ima te vratio odgovarajući film u videoteku. Nakon zaprimanja vraćena kopija postaje dostupna za daljnje posuđivanje.

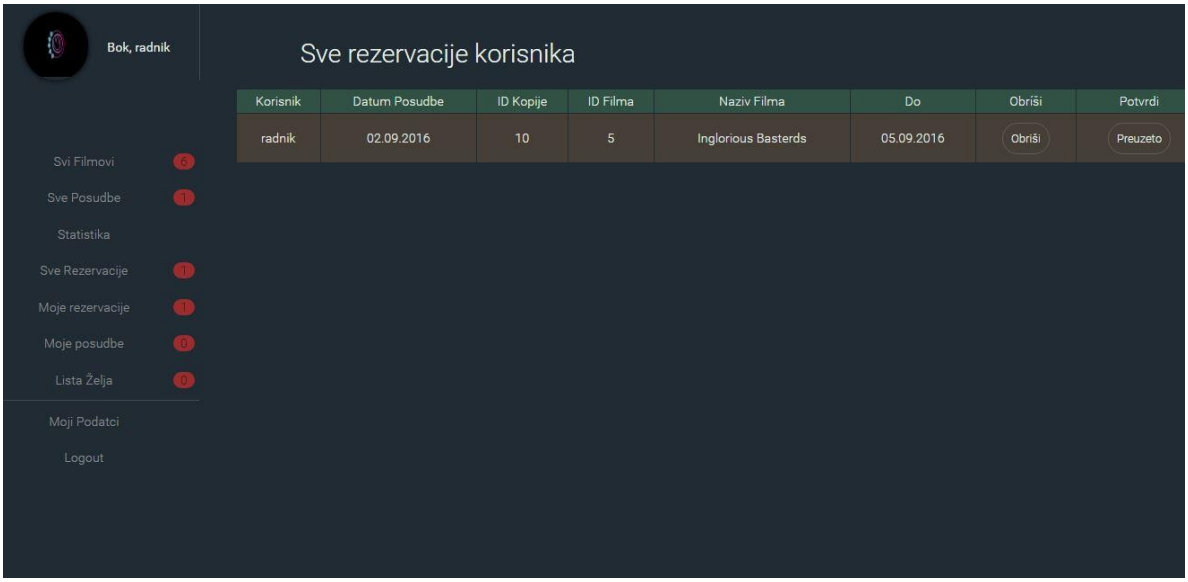


Slika 11.8.1. Sve posudbe korisnika

Izvor: autor

11.9 Sve rezervacije

Do stranice "Sve rezervacije korisnika" dolazi se klikom na gumb "Sve Rezervacije" unutar navigacijskog izbornika. Ovdje radnik videoteke ima uvid u sve filmove koji su trenutno rezervirani. Stranica nudi informacije o tome tko je rezervirao film, kada je film rezerviran, do kada rezervacija vrijedi, ime filma, id filma te id kopije. Radniku videoteke su dostupne opcije "Obriši" čime rezervaciju briše i oslobađa kopiju za daljnje posuđivanje te opcija "Preuzeto" koja u drugom smislu znači da je korisnik fizički posudio kopiju nakon čega ona iz stanja rezerviranog prelazi u stanje posuđene.

