SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Borna Radinović

SUSTAV ZA UPRAVLJANJE POLUSTRUKTURIRANIM BAZAMA PODATAKA MongoDB UZ PRIMJER IMPLEMENTACIJE ON-LINE VIDEOTEKE

ZAVRŠNI RAD

Varaždin, 2018.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Borna Radinović

Matični broj: 44124/15–R

Studij: Informacijski sustavi

SUSTAV ZA UPRAVLJANJE POLUSTRUKTURIRANIM BAZAMA PODATAKA MongoDB UZ PRIMJER IMPLEMENTACIJE ON-LINE VIDEOTEKE

ZAVRŠNI RAD

Mentor:

izv. prof. dr. sc. Markus Schatten

Varaždin, rujan 2018.

Borna Radinović

Izjava o izvornosti

Izjavljujem da je moj završni rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristila drugim izvorima, osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne te prihvatljive metode i tehnike rada.

Autorica potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

_____

Sažetak

Cilj završnog rada je proučiti, analizirati i opisati sustav za upravljanje polustrukturiranim bazama podataka MongoDB, NoSQL (engl. Not [only] SQL) pristup oblikovanju baze podataka te opis formata podataka JSON (engl. Java Script Object Notation). U radu je implementirana web aplikacija on-line videoteke (bez poslužitelja za streaming sadržaj) s pripadnom strukturom korisnika (administrator, korisnik), u kojoj korisnik ima mogućnost odabira, pretraživanja i kupovanja (zajma) filma iz baze podataka te pregleda relevantnih podataka za odabrani film, dok administrator ima mogućnost dodavati, ažurirati i brisati podatke filma iz baze podataka te ima uvid u korisnike. Završni rad sadrži uvod, pregled polustrukturiranog modela podataka, pregled NoSQL sustava MongoDB, opis implementacije aplikacije, kritički prikaz, zaključak i literaturu.

Ključne riječi: baze podataka; json; polustrukturirani podaci; nosql; mongodb; on-line videoteka

Sadržaj

1. Uvod

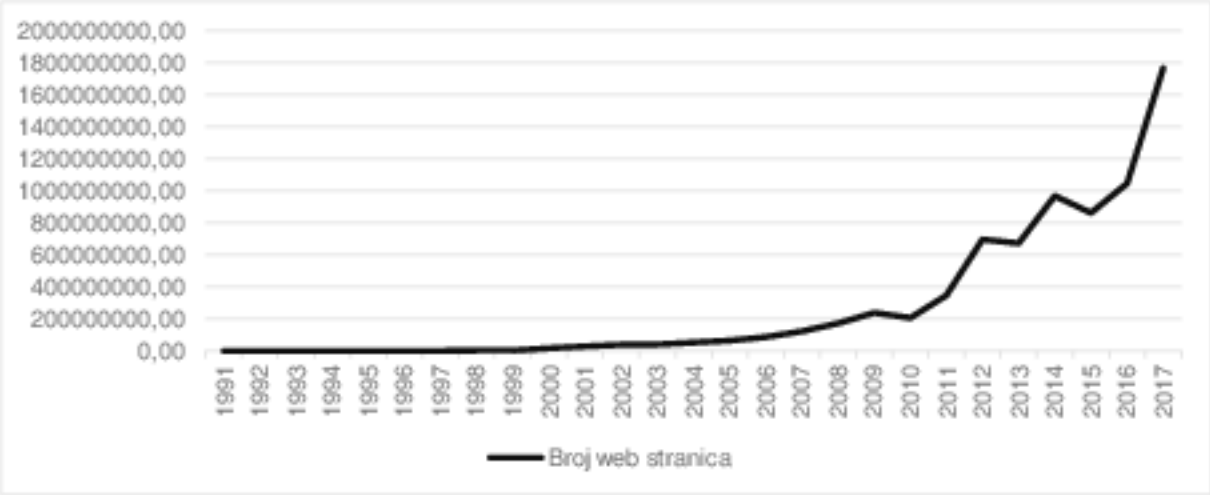
U današnje doba, zbog opsega korištenja World Wide Weba, došlo je do velikog rasta količine podataka, čija je struktura promjenjiva i sve teže odrediva. Podatke se prenosi u sve većim količinama te broj zahtjeva za podacima raste. Zbog svega navedenog, stvorila se potreba za manje strukturiranim načinom spremanja podataka, od onog poznatog u relacijskom modelu, koji bi nudio veću skalabilnost razvoju baza podataka.

Odgovor na ove potrebe daju polustrukturirane baze podataka, koje omogućavaju pohranu podataka u obliku koji je dovoljno strukturiran za potrebe razvoja, ali nije strogo organiziran kao u relacijskom modelu.

U ovom radu biti će predstavljen MongoDB, besplatni sustav otvorenog koda za upravljane polustrukturiranom bazom podataka. Ovaj sustav biti će prikazan razvojem polustrukturiranog podatkovnog modela i web aplikacije on-line videoteke za posudbu filmova od strane korisnika i upravljanje takvim sustavom od strane administratora.

2. Polustrukturirani podaci

World Wide Web (WWW) nastao je s ciljem organizirane i otvorene razmjene informacija. [1] Kroz godine, količina podataka na webu rapidno se povećala, što se može vidjeti na grafu ispod, koji prikazuje broj web stranica od začetka WWW do danas. Graf je izrađen prema podacima Real Time Statistics projekta, koji prikuplja podatke o webu u stvarnom vremenu, te su isti referencirali izumitelj weba Time Berners Lee i organizacija za standardizaciju tehnologija na webu, World Wide Web Consortium (W3C). [2]



Slika 1 Graf broja Web Stranica od 1991. do 2017.

Povećanjem količine informacija na internetu, povećala se i veličina podataka koji se prenose mrežom, a za lakši prijenos potrebno je podatke dekomponirati za slanje.

Jedan pogled na prijenos informacija je dokument dostupan preko URL-a (engl. *Uniform Resource Locator*), kojeg je korisnik preuzeo navođenjem tog URL-a. Za tu paradigmu razvijen je HTML, koji prikazuje čitke informacije korisniku i nudi povezivanje raznih dokumenata putem poveznica. Drugi pogled je onaj baze podataka, kod kojeg se naglasak stavlja na strukturiranje podataka, kako bi se oni pohranili na što efikasniji način i iz kojih se informacija može dobiti izvršavanjem upita. Kada dvije organizacije razmjenjuju informacije, ona koja je izvor podataka svoje podatke iz baze podataka strukturirat će i staviti u dokument dostupan preko URL-a, a zatim će druga morati prikupiti informacije s dokumenta, prije nego što može započeti obradu jer nema pristup samoj bazi podataka. Zbog lakše razmjene informacija, shvatilo se kako se treba napraviti drugačije rješenje, koje bi premostilo problem pristupa do baze preko dokumenata. [3]

Zbog potrebe za prijenosom većih količina podataka, do potrebe za bržom razmjenom informacija, došlo je do razvoja polustrukturiranih baza podataka. [3]

Polustrukturirani podaci su podaci koji ne zahtijevaju shemu, te ih se često karakterizira kao samoopisujuće. [3], [4] Koristeći određenu jednostavnu sintaksu, polustrukturirani podaci prikazuju točno one podatke koji su potrebni za korištenje. Za prikaz se upotrebljava neka vrsta sintakse, u kojoj se svaka vrijednost zapisuje skupa s ključem koji ju je opisuje, na primjer { ime: 'Borna', Prezime: 'Radinović' email: "bradinovi@foi.hr"}, Ključevi bi bili Ime i Prezime, a vrijednosti "Borna" i "Radinović". [3] Zbog ključ-vrijednost sintakse, jasno je razaznati što koja vrijednost predstavlja, a to takve podatke čini u jednoj mjeri strukturiranim i od tuda dolazi naziv polustrukturirani podaci.

2.1. Teorija grafova pri prikazu polustrukturiranih podataka

Autori Maleković i Schatten za prikaz polustukturiranog modela podataka koriste teoriju grafova [5], isti koncept koriste i autori Abiteboul, Buneman i Suciu [3].

U nastavku ću navesti sve definicije teorije grafova, bitne za shvaćanje prikaza polustukturiranog modela podataka.

Graf

Graf G je uređeni par (V, B), gdje je V skup vrhova, a B skup bridova uređenih parova skupa V, pri čemu je $B \subseteq V \times V$. [6]

Usmjereni graf

Neka je V skup vrhova i neka je $B \subseteq V \times V$ skup bridova. Bridovi skupa B uređeni su parovi vrhova skupa V. Za svaki brid vrijedi $b = (v_i, v_j)$, gdje v_i nazivamo izvorište brida, a v_j odredište brida. Izvorište brida ima oznaku $(izv(b))$, a odredište $(odr(b))$. Usmjereni graf G je uređeni par (V, B).

Put

Neka je $G = (V, B)$ usmjereni graf. Put od izvorišta $izv(b_1)$ do odredišta $odr(b_k)$ je svaki niz bridova $b_1/b_1/ \ldots b_k$ u kojem vrijedi da je $\big(odr(b_i)\big) = (izv(b_{i+1}))$ za $i = 1, 2, \ldots, k-1$. Broj bridova k na putanji naziva se duljina puta.

Korijen

Neka je $G = (V, B)$ usmjereni graf i neka je v_k vrh skupa V. Za vrh v_k se kaže da je korijen grafa, ako postoji put od v_k do svakog vrha v_i iz skupa V gdje $i \neq k$.

Ciklus

Na usmjerenom grafu $G = (V, B)$ ciklus je svaki put $b_i/b_{i+1}/ \ldots b_k/b_i$, to jest svaki put koji počinje u vrhu v_i i završava u vrhu v_i.

Aciklički graf

Graf u kojemu nema ciklusa.

Stablo

Za usmjereni graf $G = (V, B)$ kaže se da je stablo, ako i samo ako vrijedi da postoji jedinstveni put od vrha v_k do svakog vrha v_i iz skupa V gdje $i \neq k$.

List

Na usmjerenom grafu $G = (V, B)$ za neki vrh $v \in V$, kaže se da je list tog grafa, ako ne postoji niti jedan $b \in B$ koji je izvorište $izv(b) = v$.

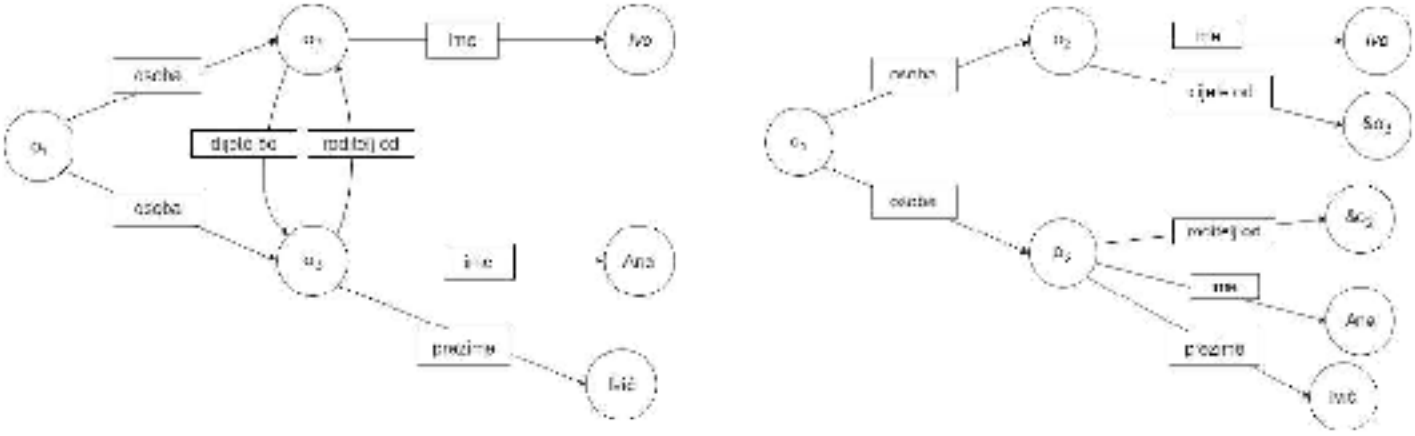
Podatkovni graf

Za usmjereni graf $G_p = (V, B)$ kaže se da je podatkovni graf ako ima označene bridove, a vrhovi su mu podatkovni objekti dvije vrste, atomarni i složeni.

Atomarni objekti su listovi, a složeni objekti imaju bridove s drugim vrhovima.

Neka je skup vrhova V_a skup identiteta atomarnih objekata, a V_s skup identiteta složenih objekata. Vrijedi da je $V_a \cap V_s = \emptyset$. Podatkovni graf može se promatrati kao uređenu trojku $G_p = (V_a \cup V_s, B, k)$, k je korijen grafa G_p i element skupa $V_a \cup V_s$.

Podatkovni grafovi su najčešće aciklični.



Slika 2 Podatkovni graf zapisan kao cikličan graf (lijevo) i isti graf u obliku peudostabla (desno) (Prema: Mirko Maleković i Markus Schatten, 2017)

Na slici 2 prikazan je primjer podatkovnog grafa iz knjige Teorija baza podataka [5]. Lijevi podatkovni graf prikazuje složene objekte o_2 i o_3 , od kojih o_2 ima brid ime prema atomarnom

objektu Ivo, dok o_3 ima brid ime i prezime prema atomarnim objektima Ana i Ivić, respektivno. Ovaj graf se može zapisati i na drugi način koji je vidljiv na grafu desno. Budući da lijevi graf nije stablo i ima ciklus, može ga se transformirati u pseudostablo.

Na lijevom grafu vrijedi da je

$\big(izv(roditelj\ od)\big) = (odr(dijete\ od))$ i $\big(odr(roditelj\ od)\big) = (izv(dijete\ od))$ te iz grafa vidimo da su $izv(roditelj\ od), odr(dijete\ od)), odr(roditelj\ od), izv(dijete\ od)) \in V_s$.

Pseudostablo dobivamo korištenjem reference na identitete objekata, umjesto direktnim pokazivanjem na objekt. Referenca na identitet objekta označava se s prefiksom &, te na grafu desno za bridove grafa roditelj od i dijete od, vrijedi da je $odr(dijete\ od)$ referenca na objekt o_3, a $odr(roditelj\ od)$ je referenca na objekt o_2.

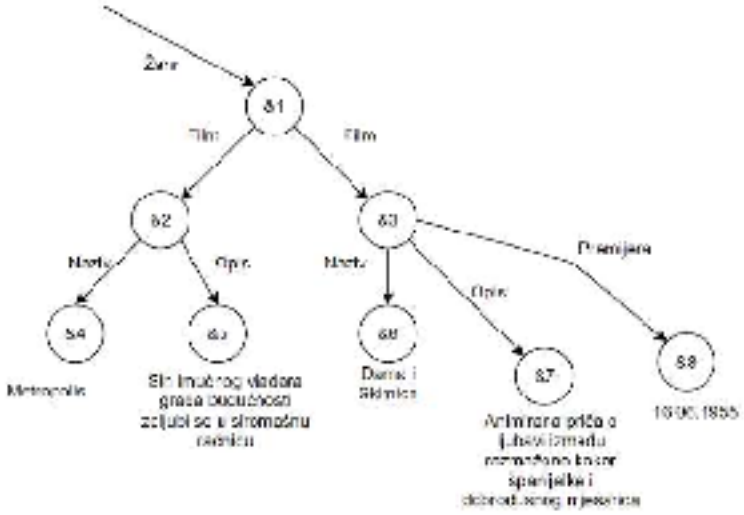
2.2. Object Exchange Model (OEM)

Object Exchange Model (*OEM*), podatkovni je model nastao na projektu Tsimmis i smatra se standardom prikaza polustrukturiranog modela podataka [3].

OEM prikazuje podatke polustukturiranog modela u obliku usmjerenog grafa s označenim bridovima. Vrhovi grafa su objekti složene ili atomarne vrste, a bridovi grafa predstavljaju veze između objekata. [4] Iz svega navedenog, može se zaključiti kako je OEM podatkovni graf.

Svaki objekt u OEM modelu je uređena četvorka (oznaka brida, jedinstveni identifikator objekta, tip, vrijednost). [4] Oznaka brida tekstualna je vrijednost, tip je atomarni ili složeni. Složeni objekti za vrijednost imaju liste identifikatora objekata [3], a ukoliko se smatraju atomarnim objektima imaju vrijednost realnog ili prirodnog broja, teksta i tako dalje. [4]

Na slici iznad, prikazan je OEM graf sa složenim objektom žanr, koji za vrijednost ima



Slika 3 Primjer OEM grafa

listu referenci na 2 objekt filma. U čvorovima grafa označeno notacijom &broj, nalazi se jedinstveni identifikator objekta. Objekti filma &2 ima atomarne vrijednosti Naziv i Opis , a objekt filma &3 ima atomarne vrijednost Naziv, Opis i Premijera.

2.3. Notacije zapisa polustrukturiranih podataka

Pojavom teme polustrukturiranih podataka u disciplinama, koje se bave podacima, bilo je potrebno odrediti notacije za njihovo zapisivanje. Postoje dvije notacije koje se koriste u bazama polustrukturiranih podataka i isto tako se pojavljuju u literaturama, a to su *JavaScript Object Notation,* skraćeno JSON, i *eXtensible Markup Language*, skraćeno XML. U nastavku bit će navedene osnove XML-a te detaljnije obrađen JSON format, zbog njegove dominante upotrebe u cijelom radu.

2.3.1. XML (eXtensible Markup Language)

EXtensible Markup Language notacija je razvijena i prihvaćena od strane W3C-a 1996. godine. XML stvoren je s namjerom stvaranja notacije za razmjenu podataka, koja bi omogućila širok spektar korištenja, bila lako čitljiva i jasna za korištenje. [7] U literaturi još od 1999. godine, XML se spominje kao jedan od prikladnijih jezika za reprezentaciju polustrukturiranih podataka [3], te će se u nastavku zato opisati samo osnove, bitne za shvaćanje početka ideje polustrukturiranih podataka.

XML je sličan HTML (*Hyper Text Markup Language*) jeziku. Bitna razlika ovih dvaju jezika, je u tome što je HTML namijenjen prezentaciji podataka, dok XML ima fokus na sam sadržaj (podatke). Zbog fokusa XML-a na podatke, postao je jedan od glavnih načina za razmjenu podataka putem weba. [3]

2.3.1.1. Osnovna sintaksa

U svojoj suštini, XML neki objekt promatranja prikazuje putem oznaka. Svaka oznaka počinje s < i završava s >. Tekst između < i > zajedno s < i >, se naziva element. Oznake su početne i završne, a između njih je sadržaj. [3]

Za primjer uzimamo oznaku <film>, ovo je početna oznaka, završna oznaka u sebi ima još / između teksta oznake i znaka < tj. </film> je zatvarajuća oznaka. Unutar oznaka, mogu se naći druge oznake ili čisti tekst, na primjer:

```
<film>
      <naziv>Dama i skitnica</naziv>
      <premjera>16.06.1955</premjera>
</film>
```

2.3.1.2. XML u vezi s polustrukturiranim podacima

Ako postoji podatkovni graf zadan uređenom trojkom $G_p = (V_a \cup V_s, B, k)$, gdje je skup V_a skup identiteta atomarnih objekata, a V_s skup identiteta složenih objekata i vrijedi da je $V_a \cap V_s = \emptyset$. Transformacija grafa funkcijom T , koja se naziva translacijska funkcija, dobio bi se XML format tog grafa, ako za funkciju T vrijedi:

$T(v_{a_1}) = v_{a_1}$, $v_{a_1} \in V_a$,

i ako je a oznaka brida b_1 , za koji vrijedi da je $odr(b_1) = v_s$ te $v_s \in V_s$, te ako je v vrh, za koji vrijedi da brid b_2 ima $izv(b_2) = v_s$, a $odr(b_2) = v$ funkcija translacije T je:
$T(v_s)$=<a> $T(v)$,

[3]

2.3.2.JSON (JavaScript Object Notation)

JavaScript Object Nation modernija je verzija jezika za razmjenu podataka te se koristi za prikaz polustrukturiranih podataka. Mnogi sustavi diljem svijeta, prihvatili su JSON kao jezik kojim će razmjenjivati podatke. Prema imenu, dalo bi se zaključiti kako je JSON ovisan o JavaScript programskom jeziku, ali to zapravo nije tako. JSON je neovisan o platformi i sve dok ga ta platforma podržava, on se može na istoj koristiti. [8] Nastao je iz JavaScript literala, što ga čini podskupom JavaScripta, ali sam po sebi nije programski jezik, nego notacija za zapisivanje podataka. [9]

2.3.2.1. Povijest JSON (JavaScript Object Notation)

Douglas Crockfrod spominje se kao „izumitelj“ JSON-a, jer ga je formalizirao i dao JSON-u ime u RFC 4627 specifikaciji. [9], [10]

RFC 4627 specifikacija izdana je 2006. godine te opisuje JSON kao „jednostavan, tekstualan i o jeziku neovisan format“, koji je temeljen na *ECMAScript* programskog standardu. [10]

Organizacija *Internet Engineering Task Force* (IETF) 2014. godine izdala je novu, prepravljenu verziju JSON standarda u RFC 7159. [11] U ovom standardu, daju istu početnu definiciju JSON-a, ali mijenjaju specifikaciju validnog JSON teksta. [9]

Neovisno o tome što je nastao iz JavaScript jezika, obje specifikacije jasno navode kako za JSON vrijede pravila ECMAScripta. ECMAScript Standard definira ECMAScript jezik, koji je nastao od JavaScripta i JScripta, objektno je orijentiran te definira objekte, tipove podataka, ugrađene objekte i operatore među njima. [12] Današnji JavaScript koristi mnoge aspekte ECMAScript standarda.

2.3.2.2. Sintaksa

Podaci se u JSON formatu zapisuju kao parovi ključ-vrijednost ili kao liste vrijednosti. [9] Svakom definiranom ključu, pridružuje se njegova vrijednost, na način da je ključ uvijek između isključivo dvostrukih " navodnika, a vrijednost ovisi o tipu podataka, koji će se razjasniti u nastavku. Separator ključa i vrijednosti je dvotočka : , lijevo od separatora je ime, a desno od separatora vrijednost. Ključ u svom nazivu smije sadržavati slova i brojeve te _ . JSON je objektna notacija, stoga ključ i vrijednost nisu dovoljni za stvaranje objekta. Početak objekta označavamo se s { , kraj s }, a ako se želi definirati polje, to se može s [na početku, i] na kraju polja, dok se elementi odvajaju sa zarezom , . Više parova ključ-vrijednost u objektu, odvaja se isto sa zarezom ,. [8]

Primjer jednostavnog JSON objekta, koristeći gore navedena sintaksna pravila:

{ "ime" : "Borna", "brojevi" : [3, 12, 2] }

2.3.2.3. Tipovi podataka u JSON notaciji

U JSON notaciji postoji šest tipova podataka:

1. object (objekt),
2. string (string),
3. number (broj),
4. boolean (bulovski),
5. null,
6. array (polje).

Objekt je korijenski element JSON-a, to jest objekt je objekt sam po sebi i može sadržavati druge objekte, kao što je navedeno u točki 2.3.2.2., označava ga se s { }.

String je niz Unicode znakova, obgrljen isključivo dvostrukim navodnicima, na primjer "film". String smije sadržavati jednostruke navodnike ' , ali ako se u string želi uključiti " dvostruke navodnike, mora se koristit obrnuta kosa crticu \ , to jest \" u tekstu. Zbog korištenja \ , kao prespojnog znaka (engl. *escape charachter*), navođenje tog znaka u tekstu mora se napraviti s dvije kose crtice unatrag \\. Ostali znakovi, koji mogu biti korišteni u stringu su:

- \/ - kosa crtica,
- \b – pomak unatrag,
- \f – nova stranica,
- \t – tab,
- \n – novi red,
- \r - početak reda,
- \u prefiks s heksadecimalnim vrijednostima u nastavku (na primjer, koristi se za emotikone).

Broj (engl. *number*) je cjelobrojna brojčana vrijednost ili decimalni broj (s točkom .). Negativni brojevi označavaju se s prefiksom znaka minus -. Brojevi podržavaju i znanstvenu e-notaciju.

Primjer upotrebe brojeva u JSON objektu:

```
{
    "broj": 55,
    "novacNaRacunu": 33.23,
    "temperaturaNaSjevernomPolu": -33,
    "triPutaDesetNaSedmu": 3e+7
}
```

Boolean tip podataka je logička vrijednost 0 ili 1, a u JSON-u se označava true za istinu, a false za laž. Primjer korištenja:

```
{ "aktiviran": false }
```

Null tip podataka predstavlja prazninu i piše se null.

```
{ "uloge": null }
```

Polje (engl. *array*), kao što je već navedeno kod opisa sintakse, označava se s [na početku i] na kraju. Elementi polja mogu biti bilo koji od navedenih tipova podataka, ali miješanje tipova podataka u istom polju se ne preporuča. JSON podaci razmjenjuju se između različitih platformi. U slučaju da se JSON objekt s poljem elemenata različitog tipa podataka, pošalje na sistem koji podržava polja u kojem elementi moraju biti istog tipa podataka, doći će do greške kod primatelja. [8]

2.3.2.4. JSON u praksi

Prema riječima autora B. Smitha, „JSON je visoko operabilni format razmjene podataka. To je omogućeno standardizacijom jednostavne gramatike, koja se može prevesti u bilo koji drugi jezik poznavanjem njegove gramatike“. [9] Ova činjenica omogućava upotrebu JSON-a u više područja. JSON se može spremiti na disk računala u obliku datoteke s ekstenzijom .json, pri prijenosu podataka putem mreže postoji application/json MIME tip za raspoznavanje tipova podataka u paketima. U razvoju web aplikacijskih programskih sučelja (takozvan API, engl. *application programming interface*), JSON se koristi kao vraćeni format podataka na zahtjev korisnika. [8] Neki od poznatijih API-a koji koriste JSON su: Google Maps API, Google Search API, Twitter API, Ebay Shopping API i mnogi drugi. [13]

Kada je riječ o bazama podataka, JSON se koristi kao format pohranjivanja podatka u bazama namijenjenim spremanju i upravljanju polustrukturiranim podacima. Neke od baza podataka koje koriste JSON su MongoDB, Cassandra, CouchDB i tako dalje [14].

3. NoSQL (not-only SQL)

Baze podataka jedan su od glavnih dijelova razvoja programskog proizvoda. Najkorištenija vrsta baza podataka su relacijske baze podataka, koje karakterizira čvrsta shema, konzistentnost i dostupnost podataka. Zbog potrebe za manjom strukturiranosti podataka, objašnjene u prethodnim poglavljima, koja je skalabilna na webu i odgovora na problem, limitacija SQL rješenja, došlo je do pojave Not only SQL (NoSQL) pokreta u ranim godinama 21. stoljeća. U to isto doba došlo je do velike ekspanzije web sadržaja i uređaja na webu. [15]

Prof. dr. Stefan Edlich na web stranici NoSQL arhiva definira NoSQL kao „novu generaciju baza podataka koja je ne relacijska, distribuirana, otvorenog koda i horizontalno skalabilna“ [16].

Upravljanje bazama podataka velikih dimenzija zahtjeva skalabilnost, fleksibilnost, dostupnost i organizaciji prihvatljive cijene, a odabir najvažnijeg zahtjeva je na upravitelju baze podataka. [16] Skalabilnost u bazama podataka je potrebna zbog efikasnog načina upravljanja, nagle navale za potražnjom velikom količine podataka ili operacijama nad bazom podataka. Kada se govori o fleksibilnosti, ona predstavlja razinu mogućnosti promjene strukture baze, bez drastičnog utjecaja na sustav koji koristi tu bazu podataka. Gledano s cijenovnog aspekta, cilj je za što prihvatljiviju cijenu dobiti bazu podataka, koja najbolje odgovara sustavu za koji se ona razvija. Gašenje baze podataka, zbog održavanja ili kvara, u većini slučajeva nije opcija, pogotovo kod organizacija koje posluju preko weba, stoga kontinuirana dostupnost baze podataka igra veliku ulogu.

NoSQL baze podataka odgovaraju na ove zahtjeve, na drugačiji način od relacijskih baza podataka. Kada se rješava problem skalabilnosti, lakše rješenje je postaviti nove poslužitelje, što se naziva horizontalno skaliranje. Relacijske baze podataka teže podnose horizontalnu skalabilnost, zbog svoje kompleksnosti i cijene. NoSQL baze podataka razvijane su s ciljem horizontalne skalabilnosti i takav pothvat zahtjeva manje intervencije upravitelja baza podataka. [17]

Kod pristupa modeliranja podataka u relacijskim bazama podataka, počinje se promatranjem domene od značaja za bazu podataka na kojoj se uočavaju potrebni entiteti i veze između njih, takozvano konceptualno modeliranja baze podataka. Nakon toga slijedi logičko modeliranje, u kojem se određuju atributi, ključevi, provede normalizacija i tako dalje. U fazi logičkog modeliranja zapravo se čvrsto odredi izgled modela i načina na koji će podaci biti spremani te kako će se njima pristupati. Navedeni model nije fleksibilan u slučaju promjena tokom razvoja. U slučaju da se bavimo polustrukturiranim podacima, ovaj model predstavljat će probleme.

NoSQL baze podataka daju fleksibilnost modela, koja je u nekim slučajevima potrebna. Sheme nisu čvrsto određene, atributi u entitete se mogu dodavati i micati bez većih utjecaja na ostatak entiteta u bazi, veze nisu toliko kompleksne i sustav za upravljanje bazom podataka ne provjerava integritet podataka na visokoj razini, koja je poznata iz relacijskih baza podataka. [17]

Dostupnost NoSQL baza lakše je regulirati, zbog toga što njihova mogućnost za horizontalnom skalabilnošću rezultira radom baze podataka na više manjih poslužitelja, što je jeftinija opcija od većeg poslužitelja. U slučaju da se dogodi kvar ili se poslužitelj mora održavati, cijela baza neće biti odjednom nedostupna, nego samo onaj mali dio na kojem se kvar dogodio ili je održavanje u tijeku. [17]

NoSQL rješenja su u većini slučajeva otvorenog koda te njihova cijena ne igra ulogu kod odabira baze podataka, ako je došlo do odluke da je noSQL baza podataka pogodna za razvoj, što uvelike rasterećuje onoga tko bira adekvatnu bazu podataka za svoj sustav. Kvalitetna rješenja relacijske baze podataka znaju biti skupa, što ih čini neprihvatljivim manjim organizacijama ili pojedincu. [17]

3.1. ACID i BASE

Kada se govori o svojstvima koja opisuju baze podataka, postoje dva akronima koja predstavljaju dva pristupa osnovnim svojstvima baze podataka. ACID označava atomarnost (engl. *atomicity*), konzistentnost (engl. *consistency*), izolaciju (engl. *isloation*) i trajnost (engl. *durability*). BASE označava načelno dostupno (engl. *basicly available*), varijabilno stanje (engl. *soft state*) i naposljetku dostupno (engl. *eventually consistent*). [17]

3.1.1. ACID

ACID se tradicionalno implementira u relacijskim bazama podataka i njegovo značenje je:

- Atomarnost – nedjeljivost, transakcija je atomarna, ako se ne može izvršiti bez jednog njenog dijela („sve ili ništa“); [17]
- Konzistentnost – transakcija garantira da ako se primjenila na konzistentno stanje baze podataka, podaci nakon završetka transakcije moraju biti u konzistentnom stanju, ali za vrijeme transakcije moguća su inkonzistentna stanja, na primjer, ako se s jednog računa uzme 100 kn, na drugom računu se stanje mora povećati za 100 kn, inače nije održana konzistentnost baze podataka; [17]
- Izolacija – promjene na bazi podataka tokom transakcije, nisu vidljive ostalim korisnicima baze podataka do trenutka njihovog završetka; [17]

- Trajnost – jednom kada je završila transakcija, njene promjene su trajno zapisane, čak i u slučaju gašenja poslužitelja baze podataka (zapisane su na disku ili nekom drugom obliku vanjske memorije). [17]

3.1.2. BASE

BASE skup svojstava smatra se uobičajenim za NoSQL baze podataka i njegova svojstva tumače se na sljedeći način.

- Načelno dostupno – sustav je načelno dostupan u smislu, ako dođe do pada nekog dijela sustava isključivo je taj dio nedostupan, ostalim dijelovima sustava se može pristupiti bez pogreške. Svojstvo načelne dostupnosti moguće je u NoSQL sustavima, jer ako jedan poslužitelj padne, u većini slučajeva postoji drugi spreman s istim podacima, koji će opskrbljivati potražnju za tim podacima. [17]
- Varijabilnog stanja – preko starih podataka baze prepisuju se novi ažurniji podaci. [17]
- Naposljetku konzistentno – zbog rada na više poslužitelja, ako se dogodi promjena na jednom, ista nije vidljiva na ostalima, ali će naposljetku biti vidljiva nakon ažuriranja kopija na ostalim poslužiteljima. Postoji više vrsta naposljetku konzistentnog sadržaja. Spontana konzistentnost osigurava da baza pohranjuje podatke, prema njihovom redoslijedu ažuriranja. Konzistentnost zvana „čitaj svoje upise“, osigurava da svaki put kada zapišemo novu vrijednost preko stare, na sljedećem čitanju dobiju se novo upisani, a ne stari podatci. Konzistencija sesije brine se o „čitaj svoje upise“ konzistentnosti tokom jedne sesije, na primjer jedno spajanje klijenta na poslužitelj smatra se jednom sesijom. Monotona konzistentnost čitanja brine se o situaciji izvršavanja upita, neposredno nakon ažuriranja podataka na jednom od poslužitelja, ako je podataka ažuriran, mora se osigurati da sljedeći upit vrati novi podatak, makar on ne postoji na onom poslužitelju gdje je stigao upit. Kod monotone konzistentnosti pisanja, osigurava se situacija ažuriranja iste vrijednosti istovremeno. U tom scenariju baza podataka garantira da će se ažuriranja obaviti redoslijedom kojim su se dogodila. [17]

3.2. Vrste NoSQL baza podataka

NoSQL baze podataka karakteriziraju se prema načinu na koji pohranjuju podatke. Postoje četiri glavne vrste NoSQL baza podataka:

1. stupčano-orijentirane,
2. dokumentno orijentirane,
3. ključ-vrijednost baze podataka,
4. graf baze podataka.

3.2.1.1. Stupčano-orijentirane baze podataka

U stupčano-orijentiranim bazama podataka, podaci se spremaju u stupce umjesto u redove. Ako se naprimjer, uzme tablica:

Tablica 1 Primjer tabličnog zapisa podataka

Ime	Prezime	Godina studija
Borna	Radinović	3
Pero	Perić	2

U relacijskoj bazi podataka, podaci bi bili zapisani:
Borna,Radinović,3
Pero,Preić,2

U stupčano-rijentiranim bazama, zapis bi izgledao ovako:
Borna, Pero
Radinović, Perić
3,2

Ovakav tip baze podataka koristi se u on-line analitičkom procesiranju, zbog potrebe obrade podataka po stupcima umjesto po redovima. Prednosti ovakve vrste baze podataka su: lako dodavanje novih stupaca, zbog načina njihova internog spremanja, nadalje zbog načina spremanja podataka, lakše je izračunani statističke podatke, kojima je za račun potreban stupac, znači iz memorije se neće uzeti ostali podaci reda, nego samo stupac potreban za izračun, što ubrzava proces računanja.[15]
Primjeri ovakvih baza podataka su: HBase, MapR, Amazon SimpleDB, IBM Informix, Clouddata, KUDU i tako dalje. [16]

3.2.1.2. Ključ-vrijednost baze

Ključ-vrijednost baze podatka, kao što im i samo ime govori, pohranjuju vrijednosti pod određenim ključem. Ključevi su striktno vezani na vrijednost. Autor D. Sullivan je odnos ključa i vrijednosti definirao usporedbom: „ključevi su vezani za vrijednost na isti način na koji je oznaka prtljage vezana za prtljagu u zračnim lukama“, sa čime je htio reći kako se bez ključa ne može doći do vrijednosti. Kod stvaranja ključa se mora voditi računa o vrijednosti, tako sve vrijednosti asocirane s istim entitetom koji se promatra, bi trebale u ključu imati nešto zajedničko, što bi asociralo na njihovu grupu i uz to nešto što ih čini jedinstvenima, jer se inače ne može doći do specifične vrijednosti. Kod generiranja ključevam, koriste se predefinirane vrijednosti (prefiksi i sufiksi) ključa i generirane vrijednosti za specifični dokument. [17]

Na primjer, studenti i profesori na fakultetu i za njih će se spremati imena i prezimena, u ključ-vrijednost bazi, to bi izgledalo ovako:

prof1.ime
prof1.prezime
prof2.ime
prof2.prezime
stud1.ime
stud2.prezime

Iz primjera se vidi da je bitno staviti predefinirani prefiks za studente i profesore, kako bi ih se moglo razlikovati. Vrijednosti u ovoj vrsti baze podataka mogu biti tekstualne vrijednosti, brojevi i BLOB. [17]

Upiti se izvršavaju na temelju ključeva, a ne na temelju vrijednosti, ali je moguće razlikovati tipove podataka, koje sadrže određeni ključevi. U praksi, ključ-vrijednost baze podataka pogodne su za optimizirano pretraživanje pomoću ključeva i služe kao dobra privremena memorija. [15]

Primjeri ovog tipa baza podataka su: DynamoDB, Azure Table Storage, Redis, Aerospike, LevelDB, RocksSB, Scalien, Voldemort, Dynomite itd. [16]

3.2.1.3. Graf baze

Graf baze podataka tip su baza podataka, u kojim su veze između čvorova (objekti promatranja) prikazane grafom. [15] Kao i u teoriji grafova postoje čvorovi i bridovi. Čvorovi su objekti pohranjeni u bazi, a bridovi su veze između tih objekata.

Čvorovi mogu imati mnoge veze prema drugim čvorovima i te veze predstavljaju međusobni odnos čvorova. Ovakve veze su na primjer, međuljudski odnosi, transportni sustav i mrežna topologija. [15] Ova vrsta baze smatra se najspecijaliziranijom od sve četiri vrste NoSQL baza podataka. Svaki čvor ima svoje atribute, koji ga pobliže opisuju i reprezentiraju podatke u njemu. [17]

Dizajnirane su za praćenje međusobne povezanosti između podataka. [17] Neke od takvih baza podataka su: Neo4j i FlockDB. [15]

3.2.1.4. Dokumentno orijentirane

Dokumentno orijentirane baze namijenjene su pohranjivanju, ažuriranju i dohvaćanju polustrukturiranih podataka. Ova vrsta baza podataka radi s dokumentima notacije, poznatim u zapisivanju polustrukturiranih podataka: XML, JSON i BSON. Umjesto tradicionalnih redova, zapis se događa u obliku dokumenta koji ne mora imati sve stupce iste. [15] Dokumenti se odnose na podatke pohranjene u tekstualnom ili binarnom string formatu. [17]

Na primjer, postoje dvije osobe , Pera Perić i Ana Anić. Ana studira, dok Pero ne studira. Kada bi se njihovi podatci pohranjivali u bazu podataka, ne bi bilo potrebno pohraniti one podatke, koji nisu svojstveni za taj objekt promatranja. Tako bi dokumenti u JSON formatu, za pohranu podataka navedene dvije osobe mogli izgledati:

```
{
    "ime": "Ana",
    "prezime": "Anić",
    "godinaStudija": 3
}
```

```
{
    "ime": "Pero",
    "prezime": "Perić"
}
```

Navedena fleksibilnost zapisa te njihova jednostavna čitljivost, pozicionirala je dokumentno orijentirane baze podataka u najkorištenije NoSQL baze podataka. Ovaj tip baza podataka pogodan je za web aplikacije, koje pohranjuju često različit i polustrukturiran sadržaj.[15]

Upiti na dokumentno orijentiranim bazama podataka, izvršavaju se postavljanjem uvjeta na neki atribut dokumenta. Ako neki dokument u kolekciji dokumenata, ima taj atribut i zadovoljava uvjet, bit će vraćen upitom. [17]

Dokumenti mogu sadržavati druge dokumente, što je jedna od prednosti ovih vrsta baza naprema relacijskim, tim pristupom nestaje potreba za JOIN upitima. [17]

Neke od dokumentno orijentiranih baza podataka su: MongoDB, OrientDB, Azure DocumentDB, CouchDB, RethinkDB itd. [16]

Dokumentno orijentirana baza podataka MongoDB, bit će u nastavku detaljnije opisana te će biti objašnjeno kako funkcioniraju dokumentno orijentirane baze na primjeru MongoDB.

4. MongoDB

MongoDB je NoSQL dokumentno orijentirana baza podataka, stvorena s namjerom brzog razvoja web aplikacija i rješenja problema kompliciranog horizontalnog skaliranja baza podataka. [18], [19] Svoje dokumente MongoDB sprema u formatu vrlo bliskom JSON-u [18], što ovu bazu čini bazom za spremanje polustrukturiranih podataka.

4.1. Povijest MongoDB

Sredinom 2007. godine u Sjedinjenim Američkim državama u gradu New Yorku, osnovan je start-up pod nazivom 10gen, kojem je glavni opseg djelatnosti bila „platforma kao servis" (engl. *platform-as-a-service,* PAAS), s namjerom posluživanja web aplikacija i omogućavanjem njihova skaliranja svojim klijentima. 10gen platforma imala je za cilj osloboditi razvojnog programera razmišljanja o tome kako će skalirati cijelu fizičku i programsku arhitekturu svojih aplikacija, htjeli su postići neku vrstu automatizacije skaliranja. Takva ideja skaliranja nije bila najbolje dočekana od razvojnih programera, jer nisu htjeli izgubiti toliku kontrolu nad svojim proizvodom, ali su i dalje bili zainteresirani za aspekt baze podataka 10gen platforme, stoga je rođen projekt razvoja baze podataka, koja će tek kasnije biti nazvana MongoDB. MongoDB je zatim razvijena kao baza podataka otvorenog koda [18] sa sposobnošću horizontalnog skaliranja [19].

4.2. Značajke MongoDB baze podataka

U nastavku, biti će opisane glavne značajke MongoDB baze podataka, koje su bitne za razumijevanje iste.

4.2.1. Koncepti pohrane i tipovi podataka

Dokumenti MongoDB definirani su kao skup ključeva za koje su vezane vrijednosti. Dokumenti su JSON formata, što znači da su ključevi stringovi u dvostrukim navodnicima. Ključevi u imenu ne smiju sadržavati znak \0 (null znak), jer taj znak interno predstavlja kraj ključa, znakovi . i $ rezervirani za operacije nad bazom i trebali bi se samo tako koristiti. Svaki dokument u MongoDB je maksimalne veličine 16 megabajta (MB) i slijedno se pohranjuje na disk. Svaki dokument pri pohrani dobiva određenu podlogu (engl. *padding*), što omogućava proširenje veličine dokumenta sve dok se ne premaši veličina podloge, u tom slučaju dokument se seli poslije zadnjeg zapisa na disku.

Kolekcije su grupe dokumenata [19], koje određuje onaj koji upravlja bazom podataka. Nema ograničenja o tome koji će se dokumenti spremati u kolekciju, to ovisi o osobi koja sprema dokumente u kolekciju. Takav pristup naziva se dinamična shema. Osim grupiranja dokumenata u kolekcije, moguće je kolekcije grupirati u baze podataka. [19]

Kako bi se bolje percipirao ovaj koncept, mnogi autori uspoređuju dokumente i kolekcije s tablicama i redovima. [19] Dokumenti su kao redovi u relacijskim bazama podataka, a kolekcije su tablice koje sadrže te redove, no bitna razlika je ta što dokumenti i kolekcije nisu striktno ograničeni shemom, kao redovi i tablice u relacijskim bazama podataka. Za primjer korištena je tablica studenata od prije, ali je maknut zapis u jednoj ćeliji.

Tablica 2 Tablica student

Ime	Prezime	Godina studija
Borna	Radinović	3
Pero	Perić	null

Kada bi se ova tablica prenijela u format dokumenta i kolekcije, postojala bi kolekcija student, a dokumenti bi izgledao ovako:

```
{
    "ime": "Ana",
    "prezime": "Anić",
    "godinaStudija": 3
}
```

```
{
    "ime": "Pero",
    "prezime": "Perić"
}
```

4.2.1.1. JSON, BSON i tipovi podataka

Vrijednosti u dokumentima mogu biti stringovi, brojčane, boolean, objekti, polja i null. MongoDB koristi verziju JSON formata, zvanu Binarni (engl. *Binary*) JSON (BSON). [19]

BSON je format koji se koristi za spremanja dokumenata na disk. Razlog njegove primjene je lakša implementacija pogoniteljskih programa, koji će na svoj način prevoditi binarni format podataka. JSON, kako je i prije navedeno, je tekstualni format te ga je teže prevesti od binarnih znamenki.

Osim JSON tipova podataka koji su objašnjeni u točki 2.3.2.3. i za koje su navedeni primjeri, MongoDB proširuje tipove podataka dodavanjem datuma, regularnih izraza, objektnih identifikatora i JavaScript funkcija. [19]

Datum se u dokumentu koristi kao i datum u JavaScriptu:

```
{ "sada": new Date() }
```

Regularni izrazi koriste se kod upita:

```
{ "upit": /regularniizraz/i }
```

Identifikatori se pridružuju svakom novostvorenom dokumentu i koriste se kao globalno jedinstveni identifikator objekta u cijeloj bazi. Zauzimaju 12 bajtova na disku i pridružen im je

string od 24 heksadecimalne znamenke. Generira se iz trenutnog vremena (0, 1, 2, 3 bajt), identifikator sustava (4, 5, 6), na kojem je pokrenuta baza podataka (7 i 8), procesnog identifikatora i ostala 3 bajta su inkrementirani broj. [19]

4.2.2. Repliciranje

Repliciranje u MongoDB moguće je s topologijom replicirajućeg skupa (engl. *replica set)*. Zbog potrebe za redundancijom, replicirajući skup umnaža podatke na više poslužitelja, kako bi baza podataka ostala dostupna, nakon mogućeg kvara nekog od poslužitelja. U replicirajućem skupu postoji više poslužitelja, od kojih je jedan primarni. Primarni poslužitelj ima mogućnost pisanja i čitanja, dok ostali poslužitelji u replicirajućem skupu imaju samo mogućnost čitanja. Sekundarni poslužitelj može postati primarni, u slučaju kvara do tada primarnog poslužitelja. [19]

4.2.3. Skaliranje

MongoDB dizajniran je za horizontalno skaliranje u vidu i to postiže distribucijom podataka na dijelove (engl. *sharding*). Ovaj postupak omogućuje automatsko upravljanje podacima distribuiranim u dijelove (engl. *shards*) na više poslužitelja. [19]

Dio (engl. *shard*) baze podataka je podskup podataka koji se distribuira. Distribucija podataka na dijelove izvodi se na velikim kolekcijama, to jest velike kolekcije bivaju podijeljene na više poslužitelja. Jedan dio baze podataka se u većini slučajeva nalazi na jednom poslužitelju. Podaci kolekcije tako se distribuiraju na više poslužitelja ispred kojih se nalazi *mongos*, programski usmjernik koji pohranjuje podatke o tome koji podaci se nalaze na kojem poslužitelju tj. u kojem dijelu baze podataka. Korisnik ne mora voditi računa o raspodjeli podataka tijekom unosa te će sve obavljati mongos. [19] MongoDB još nudi mogućnosti definiranja načina raspodjele podataka putem indeksa i ključeva dijelova, ali u ovom radu neće se ulaziti u detalje distribucijom podataka na dijelove.

4.2.4. Ostale značajke

4.2.4.1. Ad hoc upiti

MongoDB fleksibilan je i po pitanju upita, stoga nudi mogućnost ad hoc upita. Dizajner baze podataka ne mora nužno imati na umu upite, koji će biti izvršavani nad bazom, kada je dizajnira.

4.2.4.2. Indeksiranje

Indeksi čuvaju raspored podataka u bazi i putem njih je izvršavanje upita brže. Praktični su kod velikih kolekcija, jer ako postoji indeks nema potrebe za potpunim pretraživanjem cijele kolekcije. Indeksi su strukture podataka, koji pohranjuju male dijelove kolekcije, koji su lakši za pretraživanje od cijele kolekcije. MongoDB nudi indekse za sortiranja, tekstualne indekse i tako dalje.
[20]

4.2.4.3. Agregiranje podataka

Agregacije podataka se koriste za njihovu analizu na različite načine. Korištenjem značajke agregacijskih cjevovoda, u MongoDB-u omogućena je grupacija, projekcija, sortiranje, limitiranje, preskakanje i filtriranje dokumenata kolekcije nad kojima se izvršava agregacija. [19]

4.2.4.4. Brzina i trajnost

MongoDB nudi nekoliko opcija, koje se odnose na trajnost baze. MongoDB sustav je moguće konfigurirati tako da se jedan zapis garantira unesenim, tek nakon što je zapisan na svaku repliku baze podataka. Postoji opcija *fire and forget,* kod koje se kod upisa podataka miče inače potrebna potvrda zapisa. Ova je opcija pogodna kod zapisa podataka malog opsega, na primjer klikovi na oglasu. Konfiguracija zvana dnevnik (engl. *jurnaling*), zapisuje podatke u dnevnik svakih 100 ms, što u slučaju kvara i gašenja poslužitelja garanitra da svi upisi u bazu, koji se nisu uspjeli do kraja izvršiti, budu izvršeni nakon ponovnog pokretanja poslužitelja. [18]

4.2.4.5. Pohrana datoteka

Pohrana datoteka (binarnih dokumenata) u MongoDB-u, vrši se preko *GridFS* mehanizma. [19] *GridFS* nije zapravo značajka MongoDB-a, ali se nalazi u svim driverima za MongoDB. Radi na principu pohrane dokumenata u komade od 255KB. [18] Preporuča se za pohranjivanje velikih binarnih datoteka, koje se neće često mijenjati, te usporava performanse poslužitelja. [19]

4.3. Rad s MongoDB-om

Rad s MongoDB bazom omogućen je putem sučelja JavaScript komandne linije, pod nazivom Mongo naredbeni redak (engl. *mongo Shell*). Mongo naredbeni redak može izvoditi JavaScript kod i automatski se spaja na bazu podataka na računalu te nudi mogućnost spajanja na udaljenu bazu podataka. Putem Mongo naredbenog retka, administrator baze

podataka i svatko tko je ovlašten, može kreirati, ažurirati, pregledavati i brisati baze podataka, kolekcije i dokumente. [19]

4.3.1. Osnovne naredbe

Neke od osnovnih naredbi za rad s MongoDB bazom podataka su:

`db`
Ispisuje ime baze podataka u kojoj se trenutno nalazimo.

`use imeBaze`
Prebacuje korisnika u bazu podataka za koju je naveo ime.

`mongo --username <korisnickoime> --password <lozinka> --host <domenabaze> --port <brojporta>`
U komadnoj liniji operacijskog sustava pokretanjem ove naredbe, spaja se na udaljeni poslužitelj na kojem je pokrenut MongoDB. (Napomena: kod pokretanja ove naredbe na Windows računalu, potrebno je pozicionirati se u direktorij gdje je instaliran Mongo naredbeni redak te umjesto mongo na početku unjeti `./mongo`)

4.3.2. Kreiranje i brisanje baze podataka

Baze podataka spremaju se u razdvojene datoteke na disku, te ih korisnik može kreirati koliko želi po potrebi, uz definiranje prava pristupa za iste. Baze podataka u imenu ne smiju imati razmak, imena moraju biti sačinjena od UTF-8 znakova te su osjetljiva na velika i mala slova. Korištenje znakova /, \, ., ", *, <, >, :, |, ?, $ i \0 u imenu je zabranjeno. Sustav za upravljanje podataka MongoDB kod instalacije, kreira tri baze podataka koje su potrebne za rad sustava te su njihova imena rezervirana samo za te svrhe. Baze podataka koje se kreiraju su: admin, local i config. Admin baza podataka korijenska je baza podataka sustava i sadrži autentifikacijske podatke. Local baza podataka služi pohrani kolekcija, koje su jedinstvene za poslužitelj na kojemu su pohranjene, a config baza podataka čuva podatke o distribuciji baze podataka na dijelove, ako ona postoji. Naredba za kreiranje baze podataka je ista, kao i ona za prebacivanje iz jedne baze podataka u drugu bazu podataka:

```
use imeBaze
```

Brisanje baze podataka radi se tako da se prvo naredbom `use` pozicionira u bazu podataka koju se želi obrisati, te se izvrši:

```
db.dropDatabase()
```

[19]

4.3.3. Kreiranje dokumenta

Dokumenti se kreiraju umetanjem JSON objekta, koji se želi spremiti kao dokument, naredbom insert na način:

```
db.imekolekcije.insert(<JSON objekt>)
```

Objekt se može navesti direktno u naredbi kao `{ "imeatributa": <vrijednost> }` ili se može prvo spremiti u JavaScript varijablu u Mongo naredbenom redku, te se zatim predati kao referenca u naredbu `insert`. U slučaju da se želi kreirati više dokumenata, naredba insert se zamjeni naredbom `batchInsert()` i kao argument joj se preda polje JSON objekata, koje se želi unijeti u bazu. [19]

4.3.4. Kreiranje kolekcije

Kreiranje kolekcije radi se implicitno kod kreiranja dokumenata i eksplicitno naredbom `createCollection()`. Implicitno kreiranje kolekcije događa se kada se u naredbi `db.imekolekcije.insert(<JSON objekt>)` navede ime kolekcije, koja ne postoji, te se zatim stvori ta kolekcija s dokumentom, koji je stavljen u argument funkcije `insert`.

Naredba `db.createCollection(<imekolekcije>,<konfiguracijskiJSONobjekt>)`, koristi se kada se želi kreirati kolekcija s određenim opcijama. Prvi argument funkcije je ime kolekcije, a drugi je JSON objekt s konfiguracijskim parametrima. Pregled svih parametara konfiguracije kolekcije, dostupan je u MongoDB dokumentaciji. [21]

4.3.5. Ažuriranje dokumenata

Ažuriranje dokumenta omogućeno je naredbom `.update()`, ova naredba prima 2 argumenta `.update(<uvjet>,<JSONobjekt>)`. Drugim argumentom postavlja se uvjet za dokument koji je potrebno ažurirati, svaki dokument koji zadovolji uvjet biti će ažuriran s JSON objektom, koji se predaje u drugom argumentu. Uvjet se postavlja u obliku JSON objekta, koji sadrži atribute, koje bi trebao sadržavati dokument koji želimo ažurirati (više o uvjetima nad MongoDB bit će riječi u točki 4.3.7 Upiti). Ako se preda običan JSON objekt, u drugom argumentu funkcije taj objekt zamijenit će postojeći dokument, koji zadovoljava uvjet u prvom argumentu. U slučaju kada više dokumenata zadovolji uvjet i preda se samo JSON objekt, u drugom argumentu ažuriranje će biti izvedeno samo na prvom nađenom dokumentu.

Operator `.update()` ima treći opcionalni argument. Treći argument je JSON objekt s postavkama ažuriranja. Ako se u atribut objekta, navedenog u trećem argumentu, postavi atribut upsert na `true`, funkcija `update` stvorit će novi dokument, ako niti jedan ne zadovoljava uvjet prvog argumenta.

Potpuna zamjena dokumenta novim dokumentom nije uvijek poželjna, stoga se mogu koristiti modifikatori (engl. *modifiers*) u MongoDB, koji će ažurirati samo dio dokumenta. Modifikatori se navode kao atributi objekta u drugom parametru funkcije `.update()`. [19]

Neki od najkorištenijih modifikatora su:

- $set,
- $inc,
- $dec,
- $push,
- $pull.

Modifikatori se koriste tako da se u JSON objekt drugog argumenta funkcije, modifikator navede kao atribut te se za njegovu vrijednost postavlja drugi (ugrađeni) JSON objekt, koji za svoje atribute ima atribute koje se žele modificirati u objektu, i za vrijednosti tih atributa, vrijednosti za koje se želi da ih modifikator koristi kod modifikacije.

`$set`

Koristi kod promjene vrijednosti željenog atributa na način:

```
db.imeKolekcije.update(<uvjet>,
{ $set: {  "atributObjektaKojiSeAzurira": <željenaVrijednost> } })
```

U slučaju nepostojanja atributa, u modifikatoru set on se stvara.

`$inc`

Inkrementira brojčanu vrijednost atributa. [19]

```
db.imeKolekcije.update(<uvjet>,
{ $inc: {  "atributKojiSeInkrementira": <inkrement> } })
```

`$dec`

Dekrementira brojčanu vrijednost atributa. [19]

```
db.imeKolekcije.update(<uvjet>,
{ $dec: {  "atributKojiSeDekrementira": <denkrement> } })
```

`$push`

Dodaje element u ugrađeno polje dokumenta. [19]

```
db.imeKolekcije.update(<uvjet>,
{ $push: {  "poljeDokumenta": <vrijednostZaPolje> } })
```

`$push`

Uklanja element polja, koji zadovoljava navedeni uvjet. [19]

```
db.imeKolekcije.update(<uvjetObjekta>,
{ $pull: {  "poljeDokumenta": <uvjetPolja> } })
```

4.3.6. Brisanje dokumenata

Dokument se briše naredbom `.remove()` na način:

```
db.imeKolekcije.remove(<uvjet>)
```

Naredba će obrisati sve dokumente, koji zadovoljavaju uvjet naveden u argumentu naredbe.

4.3.7. Upiti

Upiti u MongoDB izvršavaju se metodom `.find()`, koja kao argument prima uvjetni dokument, a koji je do sada u tekstu bio zvan samo uvjet. Uvjetni dokument je JSON objekt, koji kao argumente ima uvjete upita. Postoje dvije vrste uvjeta upita, uvjeti jednakosti i uvjetni operatori.

4.3.7.1. Uvjeti jednakosti

Uvjet jednakosti je JSON objekt s jednim atributom koji ima pridruženu vrijednost. On predstavlja atribut, koji se traži u dokumentu, koji za vrijednost ima onu vrijednost navedenu u uvjetu jednakosti. Nizanje više takvih atributa, ekvivalentno je logičkom i (engl. *and*) uvjetu nad dokumentom. Navedeno je lakše predočiti na konkretnom primjeru. Recimo da se kolekcija zove studenti i sadrži dokumente u kojima svaki sadrži podatke jednog studenta. Ako se želi pretražiti studente s imenom Pero, uvjet jednakosti bi izgledao:

```
db.studenti.find({ "ime": "Pero" })
```

Naveden upit, vratit će sve studente s imenom Pero. Ako se upit izvede s još jedni atributom, na primjer `db.studenti.find({ "ime": "Pero", "ime": "Perić" })`, kao rezultat bit će vraćeni svi studenti s imenom Pero i s prezimenom Perić. [22]

4.3.7.2. Uvjetni operatori

Prilikom rada su podacima pretraživanje po jednakosti nije dovoljno, već i po nekim općenitijim uvjetima. Za pretraživanja, koja su kompleksnija od obične jednakosti, MongoDB koristi uvjetne operatore.

Uvjetni operatori usporedbe su `$lt`, `$lte`, `$gt` i `$gte`. Značenja operatora redom su manje, manje ili jednako, veće i veće ili jednako. Sva četiri operatora koriste se na isti način. Upit koji bi na primjer tražio studente s ocjenom većom ili jedakom 3 izgledao bi:

```
db.studenti.find({ "ocijena": { $gte: 3 }})
```

Uvjetni operatori `$in` i `$or`, koristili bi se za upite za koje je potrebno logičko ili. Operator `$in` traži vrijednosti, koje pripadaju nekom zadanom skupu (skup se zadaje kao JSON polje), njegova negacija je operator `$nin`, što znači da će upit vratiti sve koji nisu u njemu zadanom skupu.

Traži li se učenik s ocjenama 1 ili 2 upit bi bio sljedeći:

```
db.studenti.find({ "ocijena": { $in: [1,2] }})
```

Ekvivaletno prethodnom upitu, samo s operatorom `$or`, upit bi bio:

```
db.studenti.find({"$or": [{"ocijena": 1},{"ocijena": 2} ]  })
```

Operator `$or`, kao vrijednost ima polje uvjeta, između kojih bi bio logički operator ili. Uvjetni operator `$not` je logička negacija. Sve studente koji nemaju ocjenu jedan pronašao bi sljedeći upit:

```
db.studenti.find({ "ocijena": { $not: 1 }})
```

4.3.7.3. Regularni izrazi

Pri izvršavanju upita nad tekstualnim atributima dokumenata, moguće je koristiti regularne izraze. Regularni izrazi se koriste na način:

```
db.imeKolekcije.find( { "tekstualniAtribut": /<regularniIzraz>/ } )
```

4.3.7.4. Upiti nad ugrađenim poljima

Kao i JSON objekti, MongoDB dokumenti podržavaju ugrađena polja. Moguće je raditi upite nad elementima tih polja. Za ovakvu vrstu upita postoje prigodni operatori.

Ugrađena polja mogu se pretraživati običnim uvjetima jednakosti. Uvjeti jednakosti nad argumentom koji sadrži polje, vratit će vrijednosti navedene kao vrijednost uvjeta.

Operator `$all`, uvjetni je operator koji se koristi za pretraživanje ugrađenog polja prema zadanim vrijednostima, koje su podskup tog polja. Ako se želi naći sve studente, koji u polju kolegiji imaju kolegije Baze podataka i Web programiranje, upit nad tim polje bio bi:

```
db.studenti.find( { "kolegiji": { $all: [ "Baze podataka", "Web programiranje" ] } } )
```

Postoji operator `$slice`, s kojim se može vratiti samo neke dijelove polja, pri vraćanju rezultata nekog upita, na primjer:

```
db.studenti.find( { "ime": "Pero"}, { "Kolegiji": { $slice: [0,2] } } )
```

upit bi vratio dokumente koji u sebi imaju ime Pero, ali polje kolegiji bi u sebi sadržavalo samo kolegije od nultog do drugog indeksa.

Budući da polja mogu sadržavati druge ugrađene dokumente, nad njima se mogu vršiti upiti isto kao i na samostalnim dokumentima. Poljima ugrađenih dokumenata pristupa se točkom. Recimo da student ima još jedan atribut kontakti, u kojem su dokumenti koji sadrže sve kontakt podatke studenta. Studenta čiji kontakt podaci sadrže email pero@foi.hr, našli bi na način:

```
db.studenti.find( { "kontakt.email": "pero@foi.hr} )
```

4.3.8. Agregacije

Agregacije podataka u MongoDB sustavu su transformacije toka podataka kroz agregacijske operatore, a izvode se pomoću agregacijskog okvira. Tok podataka je skup dokumenata kolekcije. Na neki način se može reći da ono što su složeni upiti u relacijskim bazama podataka, su agregacije u MongoDB sustavu.

Agregacijski operatori su:

- $match,
- $project,
- $group,
- $sort,
- $limit,
- $skip.

$match služi filtriranju dokumenata u manji podskup dokumenata, koji odgovara određenom upitu koji se zadaje. `$project` služi projekciji atributa dokumenata i omogućava njihovu manipulaciju (na primjer preimenovanje). `$group` grupira elemente prema zadanom atributu dokumenta, i obavlja operaciju nad tom grupom poput na primjer sume. Operator $sort sortira dokumente u agregacijskom cjevovodu. `$limit` se koristi za definiranje vrijednosti vraćenih dokumenata, a operator $skip za preskakanje određenog broja dokumenata.

```
Najam.aggregate([
    { $match:  { datumNajma: { $gt: prviDanMjeseca } } },
    { $group: { _id : "$film", broj: { $sum: 1 } } },
    { $sort: {  count: -1 } },
    { $limit: 5}
  ])
```

Kod iznad, primjer je agregacije nad kolekcijom dokumenata najma, koja traži prvih pet najposuđivanijih filmova u kolekciji najmova za trenutni mjesec. U prvom koraku ove agregacije, izdvojit će se dokumenti kolekcije kojima je datum najma veći od prvog dana u mjesecu (prviDanMjeseca je JavaScript varijabla u kojoj se nalazi vrijednost prvog dana trenutnog mjeseca), nakon `$match` operatora, `$group` operator grupirat će najmove prema identifikatoru filma koji je iznajmljen i zbrojiti koliko je dokumenata s tim identifikatorom, te će rezultat biti dostupan u idućem koraku agregacije pod atributom naziva broj. Operator `$sort` sortirat će dokumente silazno prema atributu broj, i na kraju će operator `$limit` uzeti prvih pet dokumenata zadnje kolekcije.

4.3.9. Definiranje tekstualnog indeksa

Pretraživanje tekstualnih vrijednosti u dokumentima kolekcije, omogućeno je definiranjem tekstualnih indeksa. Indeks se definira na tekstualnim atributom polja, te se kasnije može koristiti u upitu za pretraživanje kolekcije za neku tekstualnu vrijednost. Indeks ovog tipa kreira se prema dolje navedenoj naredbi:

```
db.filmovi.createIndex( { naslov: "text" } )
```

Naredba iznad, kreirat će tekstualni indeks nad atributom naslov svih dokumenata kolekcije filmovi. Ukoliko se sada želi pretražiti sve dokumente kolekcije filmovi, za tekstualnu vrijednost u naslovu to bi se učinilo s operatorom $text, upitom navedenim ispod:

```
db.filmovi.find({ $text: { $search: 'pad' } })
```

Upit iznad vratio bi dokumente koji u naslovu sadrže tekst „pad“.

5. Implementacija on-line videoteke

Rad s polustrukturiranim podacima, to jest rad sa sustavom za upravljanje polustrukturiranim podacima MongoDB, nakon teoretske razrade sadržaja, najbolje je potkrijepiti implementacijom aplikacije, koja bi simulirala primjenu MongoDB tehnologije u praksi.

Praktičnim dijelom ovog rada, bit će prikazan razvoj web aplikacije on-line videoteke. Videoteka je neformalno nazvana Moviestacks, kao što će biti vidljivo u logu aplikacije u gornjem desnom kutu kasnije, kod opisa klijentske strane aplikacije. *Moviestacks* je aplikacija za posuđivanje filmova na koju će se korisnici moći samostalno registrirati. Svojim korisnicima nudit će pretraživanje filmova prema zadanim filterima. Korisnici će moći pregledavati sve relevantne podatke za filmove poput glumaca, uloga i njihovih redatelja, te nakon što korisnici odaberu željeni film, moći će ga iznajmiti na određeni period vremena. Svaki korisnik imat će svoj inventar filmova, iz kojega će se filmovi maknuti, nakon što je najam za njih istekao. Drugi i složeniji dio aplikacije je administratorsko sučelje aplikacije. Administrator će kreirati filmove, žanrove, glumce i redatelje filmova. Svaki od kreiranih podataka, administrator će moći pregledavati, ažurirati i brisati. Administrator će imati uvid u korisnike svog sustava, moći će ih kreirati i brisati po potrebi, bit će mu omogućen pregled svih najmova korisnika i moći će upravljati njima micanjem ili stvaranjem najmova.

5.1. Metode i tehnike rada

U izradi ovog rada najviše su korišteni programski alat za uređivanje izvornog koda te alati za upravljanje bazom podataka.

Baza podataka u ovom radu je dokumentno orijentirana baza podataka MongoDB, kojom se upravlja putem Mongo naredbenog retka ili putem grafičkog sučelja MongoDB Compass.

Programski dio ovog rada implementiran je u tehnologijama MEAN stoga, tj. Mongo, Express, Angular i Node.js.

Za uređivanje izvornog programskog koda korišten je alat Visual Studio Code. Visual Studio Code alat je softverske tvrtke Microsoft, koji uz svoju osnovnu funkcionalnost uređivanja izvornog koda, nudi integriranu kontrolu git repozitorija te vrlo praktičnu mogućnost integriranog Microsoft PowerShell terminala, koji je potreban za kompajliranje JavaScript i TypeScript koda.

Pri instalaciji vanjskih paketa potrebnih za implementaciju programskog dijela rada, korišten je Node Package Manager. Korišteni paketi programskog koda u implementaciji on-line videoteke su:

Angular Material: https://www.npmjs.com/package/@angular/material
Angular2-Toaster: https://www.npmjs.com/package/angular2-toaster
Bcrypt: https://www.npmjs.com/package/bcrypt
Jsonwebtoken: https://www.npmjs.com/package/jsonwebtoken
Moment: https://www.npmjs.com/package/moment
Mongoose: https://www.npmjs.com/package/mongoose
Multer: https://www.npmjs.com/package/multer
Nodemailer: https://www.npmjs.com/package/nodemailer

Svi podaci uneseni u bazu podataka aplikacije, programski su uneseni u bazu podataka putem besplatnog MovieDB API-ja, za koji je bilo bitno u samoj aplikaciji prikazati logo MovieDB organizacije, zbog poštivanja uvjeta korištenja. Logo MovieDB vidljiv je na početnoj stranici aplikacije. Dokumentacija MovieDB API:

https://www.themoviedb.org/documentation/api

Dijelovi programskog koda i neki od korištenih programskih koncepata u projektu implementacije on-line videoteke, preuzeti su iz tečaja Angular i NodeJS – Vodič za MEAN stog (engl. *Angular & NodeJS - The MEAN Stack Guide*), dostupnog na Udemy web stranici za učenje i podučavanje. [23]

5.2. Opis korištenih tehnologija

MEAN stog je besplatno rješenje otvorenog koda za razvoj web aplikacija u JavaScript programskom jeziku. MEAN stog je full-stack rješenje, u kojem se za bazu podataka aplikacije koristi MongoDB, Node.js kao poslužiteljsko rješenje, Express kao JavaScript razvojni okvir namijenjen razvoju Node.js poslužiteljske strane aplikacije i Angular front-end razvojni okvir za razvoj korisničkih sučelja.

Razvoj MEAN-a stoga bi bio nemoguć, bez izuma Google V8 Enginea. V8 je JavaScript engine otvorenog je koda i visokih performansi, koji JavaScript kod pretvara u strojni jezik te omogućuje ulazno-izlazne operacije. Pojava V8, maknula je JavaScript iz web preglednika i omogućila razvoj Node.js tehnologije. [24]

5.2.1. Node.js

Node.js je asinkroni poslužiteljski sustav upravljan događajima, baziran na V8 platformi. Specifičan je zbog svog neblokirajućeg ulazno-izlaznog sustava, što je omogućila priroda JavaScript koda. Svako ulazno-izlazna operacija ima callback. Callback je funkcija koja će se izvršiti nakon završetka ulazno izlazne operacije, to znači da se ne mora čekati da se ulazno izlazna operacija završi kako bi se ostatak koda mogao izvršavati, jer će se o završetku izvršavanja takve operacije pobrinuti callback funkcije. [24]

5.2.2. Express

Express.js je razvojni okvir za Node.js, koji omogućuje jednostavniji razvoj poslužiteljske strane aplikacije i pojednostavljuje upravljanje statičnim datotekama na poslužitelju. Glavni koncepti Expressa su ruteri, middleware i rute. Express omogućuje obradu zahtjeva na serveru, tako da za definirane rute izvršava zadani kod. Rute se definiraju tokom razvoja aplikacije i dio su URL-a za zahtjeve na poslužitelj. Zahtjevi na poslužitelj stižu na njegove rute, a svaka ruta ima svoju callback funkciju, koju izvršava u slučaju pristiglog zahtjeva. Expressom se može upravljati svim HTTP zahtjevima (GET, POST, PUT ,PATCH, HEAD, DELETE, CONNECT, OPTIONS, TRACE). Middleware rute je dio koda, koji se izvršava svaki puta kada dođe zahtjev na tu rutu, isti se middleware može koristiti na više različitih ruta. Primjer korištenja ovih koncepata bit će opisan pri opisu implementacije aplikacije. [24]

5.2.3. Angular

Angular je prije bio poznat kao AngularJS i koristio je druge koncepte od današnje distribucije. Angular koji će se koristiti u ovoj implementaciji, na službenoj dokumentaciji navodi se kao Angular, a kolokvijalno je poznat kao Angular 2. AngularJS koristio je MVC arhitekturu i bio je pisan pomoću čistog JavaScripta, dok Angular koristi TypeScript i komponentnu arhitekturu zajedno s predlošcima. TypeScript je superskup JavaScripta, što znači da proširuje JavaScript određenim funkcionalnostima, na primjer tipovi podataka, klase, sučelja i tako dalje. Svrha Angulara je razvoj brzih korisničkih sučelja, koja ne zahtijevaju ponovno učitavanje stranice, već radi asinkrono u pozadini putem AJAX-a (engl. *Asynchronous JavaScript And XML*). [24]

5.3. Zahtjevi aplikacije

U nastavku zahtjevi aplikacije bit će opisani po principu naziva zahtjeva (tekst bold stila), ispod kojeg se nalazi opis zahtjeva. Za svaki zahtjev navedeni su koraci scenarija tog

zahtjeva te izmimke. Iznimke predstavljaju scenarij, koji se ne bih trebao dogoditi, ali u slučaju njegove pojave, aplikacija mora upravljati njime, tako da ne dođe do greške pri izvršavanju. Način upravljana iznimkom naveden je kao rješenje.

Registracija

Svaki korisnik mora se prvo registrirati kao korisnik aplikacije sa svojim osobnim podacima, kako bi pristupio ostatku web aplikacije (dijelovi za pregled, pretraživanje i rezervaciju filma).
Scenarij

1. Otvaranje forme za registraciju.
2. Unos podataka potrebnih za registraciju (ime, prezime, korisničko ime, email, lozinka i datum rođenja, potvrda izjave o privatnosti i uvjeta korištenja).
3. Unos lozinke dva puta.
4. Slanje aktivacijske email poruke na email adresu korisnika.
5. Preusmjeravanje korisnika na početnu stranicu nakon uspješne registracije.

Iznimke

- Korisnik je unio lozinku koja ne zadovoljava traženi format.
  Rješenje: Ne dopušta se registracija, dok format nije zadovoljen.
- Korisnik nije ponovio unos lozinke identično na polju za unos i potvrdu lozinke.
  Rješenje: Ne dopušta se registracija, dok unesene lozinke nisu identične.
- Korisnik nije unio validan email.
- Rješenje: Ne dopušta se registracija, dok email nije validno unesen.

Prijava

Korisnik unosi svoje korisničke podatke za omogućavanje pristupa ostatku aplikacije (dijelovi za pregled, pretraživanje i rezervaciju filma).
Scenarij

1. Korisnik unosi svoj registrirani email i lozinku.
2. Korisnik stišće gumb prijava.
3. Poslužitelj provjerava korisnikove podatke.
4. U slučaju validnih korisničkih podataka, stvara autentikacijski token i šalje ga se korisniku.
5. Korisniku se omogućava pristup pretraživanju, pregledu i najmu filmova.

Iznimke

- Korisnik nije aktivirao svoj korisnički račun.
  Rješenje: Korisnika se obavještava o grešci i ne dopušta se prijava.
- Korisnik je unio krivu lozinku.
  Rješenje: Korisnika se obavještava o grešci i traži se ponovni unos lozinke.
- Korisnik je unio neregistrirani email.
  Rješenje: Korisnika se obavještava o grešci i na formi se nudi poveznica za registraciju.
- Korisnik je zaboravio lozinku.
  Rješenje: Korisniku se nudi opcija „Zaboravljene lozinke“, u kojoj unosi svoj email te mu zatim na email stiže nova generirana lozinka.

Pretraživanje filmova

Pretraživanje filmova prema tekstualnom unosu, filterima i sortiranje rezultata pretrage.
Scenarij
1. Korisnik unosi tekstualni pojam za pretraživanje, bira filtere pretrage (žanr, godina) i vrstu sortiranja rezultata pretrage (datum premijere filma ili broj posudbi na razini svih korisnika aplikacije).
2. Korisniku se prikazuju rezultati pretrage.

Iznimke
- Ne postoje rezultat za zadanu pretragu.
  Rješenje: Korisniku se prikazuje prazno područje, gdje su inače rezultati pretrage.

Pregled podataka filma

Klikom korisnika na film u rezultatima pretrage ili film naveden kod uloga glumaca na zaslonu glumca, otvara se zaslon s podacima filma (podaci filma, uloge filma i pripadajući glumci, redatelji)
Scenarij
1. Korisnik klikće na film u rezultatima pretrage ili film naveden kod uloga glumca na zaslonu glumca.
2. Korisnika se prebacuje na stranicu filma.
3. Prikazuju se podaci filma.
4. Prikazuju se uloge u filmu i njima pripadajući glumci.
5. Prikazuju se redatelji filma.

Iznimke
- Ne postoje neki od podataka.
  Rješenje: Aplikacija mora nastaviti rad bez greške, ali na tim mjestima mora biti bez sadržaja.

Pregled podataka glumca

Klikom na glumca navedenog u ulogama filma, korisnika se preusmjerava na stranicu glumca na kojoj su prikazani podaci glumca, uloge glumca zajedno s filmovima kojima te uloge pripadaju i prikazuju se filmovi u kojima je taj glumac bio redatelj.
Scenarij
1. Korisnik klikće na ime glumca navedenom u ulogama filma (vidi scenarij Pregled podataka filma).
2. Korisnika se preusmjerava na stranicu glumca.
3. Prikazuju se podaci glumca.
4. Prikazuju se uloge glumca, zajedno s filmovima kojima te uloge pripadaju.
5. Prikazuju se filmovi u kojima je taj glumac bio redatelj.

Iznimke
- Ne postoje neki od podataka.
  Rješenje: Aplikacija mora nastaviti rad bez greške, ali na tim mjestima mora biti bez sadržaja.

Posuđivanje filmova

Korisnik može posuditi svaki film, koji mu je dostupan u rezultatima pretrage. Posuđivanje filma znači odabir film i trajanje posudbe, nakon koje se film stavlja u inventar posuđenih filmova korisnika za vrijeme trajanja posudbe.
Scenarij
1. Klik na gumb posudbe na rezultatima pretrage ili na stranici filma.
2. Odabir trajanja posudbe.

3. Potvrđivanje posudbe.
4. Otvaranje posudbe za korisnika za traženi period.
5. Preusmjeravanja na stranicu pregleda posuđenih filmova.

Iznimke

- Film već postoji u inventaru korisnikovih posuđenih filmova.
  Rješenje: Obavijestiti korisnika o aktivnom najmu filma, zatim ponuditi produljenje posudbe ili završavanje radnje.

Pregled posuđenih filmova

Korisnik može u svakom trenutku pogledati svoj inventar posuđenih filmova.
Scenarij

1. Klik na poveznicu Moji Filmovi.
2. Prikaz posuđenih filmova.

Iznimke

- Posudba je istekla.
  Rješenje: Istaknuti film kao istekao i onemogućiti ga.

Pregled podataka profila.
Korisnik može pregledati svoje podatke u aplikaciji klikom na poveznicu Moj Profil.
Scenarij

1. Klik na Moj Profil.
2. Prikazivanje podataka profila.
3. Prikaz opcija promjene lozinke i promjene podataka profila.

Iznimke

- Nema.

Promjena podataka profila

Korisnik može promjeniti svoje podatke profila (ime, prezime, datum rođenja i korisničko ime).
Scenarij

1. Klik na gumb Promjeni Podatke Profila.
2. Otvaranje forme za promjenu podataka profila.
3. Unos novih podataka profila.
4. Spremanje unosa.

Iznimke

- Nema

Promjena lozinke

Korisnik ima mogućnost promjene lozinke za pristup svom profilu.
Scenarij

1. Klik na gumb Promjeni lozinku.
2. Otvaranje forme za promjenu lozinke.
3. Unos trenutne lozinke.
4. Unos nove lozinke i potvrde nove lozinke.

Iznimke

- Pogrešna trenutna lozinka.
  Rješenje: Korisnika se obavještava o pogrešci i nova lozinka se ne sprema.

- Lozinka i potvrda lozinke se ne podudaraj.
  Rješenje: Sve dok se lozinka i njena potvrda ne podudaraju, onemogućava se spremanje nove lozinke.
- Lozinka ne odgovara traženom formatu lozinke.
  Rješenje: Sve dok lozinka ne odgovara traženom formatu, onemogućava se spremanje nove lozinke.

Kreiraj Film

Mogućnost Administratora da kreira film unoseći podatke filma, uloge filma s glumcima koji su glumili u tim ulogama i redatelje filma. Kod svakog kreiranje filma, administrator šalje sliku filma (plakat filma) na poslužitelj.

Scenarij

1. Otvori formu za kreiranje filma.
2. Unesi podatke filma.
3. Spremanje podatke filma.
4. Omogućavanje unos uloga i redatelja za film.
5. Unos redatelja i uloga filma.
6. Završavanje radnje.
7. Preusmjeravanje na stranicu kreiranog filma.

Iznimke

- Neko od polja podataka filma nije uneseno (uključujući odabir slike).
  Rješenje: Onemogući spremanje filma, dok sva polja nisu popunjena.

Kreiraj Glumca

Mogućnost Administratora da kreira glumca s njegovim opisnim podacima. uz podatke filma šalje se i slika glumca na poslužitelj.

Scenarij

1. Otvaranje forme kreiranja glumca.
2. Unos podataka gluma.
3. Spremanje unosa.
4. Preusmjeravanje da stranicu novostvorenog glumca.

Iznimke

- Neko od polja nije popunjeno.
  Rješenje: Obajvesti korisnika o grešci.

Kreiraj Žanr

Mogućnost Administratora da kreira žanr filma.

Scenarij

1. Otvaranje forme za unos žanra.
2. Unos naziva žanra.
3. Spremanje žanra.

Iznimke

- Polje imena žanra nije uneseno.
  Rješenje: Zanemariti unos. Obavijestiti korisnika o grešci.

Kreiraj Najam

Mogućnost Administratora da stvori najam filma za korisnika, uz proizvoljno trajanje i datum najma.

Scenarij

1. Otvori formu unosa novog najma.

2. Odabir filma najma.
3. Odabir korisnika najma.
4. Unos ostalih podataka najma.
5. Spremanje najma.

Iznimke
- Neko od polja nije uneseno.
  Rješenje: Zanemari unos. Obavijestiti korisnika o grešci.

Kreiraj Korisnika

Mogućnost Administratora da bez postupka registracije kreira novog korisnika sustava.
Scenarij
1. Otvaranje forme za kreiranje korisnika.
2. Unos podataka korisnika.
3. Postavljanje uloge korisnika.
4. Spremanje korisnika.

Iznimke
- Korisničko ime već postoji.
  Rješenje: Obavijestiti korisnika o grešci i ne unijeti korisnika.

Pregled Korisnika

Mogućnost Administratora da pregledava sve korisnike unesene u sustav.
Scenarij
1. Otvaranje pregleda korisnike.
2. Straničenje naprijed, nazad, zadnja i prva stranica.
3. Prikazivanje gumba za ažuriranje i brisanje korisnika.

Iznimke
- Nema.

Pregled Najmova

Mogućnost Administratora da pregledava sve najmove korisnika.
Scenarij
1. Otvaranje pregleda korisnike.
2. Straničenje naprijed, nazad, zadnja i prva stranica.
3. Prikazivanje gumba za ažuriranje i brisanje korisnika.

Iznimke
- Nema.

Pregled Najma Prema Korisniku

Mogućnost Administratora da odabirom korisnika, pregledava sve najmove tog korisnika.

Scenarij:
1. Odabir korisnika u filteru najmova.
2. Prikaz najmova korisnika.

Iznimke
- Nema.

Pregled Žanrova

Mogućnost Administratora da pregledava sve žanrove unesene u sustav.
Scenarij

1. Otvaranje pregleda korisnike.
2. Straničenje naprijed, nazad, zadnja i prva stranica.
3. Prikazivanje gumba za ažuriranje i brisanje korisnik.

Iznimke

- Nema.

Pregled Glumca

Mogućnost Administratora da pregledava sve glumce unesene u sustav.
Scenarij

1. Otvaranje pregleda korisnike.
2. Straničenje naprijed, nazad, zadnja i prva stranica.
3. Prikazivanje gumba za ažuriranje i brisanje korisnika.

Iznimke

- Nema.

Ažuriraj Film

Mogućnost Administratora da ažurira podatke filma, uloge filma i redatelje filma.
Scenarij

1. Klik na gumb ažuriraj u opciji pregleda filmova.
2. Otvaranje forme za ažuriranje, koja je popunjena trenutnim podacima filma.
3. Ažuriranje podataka.

Iznimke

- Neko od polja podataka filma je prazno.
  Rješenje: Onemogućavanje ažuriranja filma, dok sva polja nisu popunjena.

Ažuriraj Glumca

Mogućnost Administratora da ažurira podatke i sliku glumca.
Scenarij

1. Klik na gumb ažuriraj u opciji pregleda glumaca.
2. Otvaranje forme za ažuriranje koja je popunjena trenutnim podacima glumca.
3. Ažuriranje podataka.
4. Spremanje podataka.

Iznimke

- Neko od polja je prazno.
  Rješenje: Onemogućavanje ažuriranja glumca, dok sva polja nisu popunjena.

Ažuriraj Žanr

Mogućnost Administratora da ažurira ime žanra.
Scenarij

1. Klik na ažuriraj kod pregleda žanrova.
2. Otvaranje forme s popunjenom vrijednostima žanra.
3. Spremanje ažuriranja.

Iznimke

4. Polje imena žanra je prazno.
   Rješenje: Onemogućavanje ažuriranja žanra, dok se polje ne popuni ili mogućnost izlaza.

Ažuriraj Najam

Mogućnost Administratora da ažurira podatke najama.
Scenarij

1. Klik na ažuriraj kod pregleda najmova.
2. Otvaranje forme s popunjenom vrijednostima najma.
3. Spremanje ažuriranja.

Iznimke

- Nema.

Ažuriraj Korisnika

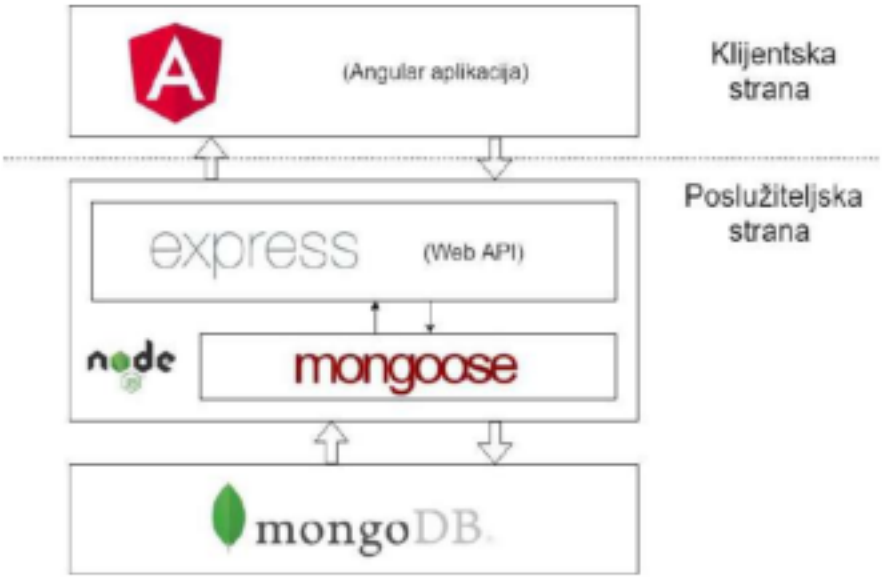
Mogućnost Administratora da ažurira podatke korisnika i postavi ga kao jedno od administratora sustava.
Brisanje Žanra, Najma, Korisnika, Glumca, Filma.
Mogućnost Administratora da obriše bilo koji od elemenata žanrova, najmova, korisnika, glumaca i filmova.
Scenarij

1. Na pregled elemenata prikazati gumb za brisane.
2. Kliknuti gumb za brisanje.
3. Obrisati element iz sustava.

Iznimke

- Nema.

5.4. Arhitektura aplikacije



Slika 4 Arhitektura aplikacije (Koristeći logotipove: [25]–[29])

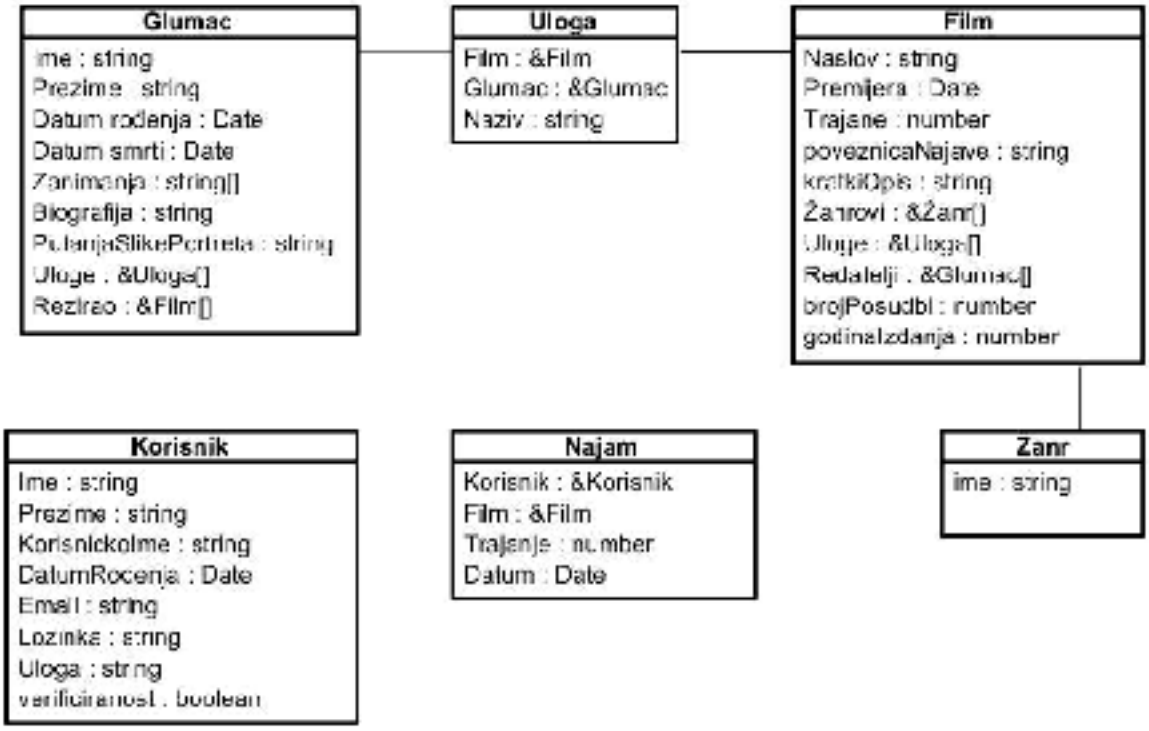
[25]–[29]

Implementacija web aplikacije on-line videoteke sastoji se od tri arhitekturalna sloja: Klijentske aplikacije kao prezentacijskog sloja, Web aplikacijskog programskog sučelja (Web API) kao logičkog sloja s pristupom podacima i MongoDB kao podatkovnog sloja. Navedeni slojevi su prikazani na slici 4.

Web API programsko je sučelje za komunikaciju klijentske aplikacije i baze podataka MongoDB te izvodi poslovnu logiku aplikacije. Implementiran je razvojnim okvirom Express i pokrenut na Node.js poslužitelju. Komunicira s bazom podataka MongoDB putem mongoose objektno dokumentno mapirajućeg (engl. *Object-Document mapper*, ODM) alata. Mongoose omogućuje kreiranje sheme objekata koji će se koristiti u aplikaciji za čitanje, pohranjivanje, ažuriranje i brisanje dokumenata u MongoDB bazu podataka.

Klijentska aplikacija implementirana je s Angular razvojnim okvirom i radi kao jednostranična aplikacija (engl. *Single Page Application*, SPA), što znači da se ekran ne osvježava, već se dinamički mijenja sadržaj zaslona putem Angular tehnologije. Klijentska aplikacija sve podatke koje treba pročitati, ažurirati, kreirati ili obrisati iz baze podataka, to čini putem Web API-a, tako da HTTP (engl. *Hyper Text Transfer Protocola*) mrežnim protokolom zatraži ili pošalje sadržaj na Web API, nakon čega slijedi odgovarajući odgovor u JSON formatu, ovisno o tome je li zatražila podatke ili ih je poslala za unos u bazu podataka.

5.5. Model dokumenata u aplikaciji



Slika 5 Dokumenti aplikacije on-line videoteke

Mongoose radi s bazom pomoću shema dokumenata i preko njih komunicira s bazom podataka, stoga je potrebno definirati te sheme. Sheme se definiraju pomoću Shema objekta iz mongoose paketa. Shema objekt u konstruktoru prima kao parametar JavaScript objekt, čiji atributi opisuju model sheme koja se definira.

Slika 5 prikazuje definirane modele dokumenata koji su korišteni u aplikaciji. Reference na objekte su njihovi identifikatori objekta u MongoDB sustavu i označeni su prefiksom &. Film sadrži naslov filma, datum njegove premjere, trajanje, poveznicu na najavu filma, kratki opis radnje filma, broj posudbi filma od strane korisnika aplikacija (sumarna vrijednost koja se inkrementira pri svakoj posudbi tog filma), godinu izdanja te polja referenci na dokumente žanrova filma, uloga u filmu i redatelje filma. Glumci filma su shema koji služi za spremanje glumaca, koji su glumili u ulogama filma i osoba koje su režirale film, režiseri nemaju poseban dokument, zbog toga što dijele sve iste atribute kao i glumac filma, a i moguće je da redatelj glumi u filmu i obratno. Svaki dokument glumca u sebi ima reference na uloge koje je imao (u atributu Uloge) u filmu te reference na filmove koje je režirao (u atributu Rezirao). Uloga je dokument koji sadrži podatke uloge glumca na nekom filmu, na način da u sebi ima referencu na glumca koji glumi u filmu, film u kojem postoji ta uloga i ime uloge. Žanr je dokument za pohranjivanje žanrova filma. Dokument korisnik sadrži sve podatke potrebne za registraciju u

aplikaciju, atribut Verificiranost ukazuje na to je li korisnik aktivirao svoj račun, a nakon aktivacije račun služi administratoru za blokadu računa. Najam je dokument koji se stvara svaki puta kada korisnik iznajmi određeni film, te sadržava sve podatke potrebne za praćenje najma.

Način kreiranja ovih Shema u JavaScript-u istog je koncepta za sve modele. Ispod je kod kreiranja sheme za dokument korisnika.

```
const mongoose = require('mongoose');
const uniqueValidator = require('mongoose-unique-validator');
const userSchema = mongoose.Schema({
  firstName: {type: String, required: true},
  lastName: {type: String, required: true},
  username: {type: String, required: true, unique: true},
  dateOfBirth: {type: Date, required: true},
  email: {type: String, required: true, unique: true},
  password: { type: String, required: true},
  role: { type: String, required: true, default: 'user'},
  verified: { type: Boolean, required: true, default: false}
});
userSchema.plugin(uniqueValidator);
module.exports = mongoose.model('User', userSchema); [30]
```

Mongoose nudi opciju validacije podataka na strani poslužitelja korištenjem validatora. Iznad je uporabljen validator za jedinstvenost unosa `mongoose-unique-validator`, koji će pri svakom unosu provjeriti postoji li već dokument s tom vrijednosti u bazi podataka.

Prvim kreiranje dokumenta nekog modela, mongoose će kreirati kolekciju za taj model s pripadnim imenom u množini, zatim će svi dokumenti tog modela biti spremani u tu kolekciju. U aplikaciji stoga postoji 6 kolekcija: Korisnici, Najmovi, Glumci, Žanrovi, Filmovi i Uloge.

5.6. Web API

Web API se Express razvojnim okvirom implementira definiranjem ruta te upravljača (engl. *controllers*) za njih. Aplikacija Web API-a koristit će definirane rute i na temelju njih preusmjeriti zahtjev u pripadni dio logike na poslužitelju. U ovoj aplikaciji definirano je šest grupa ruta (za svaku kolekciju u bazi podataka jedna grupa), svaka grupa ruta koristi se za dohvaćanje, kreiranje, ažuriranje i brisanje dokumenata te kolekcije.

Primjer kreiranje jedne rute u Express razvojnom okviru, uz pretpostavku da se pokrenuo HTTP server u Node.js, s Express aplikacijom kao callback funkcijom.

```
const express = require('express');
const userRoutes= require('./routes/users');
const app = express();
app.use('/api/users', userRoutes); [30]
```

Gore navedeni kod, kreirao bi rutu za sve HTTP zahtjeve na putanju /http/users i preusmjerio ih na rute korisnika definirane u userRoutes. Jedna ruta u userRoutes je na primjer:

```
router.post('/login', UserController.loginUser); [30]
```

Gornja linija kreirala je rutu za HTTP POST zahtjev na putanju '/api/users/login'. Drugi argument funkcije je upravljač (engl. *controller*) koji će dalje manipulirati zahtjevom i na kraju vratiti odgovor klijentu. `UserController.loginUser` je referenca na JavaScript funkciju definiranu u modulu uvezenom u `UserController` varijablu, koja u sebi ima logiku obrade zahtjeva usmjerenog na tu rutu.

5.6.1. Kreiranje dokumenata

Kreiranje dokumenata u aplikaciji izvodi se slanjem HTTP POST zahtjeva na odgovarajuću rutu Web API-ja. POST zahtjev će u tijelu sadržavati JSON objekt, koji sadrži podatke dokumenta, koji se treba kreirati ili će biti FormData formata, ako se uz podatke dokumenta na poslužitelj šalje i slika vezana za dokumenta (u slučaju kreiranja filma ili glumca).

Nakon što je poslužitelj zaprimio zahtjev, spremit će ga koristeći moongoose funkcije. Potrebno je kreirati novu instancu objekta tipa sheme dokumenta, koji je potrebno spremiti i zatim nad tim objektom pozvati funkciju save, te nakon uspješnog spremanja vratiti korisniku odgovor. Kao primjer kreiranje dokumenta, ispod se navodi kod za kreiranje novog žanra.

```
const genre = new Genre({
    name: req.body.name
  });
  genre.save().then(
    (createdGenre) => {
      res.status(201).json({
        message: 'Genre added successfully'
      })
    }
  ); [30]
```

`req.body.name` sadrži vrijednost atributa u tijelu zahtjeva pod atributom naziva name.

5.6.2. Dohvaćanje dokumenata

Dokumenti se dohvaćaju HTTP GET zahtjevom na Web API rutu. Ukoliko je riječ o dohvaćanju samo jednog dokumenta, ruta će u sebi sadržavati proizvoljni parametar, koji će upravljač te rute dohvatiti i prema tome vratiti odgovarajući dokument.

Kao primjer ispod, navodi se dohvaćanje najmova iz baze sa mongoose paketom za korisnika.

```
Rent.find({ user: req.userData.userId }).populate({ path: 'movie' }).then(
    (rentedMovies) => {
      res.json({
        message: 'Your rents fetched succesfully!',
        mymovies: rentedMovies
      });
    }
  ).catch(
    error => {
      res.status(500).json({
        message: 'Fatching your movies failed!',
     });
    }
  ); [30]
```

Funkcija .find() radi na istom principu, kao i mongoDB funkcija find() opisana u 5.3.7. Upiti. Funkcija populate() jedna je od funkcionalnosti mongoose paketa, radi na način da na temelju reference na kolekciju definirane u svojstvima atributa movie nalazi pripadni dokument u bazi podataka i vraća ga u rezultatu upita, kao ugrađeni JSON dokument pod atributom movie.

Parametar rute, mora se definirati kod same definicije rute. Ruta s parametrom, koja sadrži parametar id, definirala bi se navođenjem "api/movies/:id" kao imena rute (prefiks: označava parametar). U implementaciji bi se zatim taj parametar dohvatio iz zahtjeva. Kod dohvaćanja jednog dokumenta, koristi se funkcija .findById(). Tako bi dohvaćanje filma pod određenim identifikatorom, izgledalo kao što je navedeno ispod:

```
Movie.findById(req.params.id).populate({ path: 'genres' }).then( movie => {
    if(movie){
      res.status(200).json(movie);
    } else {
      res.status(404).json({message: 'Movie not found'});
    }
  }).catch( error =>{
    res.status(500).json({
      message: 'Fatching Movie failed!',
    });   }) [30]
```

5.6.3. Ažuriranje dokumenata

Ažuriranje dokumenata moguće je HTTP PATCH i PUT zahtjevima. Ukoliko se ažurira cijeli dokument, koristiti će se PUT zahtjev, a ukoliko se ažurira samo jedan od atributa dokumenta, koristit će se PATCH zahtjev.

Kod PATCH tipa zahtjeva, dokument u bazi ažurirao bi se mongoose funkcijom findOneAndUpdate(), koja bi ažurirala isključivo one atribute, koji su navedeni u objektu njenog drugog argumenta.

```
const patchData = {
      firstName: userData.firstName,
      lastName: userData.lastName,
      username: userData.username,
      dateOfBirth: userData.dateOfBirth
  }
  User.findOneAndUpdate( { _id: userId }, patchData ).then(
    (updatedUser) => {
      if(updatedUser){
        res.status(201).json({
          message: 'User info data updated',
          info: updatedUser
        });
      } else {
        res.status(500).json({
          message: 'User indo update data failed'
        });
      }
    }
  ); [30]
```

Iznad navedeni kod, ažurira atribute dokumenta Korisnik naveden u objektu, na koji pokazuje varijabla patchData pod zadanim identifikatorom

Kod zahtjeva tipa PUT koristi se mongoose funkcija .updateOne(), koja radi na principu istoimene MongoDB funkcije (updateOne radi kao i update, ali u slučaju više nađenih dokumenata pod zadanim uvjetom, ažurira samo prvi nađeni).

```
const genreId = req.body.id;
  genre = new Genre({
    _id: genreId,
    name: req.body.name
  });

Genre.updateOne({ _id: genreId }, genre).then((result) => {
    console.log(result);
    if(result.n > 0) {
      res.status(200).json({
        message: 'