

UNIVERZITET SINGIDUNUM

TEHNIČKI FAKULTET

**SOFTVER ZA UPRAVLJANJE REGISTROM
KINEMATOGRAFIJE**

- diplomski rad -

Mentor:

prof. dr *Đorđe Obradović*

Kandidat:

Aleksandar Budinčević

Beograd, 2021.

Sadržaj

1. Uvod	3
2. Specifikacija i model softvera	6
2.1 Dijagram slučajeve korišćenja	6
2.2 Dijagram klasa	8
3. Arhitektura softvera	9
3.1 Baza podataka	9
3.2 Backend	12
3.2.1 Sloj repozitorijuma	12
3.2.2 Sloj servisa	13
3.2.3 Sloj kontrolera	13
3.3 Frontend	14
3.4 Dijagram arhitekture i sažetak funkcionisanja softvera	15
4. Pregled korišćenih tehnologija	16
4.1 Baza podataka - SQL, MySQL i MySQL Workbench	16
4.2 Backend – Java, Java Spring, Hibernate ORM	16
4.3 Frontend– Angular 8, Typescript, HTML, CSS	17
5. Primeri korišćenja	19
5.1 Početna stranica (<i>Homepage</i>)	19
5.2 Pregled najbolje ocenjenih filmova	20
5.3 Stranica Filma	21
5.4 Stranica glumca i stranica režisera	22
5.5 Toolbar	23
5.6 Registracija novog korisnika	23
5.7 Prijava na sistem	24
5.8 Side navigation menu	24
5.9 Administrator CRUD operacije	25
5.10 Prikaz ocena i recenzija kritičara	26
5.11 Sistem za razmenu poruka	27
5.12 Sistem za preporuke	28
5.13 Forum	29
6. Zaključna razmatranja	30
Literatura	32

1. Uvod

Ovaj rad se odnosi na implementaciju softverskog sistema za upravljanje registrom kinematografije. Softver pruža pristup različitim uslugama u zavisnosti od tipa korisnika.

Korisnicima koji upravljaju softverom (*administrator*) omogućen je pregled i modifikovanje filmova i njihovih relevantnih podataka (glumci, režiseri, žanrovi), kao i nadgledanje svih korisnika softvera. Administratori su zaduženi za kontrolisanje svih podataka koje ostali korisnici mogu da unesu u bazu (recenzije, teme i poruke na forumu itd.). Kao pomoć administratorima u procesu upravljanja forumima, omogućen je sistem za klasifikaciju neprikladnih poruka, o kojem će kasnije biti više reči.

Recenzent (*critic*) je poseban tip korisnika koji ima određene privilegije. U skladu sa tim, recenzent može biti registrovan i dodat u sistem samo od strane administratora. Registrovanim recenzentima je pružena mogućnost pisanja recenzija filmova kao i davanje posebnih ocena (*critic score*). Prikaz i pregled svojih recenzija i ocena omogućuje recenzentu da modifikuje podatke koje je prethodno uneo.

Poslednji tip registrovanih korisnika su obični korisnici (*user*). Svaki korisnik softvera ima priliku da se registruje kao *user*, čime će dobiti priliku da da svoju ocenu (*user score*) filmovima, da lajkuje filmove i na osnovu njih od sistema za preporuke dobije listu preporučenih filmova, kao i da učestvuje u diskusijama na forumima. Registrovani korisnici takođe mogu da stupe u direktni kontakt sa administratorima preko sistema za razmenu poruka, u slučaju da imaju nekih poteškoća ili problema koji samo administratori mogu da reše.

Softver mogu koristiti i neregistrovani korisnici, njima nije dozvoljen unos bilo kakvih podataka, ali imaju priliku da pregledaju bazu podataka filmova, glumaca i režisera. Neregistrovanim korisnicima je takođe dozvoljen pristup i prikaz diskusija na forumima, ali ne i aktivno učestvovanje u njima.

Izabrao sam ovaj projekat za diplomski rad zbog ličnog interesovanja za kinematografiju. Glavni cilj izrade projekta je razvoj softvera koji omogućava onlajn pregled i diskusiju o filmovima. Korisnost softvera je naravno subjektivna, može da varira od osobe do osobe, ali za korisnike koje zanimaju filmovi korisnost je očigledna i široka.

Delotvornost softvera dolazi od različitih sistema koji on poseduje. Sistem za preporuke, na osnovu korisničkih interesovanja, daje sugestije korisnicima koji novi filmovi bi im mogli biti zanimljivi. Posedovanjem foruma, pružena je prilika za formiranje zajednica korisnika koji imaju slična interesovanja. Iako trenutno softver i njegova baza podataka imaju samo podatke o filmovima, softver je pogodan za bilo koje druge oblike medija (muzički albumi, knjige, televizijske serije i sl.)

Postoji veliki broj aplikacija na tržištu koje su slične onoj kojoj sam ja razvio. U narednom delu biće opisane neke od najpoznatijih onlajn baza podataka kinematografije.

IMDb

Započet 1990. godine, Internet Movie Database, poznatiji kao IMDb, je verovatno najpopularnija aplikacija ovog tipa. IMDb sadrži podatke o preko 8 miliona filmova, televizijskih programa, televizijskih serija i video igara, kao i o ljudima koji su radili na njima (glumci, režiseri i drugi članovi filmske ekipe itd.). Iako su IMDb i moja aplikacija dosta slične, postoje znatne razlike. Na primer, IMDb poseduje stranice vezane za zanimljive činjenice i greške u filmovima, kao i kratak opis radnje filma, ali ne poseduje i forum na kojem može da se vrši diskusija. Forumi su postojali do februara 2017. godine kada su ukinuti, iako su bili veoma popularni. Takođe, unos podataka je omogućen svim korisnicima, nakon čega se podaci proveravaju pre nego što se prikažu na veb stranici, dok u mojoj aplikaciji unos podataka je dozvoljen samo od strane administratora.

Rotten Tomatoes

Dok je IMDb više fokusiran na podatke, Rotten Tomatoes veći značaj daje recenzijama i ocenama. Možda najveća razlika između ove dve aplikacije je to što Rotten Tomatoes odvojeno predstavlja ocene publike i ocene kritičara, dok su na IMDb spojeni. Razdvajanje ocena smatram dobrim potezom, s obzirom da se ocene publike i ocene kritičara često razlikuju u velikoj meri, i zbog toga su u mojoj aplikaciji ocene odvojene. Rotten Tomatoes takođe ne poseduje forum za filmove, što smatram greškom. Moja aplikacija je u velikoj meri inspirisana kombinacijom ove dve aplikacije, sa dodatkom određenog broja funkcionalnosti koje smatram korisnim kao i aspektima veštačke inteligencije.

Metacritic

Kao i Rotten Tomatoes, Metacritic daje najveći značaj recenzijama i ocenama. Međutim, glavna prednost Metacritic-a u odnosu na prethodno navedene aplikacije je to što on poseduje podatke o više tipova medija. Korisnici mogu ocenjivati i pisati recenzije ne samo o filmovima i televizijskim serijama, nego i o muzičkim albumima, video igrama i knjigama. Jedan od prvih ciljeva daljeg razvoja mog softvera je proširenje baze na druge tipove medija.

FCS

Filmski centar Srbije je ustanova kulture koja obavlja poslove u oblasti kinematografije. Iako je FCS ustanova koja pruža razne usluge i bavi se velikim brojem različitih poslova u oblasti kinematografije, smatram da je korisno da je napomenem u ovom slučaju zato što FCS poseduje bazu podataka domaćih filmova koja je javno dostupno za pregled. Pregled podataka je moguć po tipovima filmova (dugometražni igrani film, dugometražni dokumentarni film, dugometražni animirani film itd.) kao i po godini prvog prikazivanja.

Svaka od ovih aplikacija ima svoje prednosti i mane, ciljeve i razloge zašto bi korisnici koristili njih a ne neke konkurentne, pa tako i moja ima svojih prednosti u odnosu na druge, pre svega u prisustvu funkcionalnosti koje se ne nalaze u drugim aplikacijama.

Ostatak rada je organizovan u 5 poglavlja :

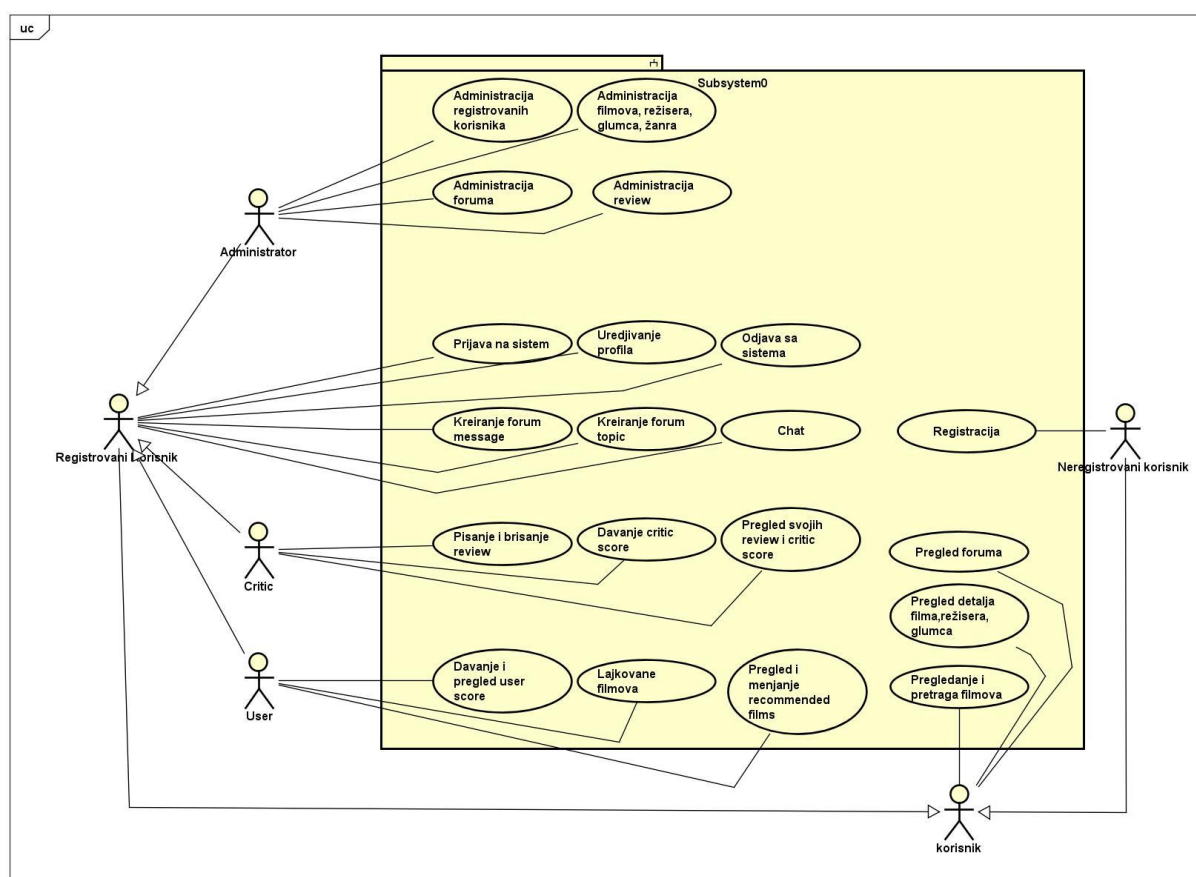
- 2. Specifikacija i model softvera
- 3. Arhitektura softvera
- 4. Pregled korišćenih tehnologija
- 5. Primeri korišćenja
- 6. Zaključna razmatranja

2. Specifikacija i model softvera

Specifikacija i model softvera su predstavljeni UML jezikom (*Unified Modeling Language*). Više informacija o UML jeziku možete naći na stranici dokumentacije [7]. U nastavku poglavlja će biti prikazani dijagram slučajeva korišćenja i dijagram klasa.

2.1 Dijagram slučajeva korišćenja

Dijagram slučajeva korišćenja (*Use Case Diagram*) služi za prikaz mogućih interakcija između korisnika softvera i sistema, kao i da omogući pogled na sistem na višem nivou. U suštini, ovaj dijagram predstavlja pojednostavljen i grafički prikaz onoga šta sistem treba da radi. Korisnici sistema mogu biti registrovani i neregistrovani, dok registrovani korisnici mogu biti administrator, critic i user.



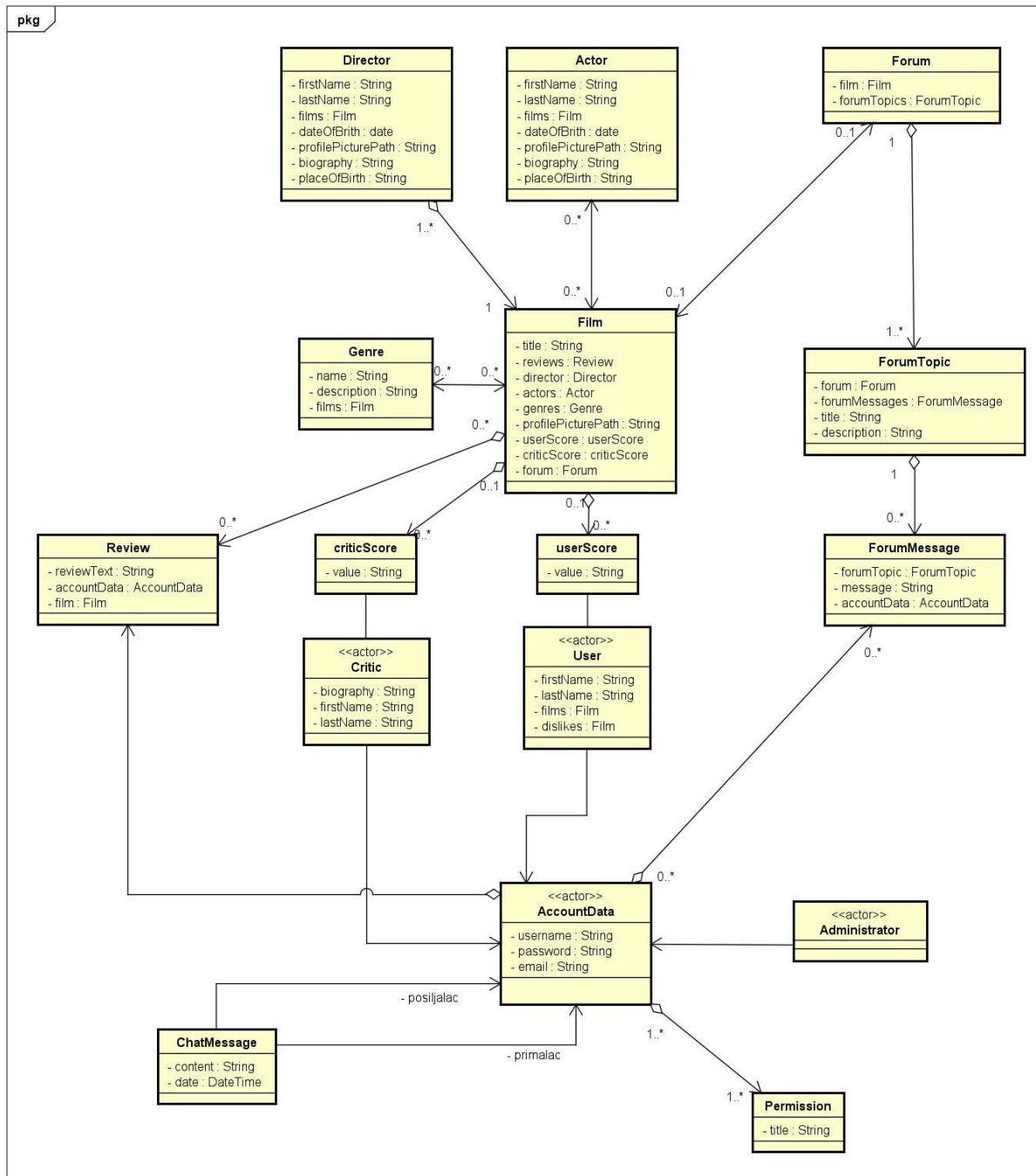
Slika 1 - Dijagram slučajeva korišćenja (Use Case Diagram)

Use Case	Korisnik (Actor)	Opis
Pregledanje i pretraga filma	Korisnik	Pregled filmova i njihova pretraga po zadatom kriterijumu
Pregled foruma	Korisnik	Pristup forumima filmova
Pregled detalja filma, glumca, režisera	Korisnik	Pristup i prikaz detalja filma, glumca ili režisera
Registracija	Neregistrovani korisnik	Registracija na sistem kao običan korisnik (<i>user</i>)
Prijava na sistem	Registrovani korisnik	Prijavljivanje registrovanog korisnika na sistem
Uređivanje profila	Registrovani korisnik	Promena podataka korisničkog profila
Odjava sa sistema	Registrovani korisnik	Odjavljivanje korisnika sa sistema
Kreiranje forum topic	Registrovani korisnik	Kreiranje nove teme za diskusiju na forumu filma
Kreiranje forum message	Registrovani korisnik	Kreiranje nove poruke na odabranoj temi na forumu filma
Chat	Registrovani korisnik	Razmena poruka sa drugim korisnicima sistema
Administracija registrovanih korisnika	Administrator	Pregled, prikaz, promena i brisanje profila korisnika
Administracija filma, režisera, glumca, žanra	Administrator	Pregled, prikaz, promena i brisanje filmova, režisera, glumaca, žanrova
Administracija review	Administrator	Brisanje recenzije filma
Administracija foruma	Administrator	Pregled, prikaz, promena i brisanje foruma, forum topic i forum message
Pisanje i brisanje review	Critic	Pisanje nove recenzije za film i brisanje već postojećih recenzija
Davanje critic score	Critic	Davanje posebne critic ocene filmu
Pregled review i critic score	Critic	Pregled i pristup svih recenzija i ocena koje pripadaju korisniku
Davanje i pregled user score	User	Davanje i pregled user ocena filma
Lajkovanje filma	User	Dodavanje filma u listu filmova koji se korisniku sviđaju
Pregled i menjane recommended films	User	Korišćenje sistema za preporuke filmova

Tabela 1 - Opis slučajeva korišćenja

2.2 Dijagram klasa

Dijagram klasa (*Class Diagram*) je tip dijagrama koji služi da opiše strukturu sistema prikazivanjem klasa sistema, njihovih atributa i odnosa između klasa. Serverska aplikacija koja čini backend sistema i klijentska aplikacija koja čini frontend poseduju iste klase i odnose između njih, tako da će klase oba sistema biti prikazane na istom dijagramu.



Slika 2 – Dijagram klasa softvera

3. Arhitektura softvera

Arhitektura softvera je sačinjena iz tri celine :

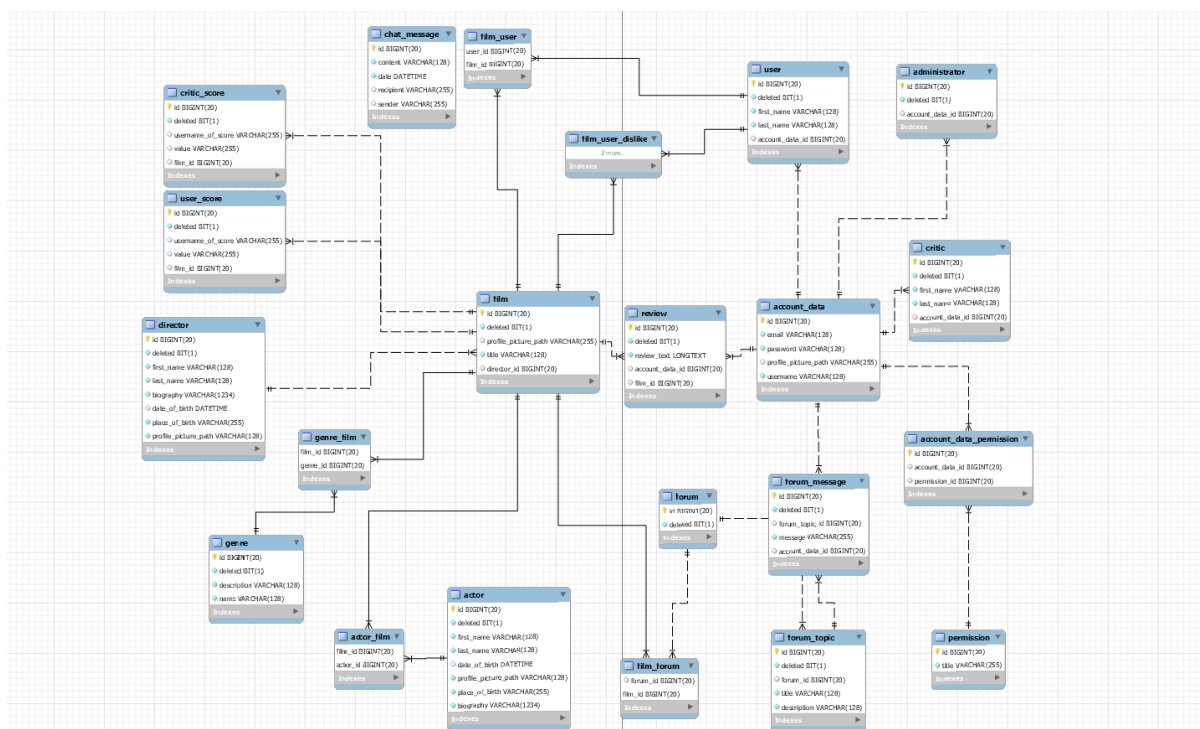
- **Baza podataka** – MySQL relaciona baza podataka
- **Backend** – Serverska aplikacija napravljena u Javi
- **Frontend** – Klijentska aplikacija napravljena u Angular radnom okviru

Pošto je glavni cilj softvera pregled, prikaz, unošenje i izmena podataka iz baze koja sadrži podatke o kinematografiji, počecemo sa opisom baze podataka.

3.1 Baza podataka

Softver koristi relacionu bazu podataka. Kao sistem za upravljanje relacionom bazom podataka (*relational database management system* ili *RDBMS*) koristi se MySQL, o kom će detaljniji opis biti u poglavlju 3.

Glavna karakteristika relacione baze podataka je da su podaci predstavljeni u vidu tabela, gde svaka tabela predstavlja jednu klasu u sistemu dok su kolone tabele atributi klase, a redovi tabele čine jednu instancu klase.



Slika 3 – Model šeme baze podataka

Na slici 3 je predstavljen model šeme baze podataka. Na modelu se mogu videti tabele, odnosno klase, kao i njihove relacije.

Neregistrovani korisnici nemaju svoju klasu u sistemu zato što oni ne poseduju specifične atribute koji bi bili potrebni sistemu, takođe zato što sve interakcije sa sistemom koje su njima dostupne su vezane samo za vizuelni prikaz podataka a ne i njihovo modifikovanje ili unos novih.

Postoje tri vrste registrovanih korisnika : administrator, critic i user. Svaki od njih ima svoje posebne atribute koji se nalaze u njihovim tabelama, a zajednički atributi, kao što su korisničko ime (*username*) i lozinka (*password*), koji svi registrovani korisnici imaju se nalaze u klasi AccountData.

Jedna od karakteristika relacionog modela baze podataka je da u svakoj tabeli jedna kolona mora biti označena kao primarni ključ. Ako je kolona, odnosno atribut klase, označen kao primarni ključ to znači da vrednost te kolone mora biti jedinstvena na nivou tabele zato što se preko te vrednosti identifikuje tačno taj red u tabeli. Označavanjem kolone kao primarni ključ se automatski postavlja ograničenje da vrednost u koloni ne može da ne postoji (*NOT NULL*). Često se u praksi svakoj klasi dodeljuje atribut "id" (skraćeno od *identifier*) koja služi kao primarni ključ. Ako jedan red u tabeli kao vrednost svoje kolone id ima 1, onda nijedan drugi red u koloni ne može da ima 1 kao vrednost svog id. U relacionom modelu baze podataka postoji takođe i strani ključ.

Strani ključ je polje u tabeli koje referiše na kolonu u drugoj tabeli i samim tim omogućava povezivanje između dve tabele. Strani ključ ne mora imati jedinstvenu vrednost na nivou tabele.

Relacije između tabela mogu biti različitih kardinalnosti. Tipovi kardinalnosti između tabela mogu biti jedan prema jedan (*one-to-one*), jedan prema više (*one-to-many*) i više prema više (*many-to-many*). Kardinalnost jedan prema jedan znači da jedan član tabele A može biti povezan sa samo jednim članom tabele B i jedan član tabele B može biti povezan samo sa jednim članom tabele A. Kao primer jedan prema jedan veze u sistemu imamo odnos između klase Film i klase Forum. Svaki film ima samo jedan forum i svaki forum je povezan samo sa tim jednim filmom. Notacijom CascadeType.ALL smo obezbedili da se prilikom kreiranja novog filma odmah kreira i njegov forum, takođe ako obrišemo film obrišaće se i njegov forum. U slučaju jedan prema više, jedan član tabele A može biti povezan sa više članova tabele B, ali član tabele B može biti povezan samo sa jednim članom tabele A. Na primer, film i recenzija (*review*) su u odnosu jedan prema više. Film može da poseduje više recenzija, ali recenzija može biti vezana za samo jedan film. I kao poslednji tip kardinaliteta imamo više prema više u kojem član tabele A može biti u vezi sa više članova tabele B i članovi tabele B mogu biti u vezi sa više članova u tabeli A. U sistemu, tabele film i glumac (*actor*) su u vezi više prema više, film može imati više glumaca, a glumac može biti u više filmova.

Za vršenje upita preko kojih se dobavljaju podaci, relaciona baza podataka koristi upitni jezik SQL (*Structured Query Language*). Primer za upit koristeći SQL bi bio :

```
SELECT * FROM Film WHERE title LIKE %Lord% ;
```

Ovaj upit bi nam iz tabele filmova vratio sve filmove čiji naslov sadrži reč "Lord"

Kao što je prethodno rečeno, stvaranje veze između dve tabele se može vršiti upotrebom stranog ključa, gde tabela ima polje koje referiše na kolonu iz druge tabele. Još jedan način za ostvarivanje veze između tabela je korišćenjem SQL operacije Join.

Join nam omogućava da kreiramo novu tabelu koja se sastoji od izabranih kolona drugih tabela. U softveru postoji nekoliko join tabela, kao primer ćemo uzeti tabelu formiranu kao vezu između tabele film i tabele genre. Pošto su tabela film i tabela genre u vezi više prema više to znači da jedan film može imati više žanrova i jedan žanr može biti povezan sa više filmova. Kako bi smo ostvarili ovu vezu, napravljena je join tabela film_genre koja poseduje kolonu za identifikaciju filma (filmId) i kolonu za identifikaciju žanra (genreId). Preko ove novonastale join tabele možemo videti koji filmovi su u relaciji sa kojim žanrovima i obrnuto.

Zašto relacionala baza podataka?

Postoje različite vrste baza podataka, kao što su relacione, NoSQL (dokument orijentisane i grafovske), Cloud database i sl. Kako bi izbor relacione baze bio jasniji, sledi kratko objašnjenje dokument orijentisanih i grafovskih baza podataka.

Za razliku od relacionih baza podataka gde se podaci čuvaju u tabelama, u dokument orijentisanim bazama podataka podaci se čuvaju u kolekcijama dokumenata. Tabela u relacionom modelu je isto šta i kolekcija u dokument modelu, a redovi u tabeli su dokumenti, gde svaki dokument predstavlja jedan podatak odnosno entitet klase. Dokument orijentisana baza je usmerena na upotrebu podataka, dok je relacioni model usmeren na skladištenje podataka. Jedna od glavnih karakteristika dokument modela je da on ne poseduje šemu, svaki dokument može da ima proizvoljan broj atributa, dok u relacionom modelu svaki red u tabeli ima isti broj atributa odnosno kolona. Dokument sadrži određen broj parova polja i vrednosti kojima je opisan podatak (npr. polje grad i njegova vrednost Novi Sad bi opisali mesto rođenja za dokument koji predstavlja osobu).

Graf orijentisane baze podataka čuvaju podatke u vidu čvorova (*node*) i veza (*edges*). Umesto tabele, podaci su predstavljeni u obliku grafa gde je jedan čvor u grafu isto šta i red u tabeli. Čvorovi predstavljaju nekakve podatke (film, žanr, glumac) a veze opisuju i predstavljaju relaciju između čvorova (sadrži, pripada itd.). U graf orijentisanom modelu, najveća vrednost se daje vezi između podataka, možda i više nego samim podacima.

Postoji više razloga zašto je za softver izabran relacioni model. Prednost relacionog modela je jednostavnost pristupa podacima zato što struktura nije kompleksna, tako da se do podataka može doći jednostavnim upitima. Pošto tabele poseduju primarni ključ, omogućena je jedinstvenost podataka, neće se doći u situaciji da postoje dve iste instance u tabeli. Samim tim što se svaki podatak može samo jednom instancirati, ako se promene vrednosti podatka u tabeli promeniće se i za sve podatke koji su u relaciji sa njim. Ova karakteristika je bitna u softveru jer sistem za preporuke koristi podatke lajkovanih filmova da preporuči nove filmove. Ako bi se podaci o filmu promenili samo u tabeli koja sadrži filmove a ne i na svim mestima gde se ti filmovima referenciraju, onda bi preporuka mogla biti neadekvatna.

Naravno, postoje i nedostaci relacionog modela. Jedan od problema relacionog modela je to što kooperativni rad na različitim verzijama istog softvera nije podržan, što nije predstavljalo problem u razvoju ovog softvera jer sam ga sam razvio. Takođe, pošto unos podataka uglavnom zahteva postojanje drugog podatka (npr. film ne može da se doda ako nema režisera) izbeći će se mogući problem relacionog modela da postoji velik broj tipova a mali broj instanci, odnosno da postoji velik broj tabela koje su slabo popunjene. Performansa i brzina upita zavise od veličine baze odnosno broja tabela, relacione baze su generalno sporije od na primer dokument orijentisanih. Možda najveći problem sa kojim sam se susreo je nastao zbog agilnog pristupa razvoju softveru koji je kao posledicu imao promene strukture i šeme baze podataka u vremenu što ne odgovara relacionom modelu, ali s obzirom na obim projekta nije predstavljao veliku prepreku.

Pošto su većina nedostataka relacione baze podataka izbegnute, a prednosti iskorišćenje za efikasniji i kvalitetniji razvoj softvera, smatram da je relacioni model bio dobar izbor za razvoj ovog softvera.

3.2 Backend

Backend deo softvera služi za komunikaciju između baze podataka i servera na koji postavlja podatke iz baze i sa kojeg preuzima podatke koje frontend pošalje.

Serverska aplikacija razvijena u Javi, uz upotrebu radnog okvira Java Spring Framework (više informacija u poglavlju 3), čini backend ovog softvera. Aplikacija je sačinjena iz slojevite arhitekture (*layered architecture pattern*). Komponente koje imaju slične funkcionalnosti su organizovane u horizontalne slojeve (*horizontal layer*), tako da svaki sloj ima specifičnu ulogu unutar aplikacije. Broj slojeva nije unapred definisan, on može biti potpuno proizvoljan. Glavni cilj ovakve strukture je raspodela softvera u manje celine čime se omogućava skalabilnost, fleksibilnost i modularnost softvera.

Serverski deo softvera se sastoji iz tri sloja :

1. Sloj repozitorijuma
2. Sloj servisa
3. Sloj kontrolera

3.2.1 Sloj repozitorijuma

Sloj repozitorijuma, odnosno sloj podataka, služi za pristup bazi podataka. U ovom sloju se nalaze interfejsi (*interface*) koji implementiraju JpaRepository. JpaRepository sadrži API za CRUD operacije kao i API za paginaciju i sortiranje.

CRUD (*create, read, update, delete*) operacije su kreiranje, čitanje, ažuriranje i brisanje i one predstavljaju osnovne operacije za manipulaciju podataka. Implementaciju ovih operacija vrši radni okvir Hibernate, tako da implementacija za osnovne operacije nije potrebna od naše strane. Ovaj radni okvir je zadužen za mapiranje Java klasnih objekata u tabele u bazi podataka. Hibernate omogućava velik broj predefinisanih upita (*query*). Ukoliko pak želimo neke specifične ili složenije upite, njih možemo dodati koristeći Hibernate-ov jezik za upite HQL (*Hibernate Query Language*). HQL je u velikoj meri sličan SQL-u, s tim da je glavna razlika što HQL radi sa objektima i njihovim atributima dok SQL radi sa tabelama. Hibernate prevodi HQL upite u SQL upite čime se vrše operacije nad bazom podataka. Međutim, moguće je Hibernate-u direktno davati SQL upite, što je slučaj u ovom softveru.

3.2.2 Sloj servisa

Sloj servisa je zadužen za poslovnu logiku konkretne aplikacije. Iz ovog sloja se šalju zahtevi za dobavljanje podataka sloju repozitorijuma. Svi servisi koji su vezani za istu klasu objekta (film, glumac, režiser itd.) su grupisani u jednu celinu, tako da postoji jasna i pregledna organizacija servisa. Grupisanjem servisa smanjuje se posledica promena jer će one uglavnom uticati samo na jednu grupu servisa, što znatno pojednostavljuje proces održavanja servisa. Dakle, sloj servisa je sa jedne strane povezan sa slojem repozitorijuma, a sa druge strane je povezan sa slojem kontrolera.

3.2.3 Sloj kontrolera

Sloj kontrolera služi za komunikaciju između sloja servisa i servera putem HTTP zahteva. U ovom sloju se navodi HTTP adresa servera od kojeg će se dobijati zahtevi, kao i adresa na kojoj će se postavljati podaci vezani za klase. HTTP zahtevi se šalju od strane klijenta zarad pristup podacima na serveru. HTTP zahtev se šalje sa određenom metodom, najčešće korišćene metode su POST, GET, PUT i DELETE, a one predstavljaju osnovne CRUD funkcionalnosti, odnosno kreiranje, čitanje, ažuriranje i brisanje. U zavisnosti kojom metodom je poslat zahtev obaviće se određena funkcionalnost. Preko ovog sloja prilikom obrade zahteva dobijamo i HTTP status na osnovu koga možemo da vidimo da li je zahtevana funkcionalnost uspešno izvršena ili je došlo do greške. Verovatno najpoznatiji status je status 404 – Not Found, do kog dolazi kada server ne može da dobavi ono šta je klijent tražio.

3.3 Frontend

Frontend je komponenta softvera koja omogućava komunikaciju i interakciju između korisnika i softvera. Samim tim, frontend predstavlja korisnički interfejs softvera, koji je skoro uvek grafičke prirode. Najjednostavnije rečeno, frontend je onaj deo softvera koji korisnik vidi, bila to veb stranica ili nekakva druga aplikacija. Klijentska aplikacija napravljena uz pomoć Angular radnog okvira čini frontend ovog softvera.

Klijentska aplikacija je GUI aplikacija (*Graphic User Interface*) koja omogućava korisniku interakciju sa sistemom. Preko nje korisnik dobija pristup podacima i određenim uslugama, u zavisnosti od tipa korisnika.

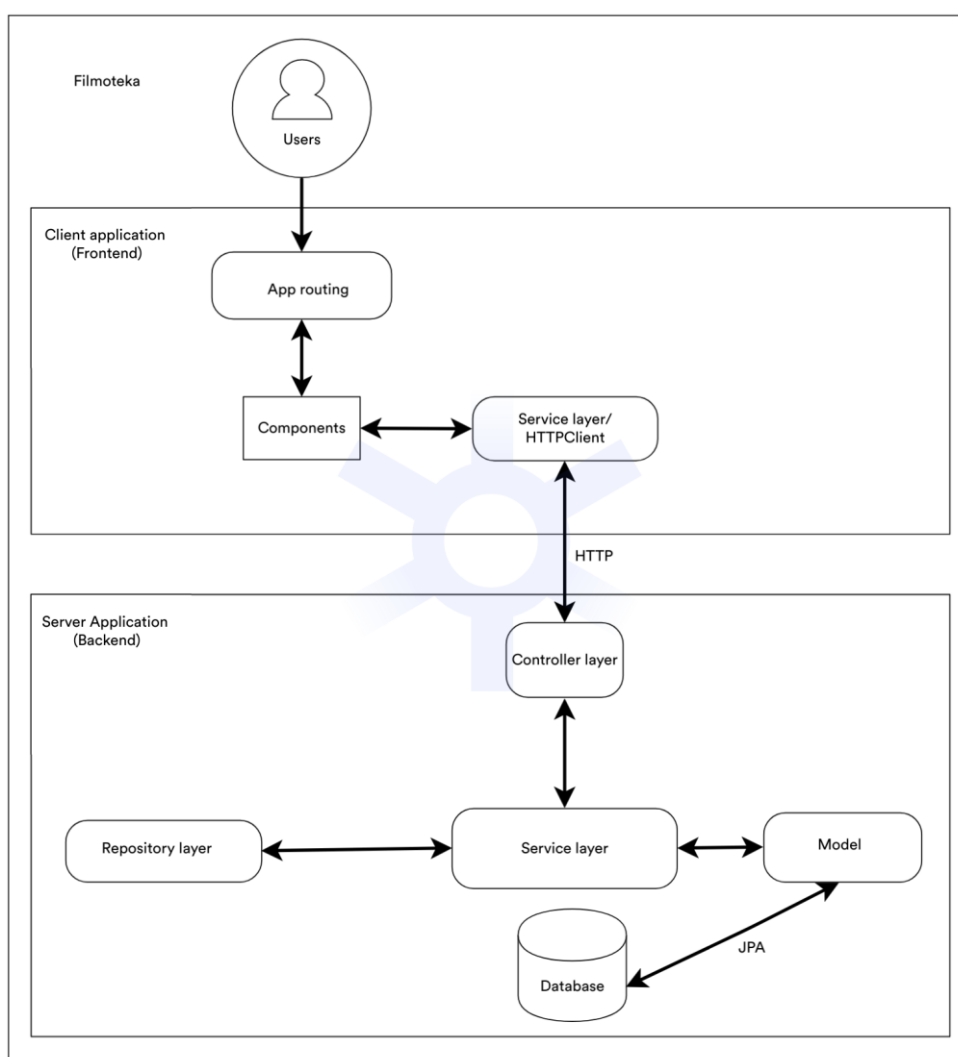
Slično kao što je u serverskoj aplikaciji implementirana slojevita arhitektura, razvoj korisničke aplikacije je baziran na komponentama. Razlika je u tome što na frontend delu svi fajlovi koji su vezani za klasu se grupišu u istu komponentu, bez obzira na tip njihove funkcionalnosti. Dakle, na backend delu bi repozitorijum, servis i kontroler vezani za klasu film bili raspoređeni u tri sloja (sloj repozitorijuma, sloj servisa i sloj kontrolera), dok bi na frontend delu servis, model i svi drugi relevantni delovi vezani za film bili zajedno u komponenti film. Kada se u Angular radnom okviru kreira komponenta ona će po početnim postavkama (*default*) imati 4 fajla : html, css, i dva typescript fajla. Više o ovim tipovima fajlova će biti reči u poglavlju 3, ali ukratko rečeno, html fajl služi za formiranje veb stranice, css za modelovanje izgleda stranice, jedan typescript fajl za testiranje i jedan za implementaciju funkcionalnosti komponente. Praksa je da se u sklopu komponente napravi i servisni fajl u kome će se definisati funkcije za dobavljanje podataka određene klase sa servera, kao i funkcionalnosti za njihovo modifikovanje (brisanje, ažuriranje i kreiranje novog podataka). Struktura komponente je proizvoljna, može da sadrži bilo koji različit broj i različite tipove fajlova, naravno ako Angular podržava korišćenje tih vrsta fajlova.

Klijentska aplikacija i serverska aplikacija komuniciraju i vrše interakciju putem HTTP zahteva. Svi kontroleri u serverskoj aplikaciji su mapirani na svoju jedinstvenu adresu. Klijentska aplikacija preko svog servisnog sloja šalje HTTP zahtev na određenu adresu, u zavisnosti koji kontroler joj je potreban odnosno sa kojom klasom podataka želi da vrši interakciju. U zavisnosti sa kojom metodom je poslat HTTP zahtev se pristupa određenoj funkcionalnosti kontrolera koja iz servisnog sloja poziva potrebnu metodu. Ako, na primer, HTTP zahtev sadrži instancu klase film i poslat je metodom POST, onda se u kontroleru filma poziva metoda koja će u bazu podataka dodati poslani film. Isti proces se vrši za brisanje, ažuriranje i pregled filmova kada se pošalje zahtev metodama DELETE, PUT i GET. Nakon primanja i izvršavanja zahteva, dobija se HTTP status na osnovu koga možemo da vidimo da li je bio neki problem, ili ako je zahtev uspešno izvršen dobićemo odgovarajući status, kao što je 200,OK ili 201,Created.

Pošto smo prošli kroz arhitekturu softvera, možemo da napravimo kratki rezime kako softver u stvari funkcioniše.

3.4 Dijagram arhitekture i sažetak funkcionisanja softvera

Komponente softvera su baza podataka (*MySQL*), backend (serverska aplikacija) i frontend (klijentska aplikacija). Baza podataka služi za čuvanje i smeštaj svih podataka u njihove odgovarajuće tabele. Backend služi za obradu podataka kao i za dobavljanje podataka iz baze i njihovo postavljanje na server gde će biti vidljivi frontend-u. Frontend je grafički korisnički interfejs u vidu veb stranice koji preuzima podatke sa servera gde ih je backend postavio, prikazuje korisniku relevantne podatke i omogućava mu određene usluge. Ukoliko korisnik želi da modifikuje neke podatke (dodavanje, brisanje, ažuriranje) onda frontend šalje te podatke backendu koji ih obrađuje na zahtevani način i postavlja u bazu podataka. Smisao ovakve arhitekture je da frontend, odnosno korisnik, nema direktnu interakciju sa bazom podataka, već se ona kontrolisano vrši putem backend komponente.



Slika 4 – Dijagram arhitekture softvera

4. Pregled korišćenih tehnologija

Za razvoj softvera korišćene su neke od najpoznatijih i najpopularnijih programskih jezika i tehnologija za kreiranje veb aplikacija. U ovom poglavlju će biti malo više reči o svakoj od njih.

4.1 Baza podataka - SQL, MySQL i MySQL Workbench

Prvo ćemo proći kroz tehnologije upotrebljene za razvoj i implementaciju baze podataka softvera. **SQL** (*Structured Query Language*) je relacioni upitni jezik koji služi za kreiranje, modifikovanje i dobavljanje podataka iz relacione baze podataka. **MySQL** je sistem za upravljanje relacionom bazom podataka koji radi kao server i obezbeđuje korisnički interfejs za pristup bazi. MySQL omogućava korisniku da direktno pristupi bazi putem SQL upita. Za razvoj softvera je korišćen i **MySQL Workbench** koji služi kao vizuelni alat za dizajniranje, razvoj i administraciju relacione baze podataka.

4.2 Backend – Java, Java Spring, Hibernate ORM

Java je objektno-orijentisani programski jezik baziran na klasama. Razvoj Java programskog jezika je zasnovan na 5 principa, a oni su da jezik mora biti :

1. Jednostavan i objektno-orijentisan
2. Robustan i siguran
3. Neutralan i prenosiv
4. Interpretiran, višenitan i dinamičan
5. Mora da poseduje visok nivo performansi

Na osnovu ovih principa su se razvile karakteristike Jave. Detaljnije o Javi i njenim karakteristikama možete pronaći na [8]. Za rad sa Java programskim jezikom korišćena je programska razvojna okolina **Eclipse**.

Java Spring Framework je jedan od najpopularnijih radnih okvira za razvoj Java aplikacija, kao i veb aplikacija. Spring je zasnovan na principu umetanja zavisnosti (*dependency injection*). U objektno-orijentisanom programiranju, zavisnost je veza između dva ili više objekta u kojoj je za implementaciju jednog objekta potreban drugi objekat. Umetanje zavisnosti je tehnika pomoću koje izbegavamo zavisnost. Upotrebom ove tehnike, Spring omogućava razvoj aplikacije čije komponente i slojevi mogu da obavljaju funkcije nezavisno jedna od druge. Zbog posedovanja velikog broja različitih modula, Spring nam pruža raznovrsne funkcionalnosti i servise.

Perzistencija podataka, odnosno mogućnost čuvanja podataka i nakon prestanka rada aplikacije, je jedan od fundamentalnih koncepata razvoja aplikacija. Za implementaciju ovog koncepta koristimo jedan od Java Spring Framework modula, Hibernate.

Hibernate ORM (ili samo Hibernate) je alat koji vrši mapiranje objekata na relacije (*object-relational mapping*, odnosno *ORM*) i time omogućuje perzistenciju objekata. ORM predstavlja preslikavanje Java objekata na tabele relacione baze podataka na osnovu anotacija koje opisuju pravila preslikavanja. ORM u suštini samo transformiše podatke iz jednog oblika u drugi. Više informacija o Java Spring Framework na [9], a o Hibernate ORM na [10].

4.3 Frontend– Angular 8, Typescript, HTML, CSS

Klijentska veb aplikacija je razvijena uz pomoć **Angular** radnog okvira. Angular je radni okvir razvijen od strane kompanije Google. Treba razlikovati Angular (poznat kao Angular 2+), koji je zasnovan na Typescript programskom jeziku, i AngularJS, od koje je nastao Angular, koji kao primarni jezik koristi JavaScript. Kao što je prethodno rečeno, arhitektura Angulara se sastoji iz hijerarhije komponenata. Za rad sa Angularom i njegovim komponentama korišćen je uređivač koda **Visual Studio Code**. Komponente Angular aplikacije se sastoje iz Typescript, HTML i CSS fajlova.

Typescript je programski jezik koji razvija kompanija Microsoft. On je ustvari verzija programskog jezika JavaScript kome je dodata opcionalna statička tipizacija i objektna orijentisanost. U jeziku sa statičkom tipizacijom, pre izvršenja operacije moraju se proveriti tipovi varijabli da bi se sprečila greška u tipu podataka. Typescript je nastao zbog nedostataka JavaScript jezika za razvoj velikih aplikacija, samim tim, Typescript je dizajniran za razvoj velikih aplikacije za izvršavanje na klijentu ili serveru.

HTML (*HyperTextMarkup Language*) je opisni jezik (*markup language*) koji se koristi kao standard za opis veb stranica. Pomoću njega možemo da organizujemo raspored veb stranice, gde i kako će se prikazivati elementi stranice kao što su naslovi, paragrafi, slike, liste, tabele i slično. HTML dokument se preko servera pošalje veb pregledaču (*web browser*) koji na osnovu dokumenta renderuje i prikazuje veb stranicu. HTML standard održava W3C (*World Wide Web Consortium*) a trenutno aktuelna verzija standarda je HTML 5.

CSS (*Cascading Style Sheets*) je jezik formatiranja koji služi za modifikovanje izgleda web stranica definisanih u HTML ili XML dokumentu. Dakle, dok HTML i XML definišu strukturu i sadržaj stranice, CSS definiše konkretan izgled stranice. Pomoću CSS dokumenta možemo menjati font i izgled slova, dimenzije tabela i lista, boju elemenata itd.

Detaljniji opis Typescript [11], HTML [12] i CSS [13] se može naći na njihovim dokumentacijama.

4.4 Ostale tehnologije

Dok su prethodno opisane tehnologije dominantno doprinele razvoju softvera, postoje i druge tehnologije, biblioteke i alati koji su pomogli boljem razvitku aplikacije. U daljem delu poglavlja će biti par reči o njima.

JWT (*JSON Web Token*) - JSON veb token ima značajnu ulogu u softveru zato što je preko njega organizovana sigurnost sistema. Token se sastoji iz tri celine : Header, Payload i Signature. U Header delu se nalazi informacija koji algoritam se koristi za generisanje Signature dela i koji tip tokena se koristi (u ovom slučaju, JWT tip). Signature služi da potvrdi da je onaj koji tvrdi da je poslao token stvarno i poslao token i da osigura da podaci u tokenu nisu promenjeni. Payload je nama najznačajniji deo tokena zato što se u njemu, u JSON obliku, nalaze podaci koje smo poslali preko tokena. Prilikom prijave korisnika na sistem generiše se token koji u svom payload delu sadrži informaciju o kom tipu korisnika je reč. U zavisnosti od tipa, korisniku se pruža pristup određenim stranicama i uslugama.

STOMP (*Simple (or Streaming) Text Oriented Message Protocol*) - STOMP je komunikacijski protokol čija je osnovna namena siguran i pouzdan prenos podataka putem tekstualnih poruka. Uz pomoć ovog protokola je implementiran sistem za razmenu tekstualnih poruka između korisnika.

OpenLayers – OpenLayers je JavaScript biblioteka koja omogućava prikazivanje geografskih mapa u veb pretraživaču. Konkretno, ovom bibliotekom je implementirana mapa koja prikazuje mesto rođenja odabranog glumca ili režisera. Geolokacija traženih mesta se vrši upotrebom **HERE Maps API**.

TensorFlow – TensorFlow je biblioteka za mašinsko učenje i veštačku inteligenciju. Putem ove biblioteke je implementirana neuronska mreža koja pomaže pri administraciji foruma. Kada korisnik želi da postavi novu poruku na forumu filma, neuronska mreža, koja je obučena na skupu od 2 miliona podataka, klasifikuje unesenu poruku. Ako je poruka klasifikovana kao neprimerena (sadrži uvrede i vulgaran jezik) onda ona neće biti dodata na forum. Ovim je znatno olakšan posao administratora pri vođenju foruma.

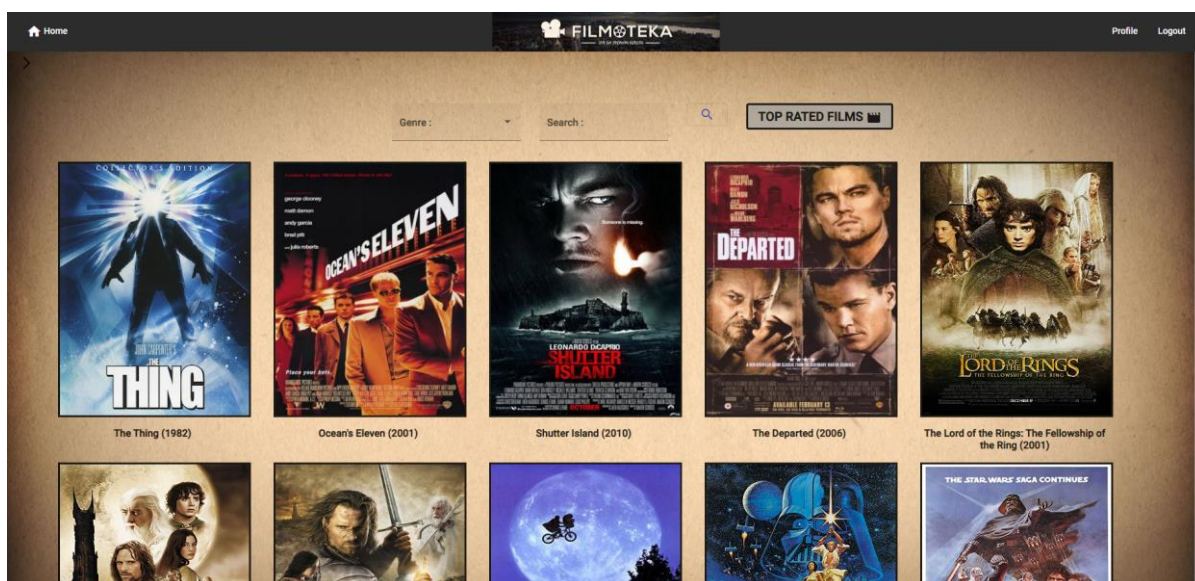
XML (*Extensible Markup Language*) - XML je opisni jezik veoma sličan HTML-u ali za razliku od njega omogućuje korisniku da definiše sopstvene formate podataka. Iako prisustvo XML jezika u softveru nije obimno, ono se u stvari sastoji iz samo jednog fajla, prisustvo je ipak veoma značajno. U serverskoj aplikaciji softvera nalazi se POM (*Project Object Model*) fajl napisan u XML jeziku. U ovom fajlu se nalazi konfiguracija serverskog dela na osnovu koje se pravi projekat. U POM fajlu možemo navesti zavisnosti (*dependencies*), pluginove, verziju projekta, opis, listu developera i slično.

Apache Maven – Maven je alat za automatizaciju izgradnje softvera koji se uglavnom koristi za aplikacije napravljene u Javi. Dva osnovna aspekta za koje je Maven zadužen su način na koji je softver izgrađen i zavisnosti (*dependencies*) softvera. Serverska aplikacija softvera je napravljena uz pomoć Maven alata.

5. Primeri korišćenja

Sada kad smo opisali specifikaciju i arhitekturu softvera kao i tehnologije koje su korišćene za njegov razvoj, možemo proći kroz konkretne primere korišćenja softvera uz pomoć implementiranih funkcionalnosti.

5.1 Početna stranica (*Homepage*)

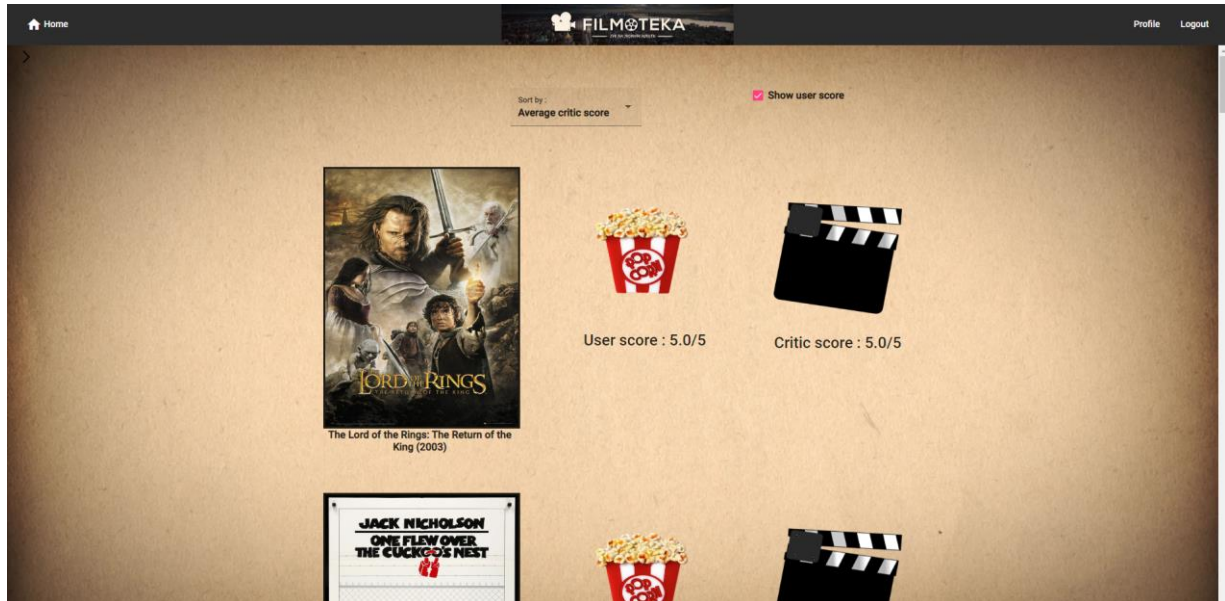


Slika 5 – Početna stranica veb sajta

Izgled većine stranica se dinamički menja u zavisnosti od tipa korisnika koji im pristupa. Ovo nije slučaj sa početnom stranicom, ona je ista za sve korisnike. Na početnoj stranici se automatski prikazuju svi filmovi koji se trenutno nalaze u bazi podataka. Korisniku je omogućeno filtriranje filmova po žanrovima kao i pretraga. Pretraga je moguća po nazivu filma, imenu režisera, imenu glumaca i godini prvog prikazivanja filma. Na ovoj stranici korisniku je omogućen izbor konkretnog filma ili odlazak na stranicu najbolje ocenjenih filmova.

5.2 Pregled najbolje ocenjenih filmova

Svaki film ima ocene recenzenta i ocene korisnika. Na ovoj stranici su prikazani filmovi sa najboljom prosečnom ocenom. Korisnik može da bira na osnovu koje prosečne ocene će se sortirati filmovi. Ukoliko želi, prilikom sortiranja filmova po najboljoj prosečnoj oceni recenzenta korisnik može da uključi i prikaz prosečne ocene korisnika, kao i obrnuto. Izbor filma je moguć i na ovoj stranici, klikom na sliku postera filma odlazi se na stranicu koja sadrži detalje izabranog filma.

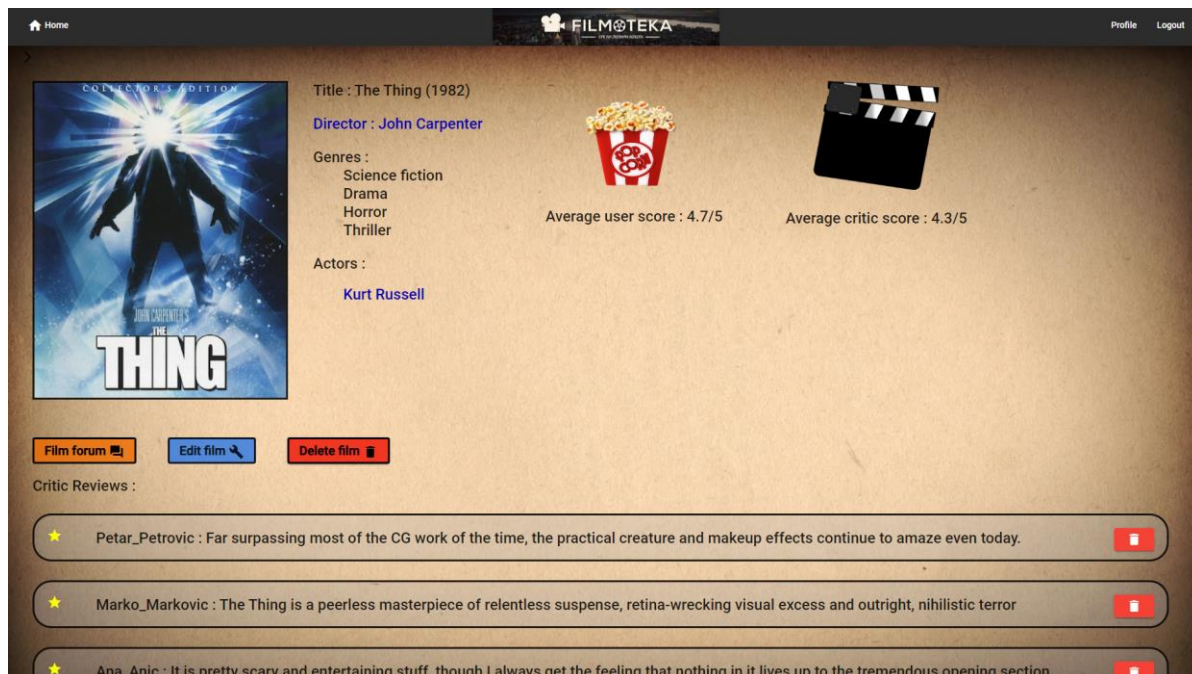


Slika 6 – Stranica za prikaz najbolje ocenjenih filmova

Kao što je slučaj sa početnom stranicom, i ova stranica je ista za sve tipove korisnika bez obzira da li su prijavljeni na sistem ili ne.

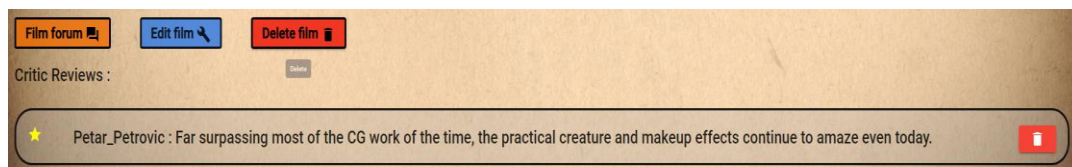
5.3 Stranica Filma

Izborom filma, ili putem početne stranice ili preko stranice za najbolje filmove, smo došli do stranice na kojoj se nalaze podaci za izabran film.



Slika 7 – Stranica filma

Na ovoj stranici možemo da vidimo podatke filma : naziv, godina prvog prikazivanja, režiser, glumci, ocene korisnika, ocene kritičara i recenzije. U zavisnosti od tipa korisnika koji posećuje stranicu prikazaće se određene komponente stranice. Ukoliko je korisnik administrator, biće prikazane opcije za modifikovanje podataka filma, brisanje filma kao i brisanje pojedinačnih recenzija. Ako je korisnik recenzent, njemu će biti omogućeno pisanje recenzije i davanje ocene kritičara, svaki recenzent može da napiše samo jednu recenziju i da da samo jednu ocenu, ukoliko promeni mišljenje može da menja oba ova podatka koja je prethodno uneo. O funkcionalnostima običnih korisnika će kasnije biti detaljniji pregled.



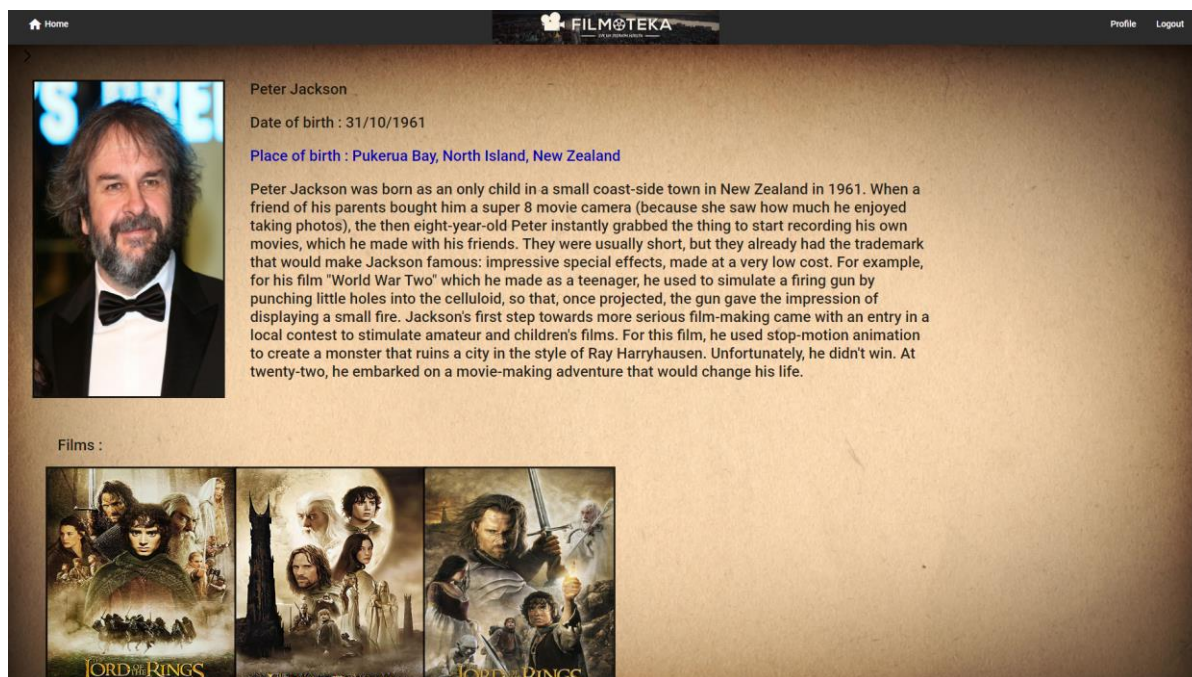
Slika 8 – Funkcionalnosti administratora



Slika 9 – Funkcionalnosti recenzenta

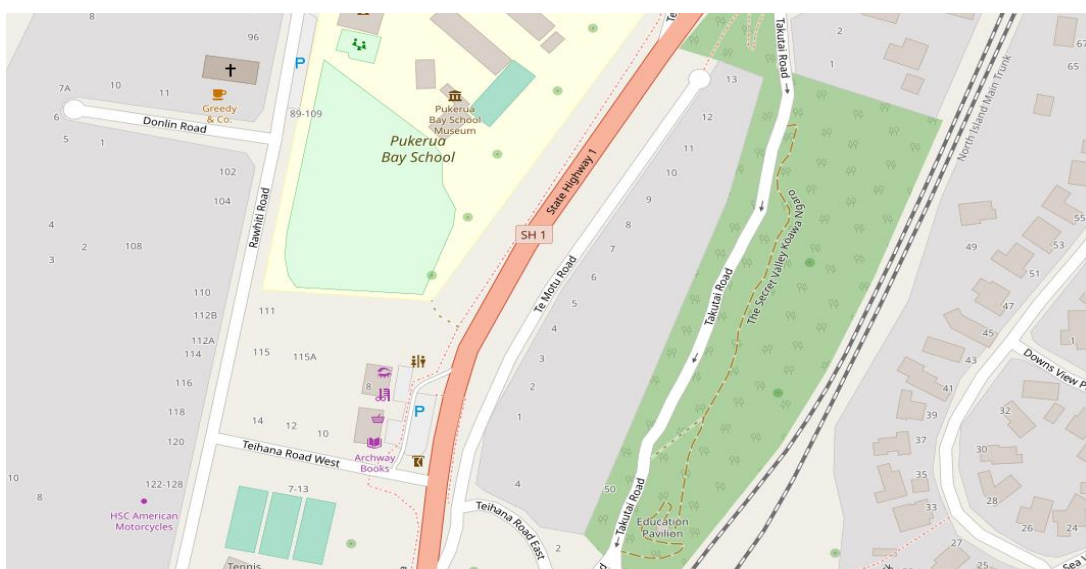
5.4 Stranica glumca i stranica režisera

Na stranici filma je moguće klikom izabrati režisera ili glumca čime će se otvoriti njihova stranica. Stranica režisera i stranica glumca su dve posebne komponente na frontend delu softvera, ali pošto trenutno imaju isti izgled i pružaju jednake usluge, ove dve stranice će biti predstavljene putem stranice režisera.



Slika 10 – Stranica režisera

Ovde vidimo detalje o režiseru kao i njegove filmove. Posebna funkcionalnost koja je omogućena na ovoj stranici je prikaz mesta rođenja upotrebom OpenLayers mape.



Slika 11 – OpenLayers mapa mesta rođenja izabranog režisera ili glumca

5.5 Toolbar

Na vrhu svake stranice je uvek prisutni toolbar. Uz pomoć njega korisnik može lako da se vrati na početnu stranicu. Toolbar menja izgled u zavisnosti da li je korisnik prijavljen na sistem.



Slika 12 – Toolbar za korisnika koji se nije prijavio na sistem

Ako korisnik nije prijavljen na sistem, nude mu se opcije registracije na sistem putem pravljenja novog korisničkog naloga, ili ako korisnik već ima napravljen nalog može da izabere opciju prijave na sistem. U slučaju da se korisnik prijavio na sistem, ove dve opcije su zamenjene sa druge dve. Jedna za odlazak na profil korisnika gde se korisniku prikazuju njegovi podaci kao i mogućnost njihovog ažuriranja, a druga opcija je odjavljivanje sa sistema.

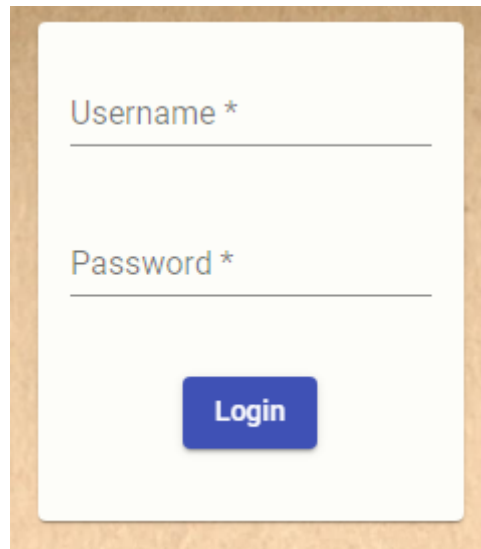
5.6 Registracija novog korisnika

Korisnik se registruje na sistem kreiranjem novog korisničkog naloga. Pošto tipovi korisnika administrator i recenzent (*critic*) imaju posebne privilegije i pristup određenim stranicama, jedini tip korisnika koji se može kreirati na ovaj način je običan korisnik (*user*). Za kreiranje *user* naloga potrebno je uneti tražene podatke koji se nalaze u dve grupe. Jedna grupa podataka su podaci koje poseduju svi registrovani korisnici, a drugoj grupi pripadaju podaci koji su posebni za korisnika tipa *user*. Za unos određenih podataka postoje uslovi koji se moraju ispuniti. Korisničko ime mora da se sastoji iz najmanje tri karaktera, dok lozinka mora da sadrži bar osam. Polje *Confirm password* treba da se sastoji iz istog podatka kao i polje *Password*, a email mora da sadrži karakter @ i da ne bude duži od 64 karaktera. Sva polja moraju biti popunjena. Korisniku se nudi i mogućnost izbora profilne slike.

Slika 13 – Registracija novog korisnika

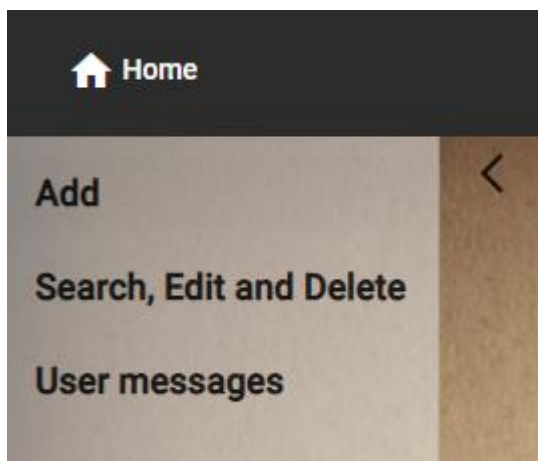
5.7 Prijava na sistem

Prijavu na sistem korisnik vrši unosom korisničkog imena i lozinke. Korisnik će biti obavešten ukoliko se korisničko ime i/ili lozinka koju je uneo ne nalaze u bazi podataka. Ukoliko je korisnik uneo adekvatne podatke, biće prijavljen na sistem i kreiraće se token koji sadrži ulogu odnosno tip prijavljenog korisnika.

A login form with a light beige background and a thin brown border. It contains two input fields: 'Username *' and 'Password *', both with horizontal lines below them. Below the password field is a blue rectangular button with the word 'Login' in white text.

Slika 14 – Prijava na sistem

5.8 Side navigation menu



Prilikom prijave na sistemu korisniku će se prikazati nova komponenta, navigacioni meni (*side navigation menu*) koji se nalazi na levom delu stranice. Preko ovog menija korisnik može da pristupi stranicama koje su dostupne samo njegovom tipu korisnika. Za administratore se omogućava prikazivanje, unošenje, ažuriranje i brisanje svih korisnika i podataka koje se nalaze u bazi. Recenzentu se prikazuju stranice za prikaz njegovih ocena i recenzija, a *user*-u pristup sistemu za preporuke.

Slika 15 – Administrator side navigation menu

5.9 Administrator CRUD operacije

Jedna od najznačajnijih uloga administratora u sklopu vođenja veb stranice je vršenje CRUD (kreiranje, prikaz, ažuriranje, brisanje) operacija nad podacima. Administrator može da kreira nove korisnike kao i da kreira filmove i sve podatke vezane za film (glumci, režiser, žanrovi). Komponente za kreiranje novih podataka se koriste i kao komponente za ažuriranje podataka. Ukoliko administrator izabere opciju modifikovanja već postojećeg podatka, komponenta za kreiranje novog podatka će se otvoriti i polja za unos će biti popunjena sa vrednostima izabranog podatka. Uneti podaci se preko servera i HTTP zahteva šalju odgovarajućem kontroleru na backend delu softvera. Ukoliko je poslat podatak koji treba da se doda u bazu kao novi onda se HTTP zahtev šalje metodom POST, a ukoliko se radi o ažuriranju već postojećeg podatka novim vrednostima onda se koristi metoda PUT.

Postoje dva načina za brisanje podataka : fizičko brisanje i logičko brisanje. Ukoliko se podatak fizički briše on će se u potpunosti ukloniti iz baze podataka. Sa druge strane, implementacija logičkog brisanja se vrši dodavanjem posebnog atributa svakoj klasi u sistemu kojoj želimo da prisvojimo logički način brisanja. Klasi se dodaje proizvoljno nazvan atribut (u praksi se najčešće atribut nazove “deleted”) koji je tipa *boolean*, odnosno može da poseduje jednu od dve vrednosti : *true* ili *false*. Backend-u se kaže da iz baze podataka uzima samo podatke čija vrednost ovog atributa je trenutno *false* i da samo njih postavi na server. Logičko brisanje umesto potpunog brisanja podatka iz baze podrazumeva samo menjanje ovog posebnog atributa na *true*. Ovim smo postigli da podatak ostane u bazi ali on neće biti postavljen na server i samim tim za korisnike on je u suštini obrisani. Prednost logičkog brisanja se ispoljava u slučaju kada želimo da ponovo vratimo obrisani podatak u bazu. U toj situaciji, dovoljno je samo promeniti vrednost jednog atributa i ceo podatak sa svim svojim atributima i njihovim vrednostima je ponovo dostupan za sistem. Ako podatak obrišemo fizički i kasnije želimo da ga vratimo, onda moramo ručno dodavati potpuno novi podatak koji ima iste vrednosti kao onaj koji smo obrisali. Jedan od nedostataka logičkog brisanja je upravo to što se podatak ne obriše u potpunosti iz baze i time zauzima prostor i čini bazu robusnijom nego što ona može biti. U softveru je trenutno implementirano fizičko brisanje podataka ali postoje atributi i metode u backend delu za logičko brisanje, tako da u slučaju da se želi promeniti način brisanja podataka to se može učiniti veoma jednostavno i brzo.

Poslednja usluga koja se pruža administratoru putem *administrator side navigation menu* komponente, koja je dostupna samo administratoru, je pretraga i pregled podataka. Administrator bira koje podatke želi da mu se prikažu, bilo to neki tip korisnika ili podaci vezani za film. Svi podaci izabranog tipa će biti prikazani u tabeli koja sadrži njihove najznačajnije attribute, kao što su korisničko ime, ime i prezime, naziv filma ako se radi o filmu itd. U tabeli se takođe nalaze prečice za ažuriranje ili brisanje izabranog podatka. Omogućena je i pretraga podataka koji se trenutno nalaze u izabranoj tabeli. U polje za pretragu ne mora da se unese tačna vrednost atributa po kom se pretražuje, dovoljno je da se u atributu nalazi deo unete vrednosti. Takođe, tokom korišćenja funkcionalnosti za pretragu korisnik ne mora da obraća pažnju na mala i velika slova prilikom unosa tražene vrednosti.

Rezultati pretrage podataka se prikazuju u tabeli koja ima prethodno definisan broj redova, odnosno broj pronađenih podataka, ali korisnik može da poveća broj prikazanih rezultata pretrage na veličinu koja njemu odgovara. Na narednoj slici je prikazan rezultat pretrage svih filmova koji se trenutno nalaze u bazi podataka.

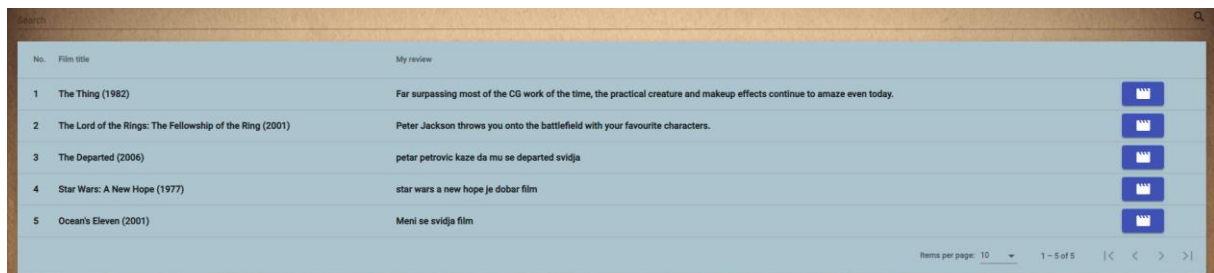


No.	Title
1	The Thing (1982)
2	Ocean's Eleven (2001)
3	Shutter Island (2010)
4	The Departed (2006)
5	The Lord of the Rings: The Fellowship of the Ring (2001)
6	The Lord of the Rings: The Two Towers (2002)
7	The Lord of the Rings: The Return of the King (2003)
8	E.T. (1982)
9	Star Wars: A New Hope (1977)
10	Star Wars: The Empire Strikes Back (1980)

Slika 16 – Pregled i pretraga filmova od strane administratora

5.10 Prikaz ocena i recenzija kritičara

Na isti način na koji je administratoru omogućena pretraga i prikaz svih podataka sistema, recenzentu su pruža ista usluga za njegove napisane recenzije i date ocene. U tabeli se nalazi dugme za odlazak na izabrani film gde recenzent može da promeni svoju ocenu ili recenziju.



No.	Film title	My review
1	The Thing (1982)	Far surpassing most of the CG work of the time, the practical creature and makeup effects continue to amaze even today.
2	The Lord of the Rings: The Fellowship of the Ring (2001)	Peter Jackson throws you onto the battlefield with your favourite characters.
3	The Departed (2006)	petar petrovic kaze da mu se departed sviđa
4	Star Wars: A New Hope (1977)	star wars a new hope je dobar film
5	Ocean's Eleven (2001)	Meni se sviđa film

Slika 17 – Pregled i pretraga recenzija kritičara

5.11 Sistem za razmenu poruka

Softver sadrži sistem za razmenu tekstualnih poruka između administratora i drugih korisnika. Sistem je implementiran putem mrežnog soketa (*web socket*) i ranije pomenutog STOMP protokola. Mrežni soket, ili internet soket, je krajna tačka dvosmernog međuprocenog komunikacionog toka koji je baziran na nekom protokolu. U suštini, mrežni soketi služe za uspostavljanje perzistentne komunikacije između klijenta i servera putem koje se mogu slati podaci između njih.

Osnovna svrha sistema za razmenu poruka je pružanje korisnicima uslugu za kontaktiranje administratora softvera u slučaju da imaju problem koji može biti rešen samo od strane administratora. Svaka poruka sadrži tekst poruke, datum slanja i profilnu sliku korisnika koji je poslao poruku. U aktivnoj iteraciji sistem podržava samo slanje tekstualnih poruka. Prvo proširenje sistema bi se sastojalo iz dodavanja mogućnosti slanja fajlova (slika, dokumenata i sl.). Takođe, sistem trenutno samo pruža uslugu slanja poruka između korisnika i administratora, a ne i slanje poruka između korisnika i drugih korisnika. Međutim, proširenje sistema da sadrži opciju slanja poruka između korisnika bi se moglo implementirati jednostavno i efikasno, ukoliko se proceni da sistem treba da sadrži ovakvu funkcionalnost.



Slika 18 – Primer upotrebe sistema za razmenu poruka

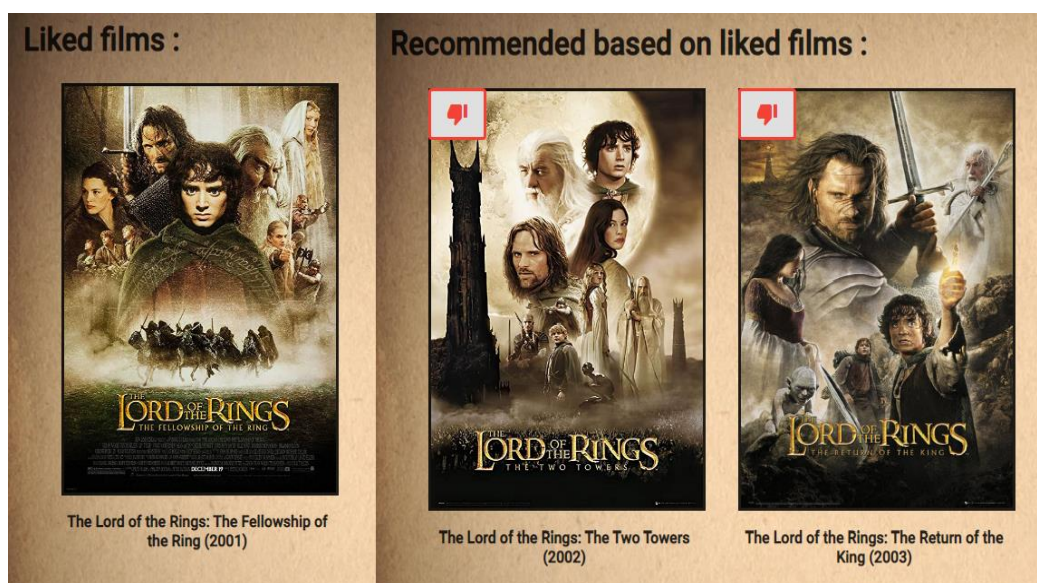
5.12 Sistem za preporuke

Najspecifičnija usluga koja se pruža korisnicima tipa *user*, a ujedno i jedna od najznačajnijih funkcionalnosti softvera, je pristup sistemu za preporuke.

Sistem je implementiran putem binarnog klasifikatora i funkcionira na osnovu liste lajkovanih filmova korisnika. Kada korisnik pristupi stranici nekog filma, ima priliku da ga lajkuje odnosno da označi da mu se film sviđa. Lista svih filmova koji su na ovaj način označeni su dostupni korisniku putem njegovog menija za navigaciju. Korisnik naravno može i da ukloni film iz ove liste, ukoliko to želi. Na osnovu liste lajkovanih filmova sistem određuje koje karakteristike poseduju filmovi koji se korisniku sviđaju. Kada korisnik želi da mu se prikaže lista preporučenih filmova sistem će tek onda početi proces preporuke, tako da lista preporučenih filmova nije podatak koji se čuva već se ona dinamički instancira, na osnovu trenutno aktivnih kriterijuma, svaki put kada korisnik želi da dobije preporuke.

Binarni klasifikator funkcionira tako što prolazi kroz listu svih filmova, iz koje su prethodno izbačeni filmovi koje je korisnik lajkovao da se korisniku ne bi preporučio film koji je već označio da mu se sviđa, i od svakog filma uzima određene karakteristike kao unos. Te karakteristike su žanrovi, glumci i režiser filma. Svaki od ovih unosa ima svoju zadatu težinu prilikom donošenja odluke. Ukoliko na osnovu unosa i njihovih težina film pređe zadati prag (*threshold*) on će biti klasifikovan kao film koji se korisniku dopada i biće dodat u listu za preporuke. Lista preporuka će prikazivati maksimalno pet filmova da se ne bi zatrpao ekran velikim brojem preporuka, ukoliko one postoje.

Nakon dobijanja liste preporuka korisnik ima dve mogućnosti ukoliko želi da dobije nove preporuke : može ili da lajkuje film, čime će on automatski biti uklonjen kao potencijalna preporuka, ili u slučaju da se korisniku ne sviđa preporučeni film i želi da ga zameni novom preporukom, onda može da označi određeni film da više ne bude preporučen. Isto kao što ima pristup listi lajkovanih filmova, korisnik ima pristup listi filmova koje je označio da ne želi da mu se preporučuju, i ukoliko želi može da ukloni film iz liste i odabrani film će mu opet biti preporučen.

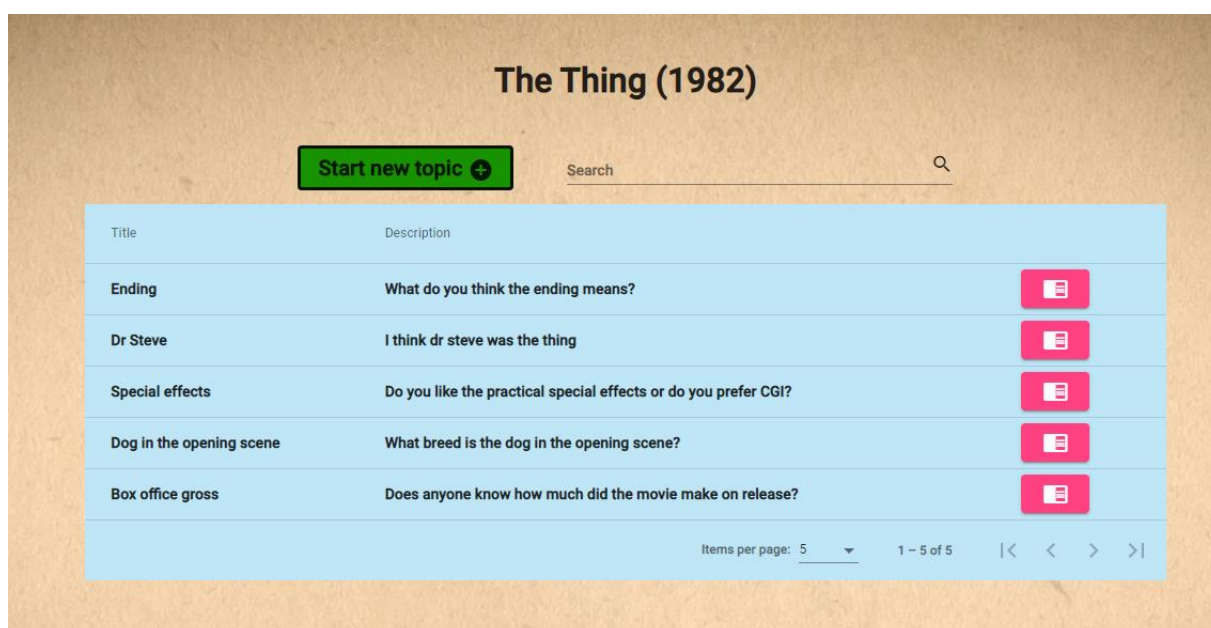


Slika 19 – Primer upotrebe sistema za preporuke

5.13 Forum

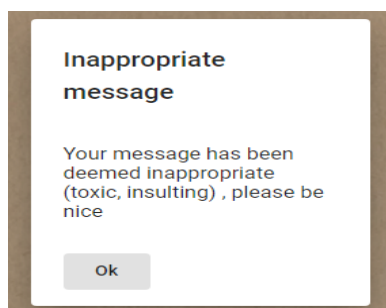
Svaki film koji se nalazi u sistemu poseduje i svoj specifičan forum na kome se mogu obavljati diskusije o filmu. Svi korisnici softvera, bilo registrovani ili neregistrovani, ulogovani ili ne prijavljeni na sistem, imaju pristup forumu. Samo korisnici koji su prijavljeni na sistem mogu da aktivno učestvuju na forumu.

Forum se sastoji iz tema (*forum topic*) koje korisnici postavljaju. Svaka tema ima svoj naslov i kratki opis. Korisnici zatim mogu da izaberu određenu temu i u njoj da postavljaju poruke (*forum message*). Svi tipovi korisnika mogu da započnu novu temu i da pišu nove poruke na izabranoj temi.



Slika 20 – Forum filma *The Thing*

Kao pomoć administratoru prilikom održavanja foruma, u sklopu foruma je implementiran sistem za filtriranje poruka. Sistem se sastoji iz modela neuronske mreže koja je prethodno obučena na skupu podataka.



Slika 21 – Filtriranje poruke

Kada korisnik proba da unese novu poruku na izabranoj temi, ona će prvo proći kroz sistem za filtriranje poruka. Ukoliko se poruka klasifikuje kao neprimerena, onda se ona neće postaviti na forum i korisnik će biti obavešten da ne može da šalje poruke uvredljive prirode. Ukoliko pak poruka ne bude klasifikovana kao neprimerena onda će nova poruka biti dodata na forum.

6. Zaključna razmatranja

U ovom radu opisana je specifikacija, arhitektura i implementacija softvera za upravljanje registrom kinematografije koji je kao svoje glavne ciljeve imao obezbeđivanje mogućnosti pregleda i modifikovanja baze podataka kinematografije, preporučivanje filmova korisnicima i organizovanje foruma za diskusiju o filmovima. Smatram da su ciljevi razvoja softvera uspešno izvršeni putem implementacije komponenti i funkcionalnosti koje se nalaze unutar aplikacije.

Softver predstavlja slojevit veb aplikaciju koja se sastoji iz tri glavne komponente: baza podataka, serverska aplikacija i klijentska aplikacija. Relaciona baza podataka napravljena uz pomoć MySQL Workbench alata služi za skladištenje svih podataka relevantnih za rad softvera. Serverska aplikacija čini backend ovog softvera, ona je napisana prvenstveno u objektno-orijentisanom programskom jeziku Java uz korišćenje Java Spring Framework radnog okvira. Frontend se sastoji iz klijentske aplikacije razvijene u Angular radnom okviru, u najvećoj meri koristeći programski jezik Typescript, kao i opisne jezike HTML i CSS za definisanje strukture i izgleda veb stranice. Za rad sa backend delom projekta korišćen je Eclipse uređivač koda, a za frontend Visual Studio Code. Nijedna od ovih komponenti nije opcionalna, svaka od tri glavne komponente ima vrlo značajnu ulogu u procesu rada softvera. Funkcionisanje softvera i ostvarivanje ciljeva projekta ne bi bilo moguće bez ijedne od njih.

Imajući u vidu da se svrha i cilj softvera ostvaruju upotrebom softvera od strane krajnjih korisnika, glavna osnova za dalji razvoj softvera bi bila bazirana na povratnim informacijama dobijenim od korisnika (*user feedback*). Ove informacije su od neizmernog značaja zato što preko njih možemo dobiti odličan uvid za dalji razvoj funkcionalnosti, grafičkog interfejsa i generalno načina upotrebe softvera. Naravno, povratne informacije od svih tipova korisnika bi bile od velikog značaja, ali najveći fokus bi se dao informacijama dobijenim od običnih korisnika, odnosno korisnika tipa *user*.

Za razvoj softvera je upotrebljen agilni pristup koji se zasniva na razvoju softvera u vidu iteracija. Prilikom završetka razvoja iteracije ona se testira i daje se ocena funkcionisanja i implementacije na osnovu koje se planira razvoj naredne iteracije softvera. Kao primer, u prvobitnoj iteraciji softver nije posedovao sistem za filtriranje poruka na forumu, OpenLayers mape i stranicu za sortiranje filmova po ocenama korisnika i ocenama kritičara. Neke od funkcionalnosti planirane za implementaciju u narednoj iteraciji softvera su :

1. Proširenje baze podataka
2. Stranica kritičara
3. Mobilna verzija klijentske aplikacije

Proširenje baze podataka – Trenutno softver pruža mogućnost pregleda baze podataka koja sadrži samo podatke vezane za kinematografiju. Prvi cilj daljeg razvoja softvera je proširenje baze da sadrži druge tipove medija kao što su knjige, muzički albumi i televizijske serije. Korisniku bi se onda na početnoj stranici pružila opcija da bira koji tip medija želi da pregleda, a polja za pretraga bi u svoje rezultate pretrage uključivala samo izabrani tip medija (dodala bi se i mogućnost da pretraga uključuje sve tipove medija, ukoliko korisnik to želi). Ovakvo proširenje ne bi trebalo da predstavlja veliku prepreku, s obzirom da bi bila potrebna samo implementacija potrebnih klasa (knjiga, autor, album itd.) na backend delu i veb stranice za njihov prikaz i pristup na frontend delu. Postojeće klase i njihove veze bi se mogle iskoristiti kao šablon za dodavanje novih klasa jer, na primer, klase glumac i film su u vezi više prema više, što bi bio isti slučaj kod klasa autor i knjiga, jedan autor može da napiše više knjiga i knjiga može biti napisana od strane više autora. Zanimljiva funkcionalnost ovakvog proširenja, koju nisam video na drugim aplikacijama ovog tipa, bi bila prikazivanje knjige na osnovu koje je napravljen film.

Stranica kritičara - Jedan manji ali koristan dodatak za softver bi bilo postojanje stranice kritičara na kojoj se mogu videti podaci o kritičaru kao i sve njegove ocene i recenzije. Ukoliko korisnik primeti da se u većini slučajeva slaže sa određenim kritičarom, putem ove stranice bi mogao da vidi kojim filmovima je taj kritičar dao najveće ocene i na taj način bi korisnik dobio listu filmova koji bi mu verovatno bili zanimljivi. Implementacija ove stranice bi omogućila i razvoj novog sistema koji korisniku prikazuje kritičare koji su dali najsličnije ocene kao on.

Mobilna verzija klijentske aplikacije – Sve više korisnika koristi mobilne uređaje za pristup veb stranicama, u tolikoj meri da velik broj aplikacija ima značajno više korisnika koji pristupaju putem mobilnih i tablet uređaja nego uz pomoć kućnih računara i laptopova. Imajući ovo u vidu, razvoj mobilne verzije klijentske aplikacije bi bilo od ključnog značaja sa komercijalne tačke gledišta. Klijentska aplikacija je u potpunosti namenjena upotrebi putem računara, tako da bi pristup ovoj aplikaciji preko mobilnog uređaja doneo neodgovarajuće rezultate. U razvojnom okruženju Android Studio bi bila razvijena verzija klijentske aplikacije za upotrebu od strane mobilnih uređaja koji koriste Android operativni sistem. Trenutna klijentska aplikacija i mobilna verzija bi bile dve potpuno odvojene aplikacije, tako da u slučaju kada se jednoj od aplikacija ne može pristupiti zbog procesa održavanja (*maintenance*) korisnici bi mogli da pristupe softveru putem druge aplikacije.

I bez ovih funkcionalnosti koje će biti implementirane u narednoj iteraciji, smatram da softver i u trenutnoj verziji sadrži funkcionalnosti koje mu daju prednost u odnosu na druge aplikacije. Najznačajnije usluge koje se pružaju korisniku u trenutnoj iteraciji softvera su sistem za preporuke, sistem za filtriranje poruka na forumu, sistem za razmenu poruka između korisnika i postojanje foruma za diskusiju.

Literatura

1. D. Živković, *Java programiranje*, Univerzitet Singidunum, Beograd, 2021.
2. M. Dobrojević and N. Bačanić Džakula, *Veb programiranje*, Univerzitet Singidunum, Beograd, 2021.
3. N. Kojic, *Web dizajn: HTML, CSS i JavaScript*, Univerzitet Singidunum, Beograd, 2020.
4. M. Milosavljević, *Veštačka inteligencija*, Univerzitet Singidunum, Beograd, 2019.
5. V. Tomašević, *Razvoj aplikativnog softvera*, Univerzitet Singidunum, Beograd, 2019.
6. M. Veinović, G. Šimić, A. Jevremović and M. Tair, *Baze podataka*, Univerzitet Singidunum, Beograd, 2018.
7. <https://www.uml.org/> (dostupno dana : 07.09.2021)
8. <https://docs.oracle.com/en/java/> (dostupno dana : 07.09.2021)
9. <https://docs.spring.io/spring-framework/docs/current/reference/html/> (dostupno dana : 07.09.2021)
10. <https://hibernate.org/orm/documentation/5.5/> (dostupno dana : 07.09.2021)
11. <https://www.typescriptlang.org/docs/> (dostupno dana : 07.09.2021)
12. <https://devdocs.io/html/> (dostupno dana : 07.09.2021)
13. <https://developer.mozilla.org/en-US/docs/Web/CSS> (dostupno dana : 07.09.2021)