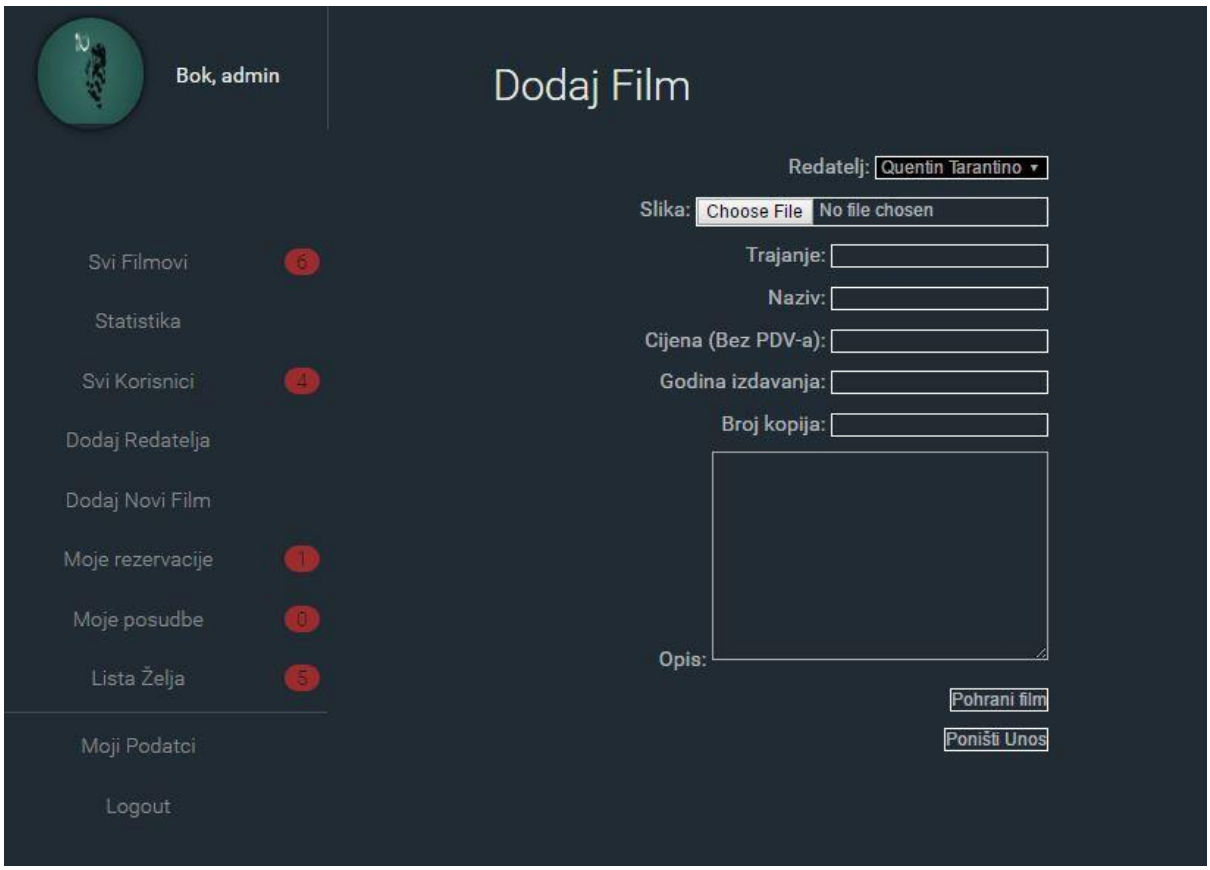


Slika 11.9.1. Prikaz svih rezervacija korisnika

Izvor: autor

11.10 Dodavanje filmova

Stranici “Dodaj Film” se pristupa klikom na gumb “Dodaj Novi Film” unutar navigacijskog izbornika. Dolaskom na ovu stranicu administrator ima mogućnost dodati novi film. Ovdje administrator unosi informacije o filmu kao što su: trajanje, naziv filma, godina izdavanja, broj kopija koje će videoteka posjedovati za posudbe… Uz sve navedeno administrator odabire redatelja tog filma (ako ne postoji doda ga putem gumba “Dodaj Redatelja”) te odabire poster filma koji se fizički pohranjuje u bazu podataka. Nakon unosa svih podataka klikom na gumb “Pohrani film” pohranjuje se film u bazu podataka.



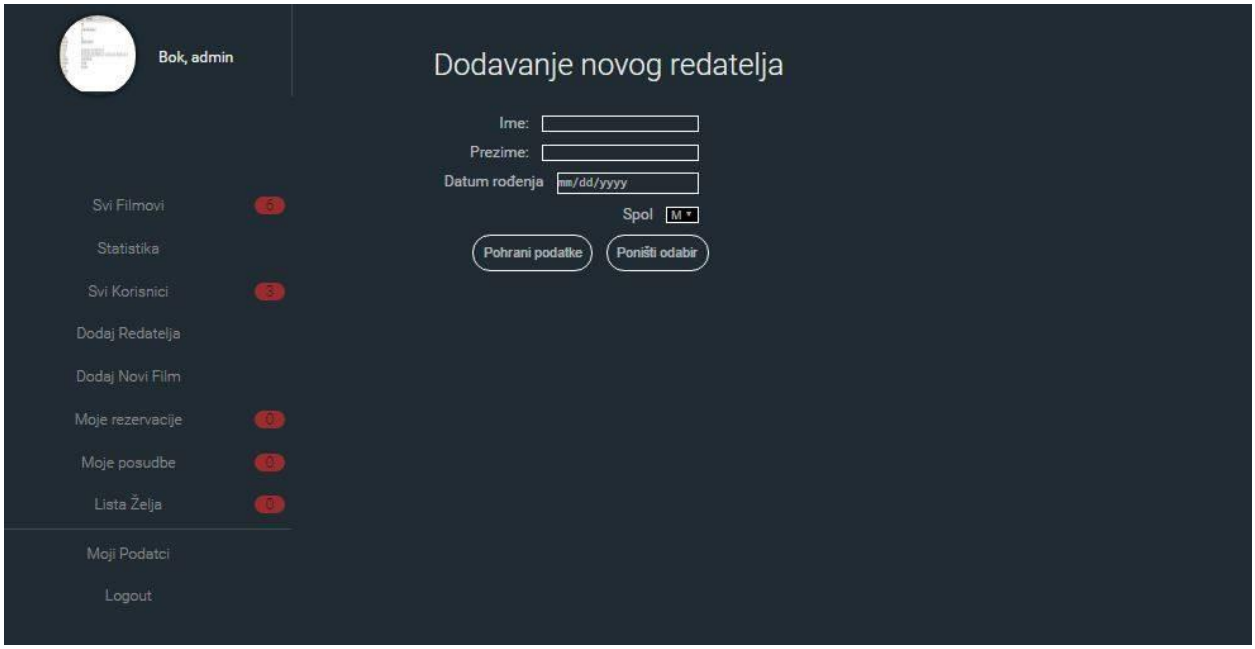
Slika 11.10.1. Dodavanje filmova

Izvor: autor

11.11 Dodavanje novog redatelja

Stranici “Dodavanje novog redatelja” pristupa se klikom na gumb “Dodaj Redatelja” unutar navigacijskog izbornika. Dolaskom na ovu stranicu administrator ima mogućnost dodati novog redatelja te kasnije prilikom dodavanja novog filma, pridružiti novokreiranog redatelja

tom filmu. Administrator unosi informacije o redatelju (ime, prezime, spol, datum rođenja) te klikom na gumb “Pohrani podatke” pohranjuje novog redatelja u bazu podataka.



Slika 11.11.1. Dodavanje novog redatelja

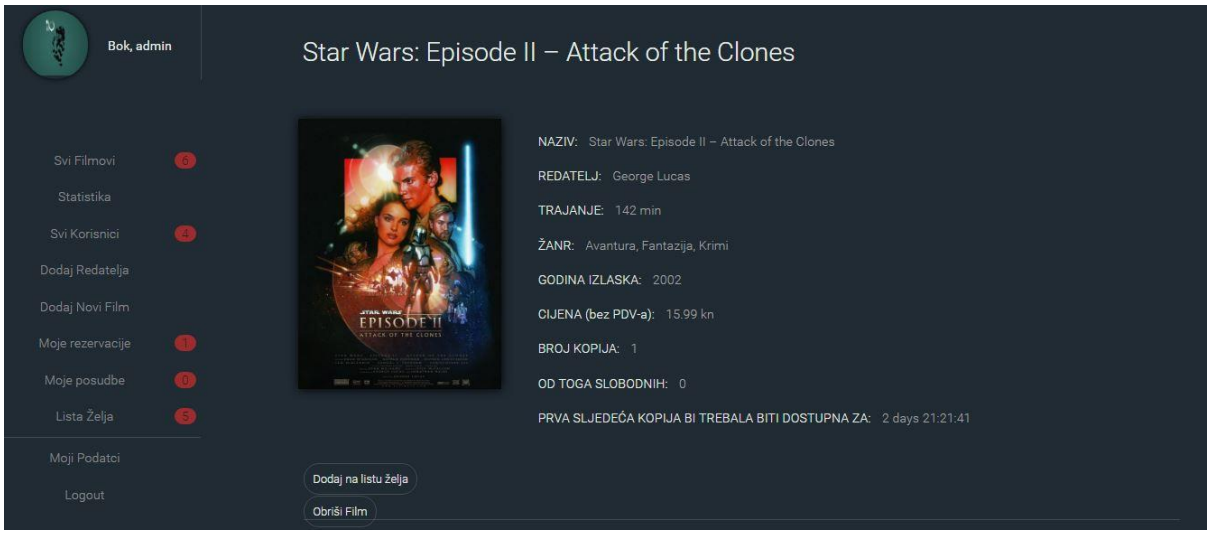
Izvor: autor

11.12 Pregled detalja filma

Pregled detalja filma se sastoji od više dijelova (samog filma i dijela za komentare). Svaki od tih dijelova će se posebno objasniti jer su prava dosta različita. Ovoj stranici imaju pristup svi korisnici, no određene razine korisnika imaju određena prava. Administrator ima prava brisanja filma (vidljivo na slici 11.12.2). S druge strane ostali korisnici to pravo nemaju. Nadalje gumbi, tj. opcije “Dodaj na listu želja” odnosno “Ukloni s liste želja” te “Rezerviraj” su dostupni svim korisnicima. Tim opcijama korisnici dodaju/uklanjaju određeni film s liste želja te rezerviraju film ukoliko je određena kopija slobodna. Ukoliko film nema niti jednu kopiju slobodnu, korisnicima se prikazuje informacija o vremenu kada bi prva slobodna kopija trebala biti dostupna.

Slika 11.12.1. Prikaz detalja filma svim korisnicima

Izvor: autor



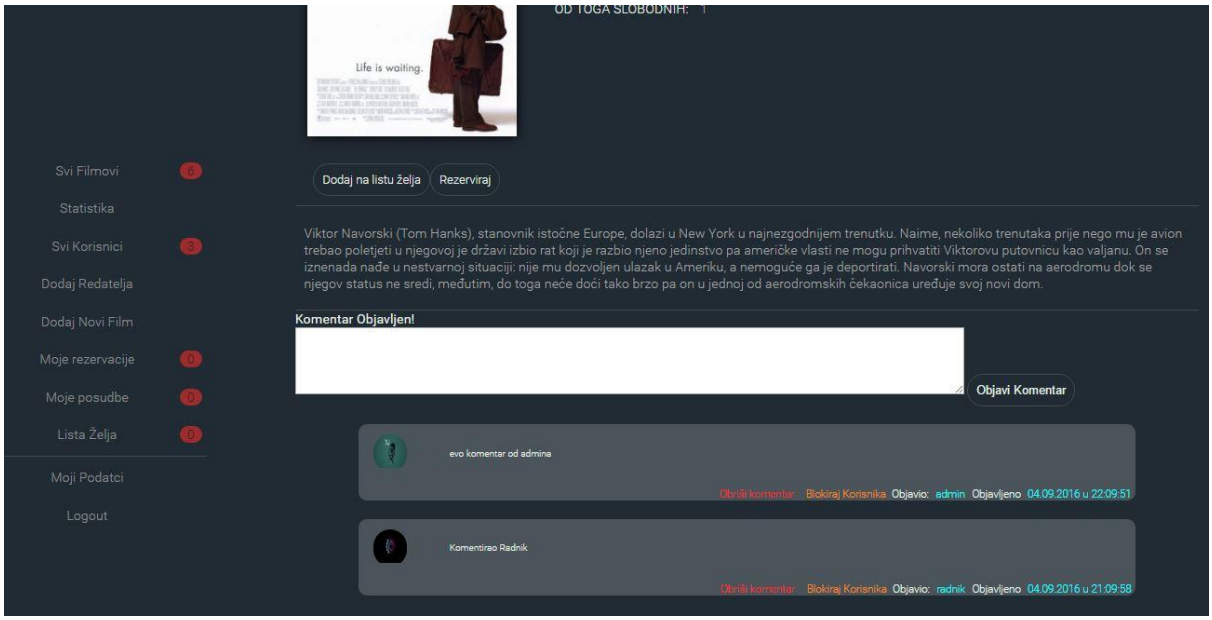
Slika 11.12.2. Prikaz detalja filma – sa administatorskim mogućnostima

Izvor: autor

11.13 Mogućnost komentiranja

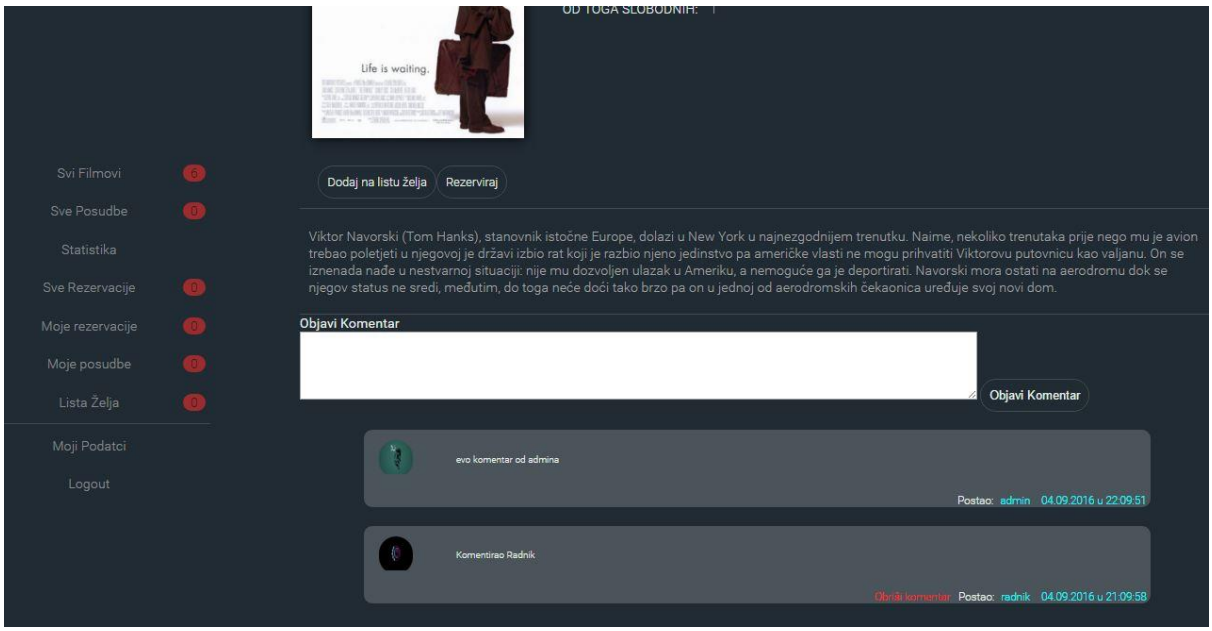
Komentiranje je opcija u sklopu pregleda pojedinog filma. Komentirati film mogu svi korisnici (koji nemaju zabrane komentiranja), no administrator ima neke dodatne opcije. Administrator može, kao i svi ostali korisnici, objaviti komentar na pojedini film (ukoliko nema zabranu komentiranja) ali i može dati korisniku zabranu, tj. zabraniti mu komentiranje na određeni rok definiran unutar naše aplikacije. Uz zabranu komentiranja administrator može obrisati svoj komentar i svaki drugi komentar korisnika (slika neki broj) dok ostale razine mogu

obrisati samo svoj komentar (prikaz na slici 11.13.1). Za sve se korisnike prilikom objave komentara objavljuje profilna slika korisnika koji je komentirao, datum kada je komentar objavljen te korisničko ime korisnika koji je komentar objavio te u slučaju administrator opcije za brisanje komentara i davanje zabrana komentiranja.



Slika 11.13.1. Mogućnosti komentiranja za administrator

Izvor: autor

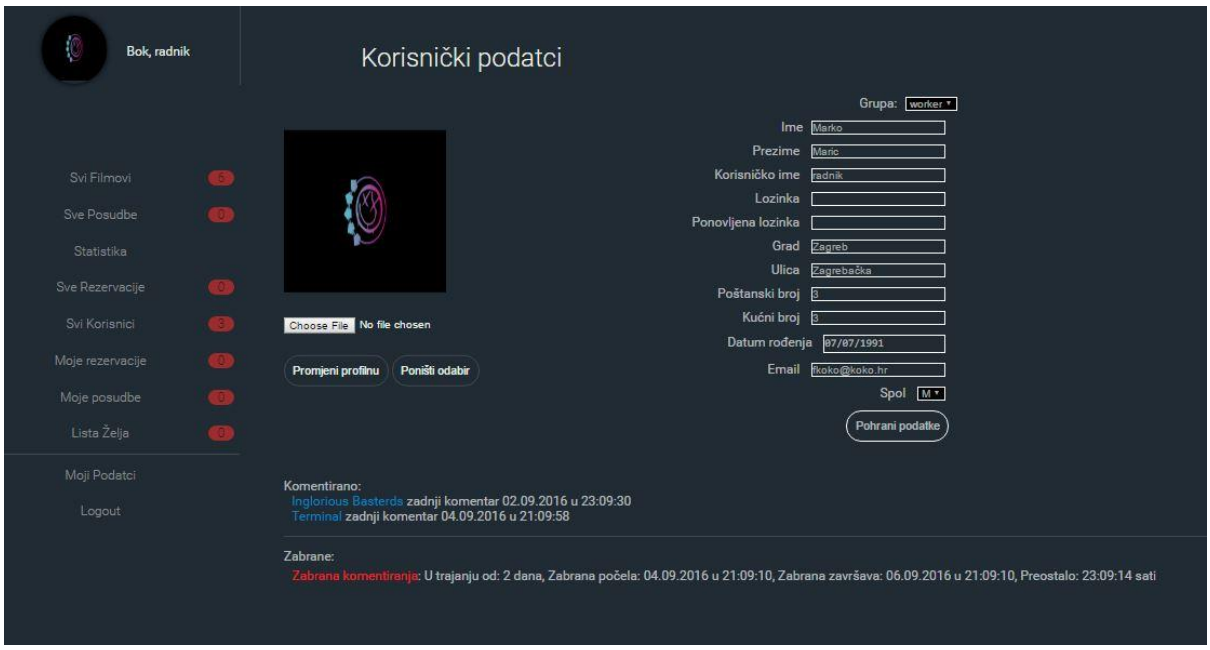


Slika 11.13.2. Mogućnosti komentiranja za ostale korisnike

Izvor: autor

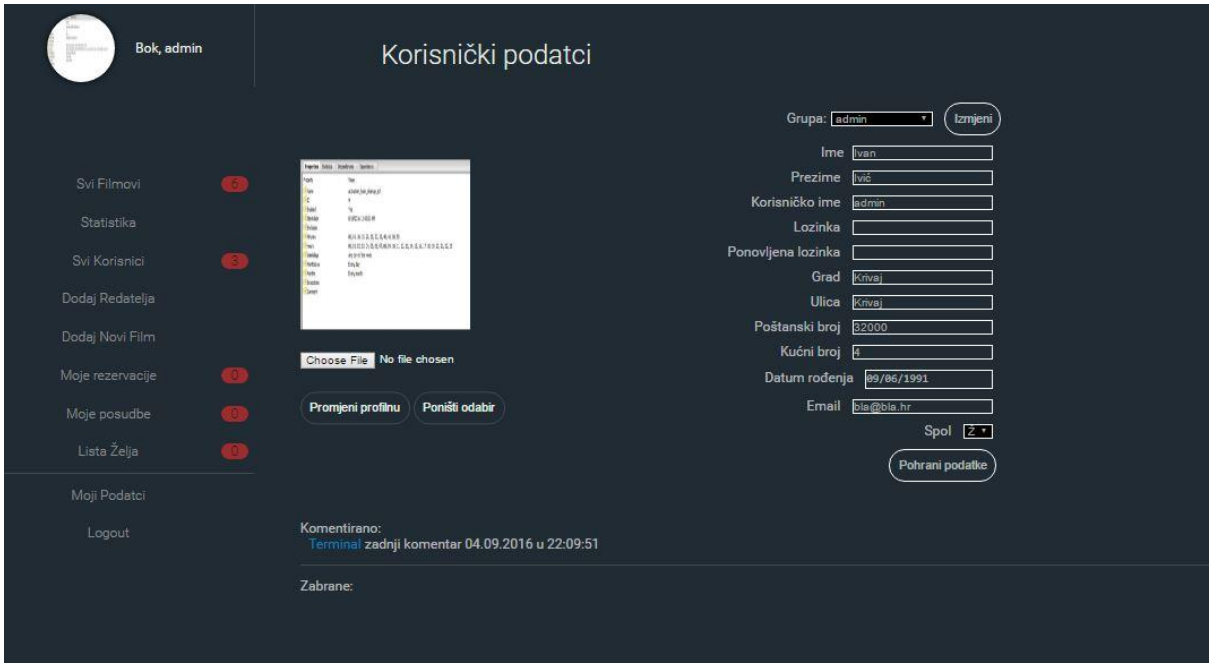
11.14 Korisnički podaci

Svi korisnici imaju mogućnosti pregleda i izmjene svojih podataka (prikaz na slici 11.14.1). Oni vide svoju profilnu sliku, informacije o sebi, što i kada su komentirali te koje i do kada zabrane imaju. Kao što je spomenuto svi korisnici imaju mogućnost izmjene svojih podataka, ali tuđih ne. Jedina razina koja može izmjenjivati tuđe podatke je administrator, ali on može samo promjeniti razinu korisnika. Kada otvori nečiji profil dostupan mu je gumb "Promjeni" te padajući izbornik iz kojeg odabire novu grupu korisnika. Mogućnosti su vidljive na slici 11.14.2.



Slika 11.14.1. Korisnički podaci – svi korisnici

Izvor: autor



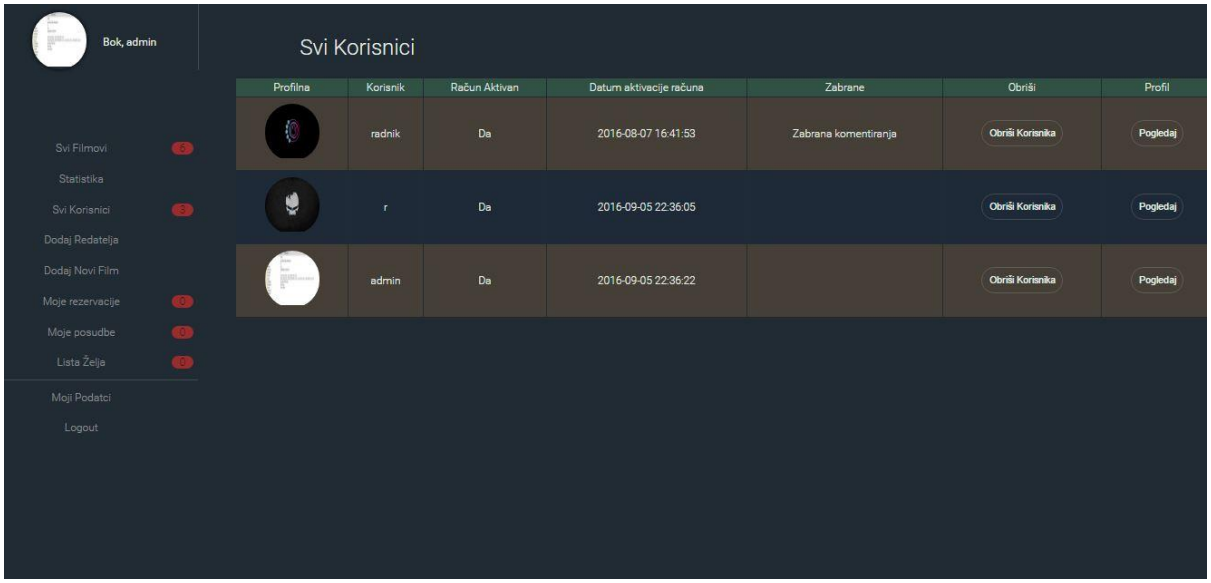
Slika 11.14.2. Korisnički podaci - administrator

Izvor: autor

11.15 Svi korisnici

Stranici "Svi Korisnici" pristupa se putem gumba "Svi Korisnici" unutar navigacijskog izbornika. Stranica prikazuje sve korisnike sustava. Ovu stranicu mogu vidjeti samo administratori i radnici svaki uz svoja ograničenja. Tako administrator vidi stranicu kao na slici 11.15.1. On ima opciju brisanja korisnika iz baze klikom na gumb "Briši" te mu može pogledati profil klikom na gumb "Pogledaj" gdje može izmjeniti prava. Radnik vidi stranicu kao na slici 11.15.2. gdje je jedina opcija koju vidi ta da može "posjetiti" profil korisnika, ali kao što je već navedeno, mu ne može izmjeniti nikakva prava niti podatke. Unutar ove stranice i admin i radnik vide sljedeće podatke o korisnicima:

- Profilnu sliku korisnika
- Korisničko ime
- Da li je korisnik račun aktivirao
- Kada je račun aktiviran
- Zabrane koje korisnik ima
- Akcije zavisno od pravima (brisanje korisnika i gledanje profila)



Slika 11.15.1. Svi korisnici- administrator

Izvor: autor

Slika 11.15.2. Svi korisnici- ostali

Izvor: autor

12. ZAKLJUČAK

Kroz ovaj rad objašnjeno je, teorijski i praktično, kako funkcioniraju i kako se koriste temporalne, objektno-relacijske, aktivne te poopćene baze podataka. Sveukupan rezultat ovog rada je baza podataka koja se može koristiti u realnom svijetu. Prilikom izrade ove baze podataka bilo je nekolicina problema poput isteka rezervacija, isteka aktivacijskih ključeva, zakasnine… Kako bi te probleme riješili korišten je pgAgent i njegove mogućnosti, naučeno je kako se koriste ponavljajući zadatci, kako definirati korake, vremenske rasporede izvršavanja I drugo. Pokazano je kako se objektno-relacijskim (nasljeđivanje) te poopćenim (složeni tipovi) može smanjiti količinu koda te ukoliko je potrebno promjene izvršiti samo na jednom mjestu umjesto na više njih.

OID, koncept iz poopćenih baza podataka, se pokazao jako korisnim. Kao što je i rečeno u radu ukoliko se koriste OID-ovi za pohranu binarnih datoteka prilikom migracije baze dovoljno je samo izvesti datoteke i one su opet tu. Jedini nedostatak toga je što baza, ukoliko se pohranjuju velike količine datoteka, postaje jako velika. Aktivne baze podataka su se pokazale jako korisnima u smislu što je u većini slučajeva unutar aplikacije za unos, brisanje i izmjenu podataka bilo dovoljno koristiti jednostavne upite, dok su se sve pozadinske provjere odvijale putem okidača. Samim time smanjenje je kod i složenost kod upita koje aplikacija vrši prema bazi podataka.

Temporalne baze podataka su se pokazale kao neizostavan dio baze podataka. Praćeni su vremenski aspekti poput datuma povratka filma, datuma posudbe, datum isteka zabrane, datuma registracije, datuma aktivacije i sl. Bez temporalnih baza podataka ne bi bilo moguće voditi evidenciju o navedenom te kao takva naša aplikacija ne bi mogla funkcionirati.

LITERATURA

Knjige:

1. Maleković M.; *Teorija baza podataka*, skripta iz kolegija Teorija baza podataka, 2008., Fakultet organizacije i informatike, Varaždin.

Članci:

2. Dayal U., Hanson Eric N., Widom J.; *Active Database Systems*, 1994., URL (http://ilpubs.stanford.edu:8090/54/1/1994-20.pdf) dostupno 5.08.2016.
3. Napredni modeli i baze podataka; *Objektno orijentirane i objektno-relacijske baze podataka*, www.fer.unizg.hr, 2014., URL (https://www.fer.unizg.hr/_download/repository/3._OOBP_i_ORBP.pdf) dostupno 3.08.2016.

Internet stranice:

1. pgAdmin.org; *pgAdmin 1.4. online documentation*, www.pgadmin.org, 2016. URL (https://www.pgadmin.org/docs/1.4/pgagent.html) dostupno 14.08.2016.
2. PHPMailer; 2016., www.github.com, URL (https://github.com/PHPMailer/PHPMailer) dostupno 7.08.2016.
3. PostgreSQL.org; *About*, 2016., URL (https://www.postgresql.org/about/) dostupno 3.08.2016.
4. PostgreSQL.org; *Date/Time Functions and operators*, 2016., URL (https://www.postgresql.org/docs/8.1/static/functions-datetime.html) dostupno 3.08.2016.
5. Schatten M.; Poopćene relacijske baze podataka, Autopoiesis.foi.hr, 2016., URL (http://autopoiesis.foi.hr/wiki.php?name=Baze%20Podataka%20-%20FOI&parent=NULL&page=poopbp) dostupno 3.08.2016.
6. TimeConsult.com; *What are Temporal Databases*, 2005., URL (http://www.timeconsult.com/TemporalData/TemporalDB.html) dostupno 3.08.2016.

POPIS SLIKA

POPIS TABLICA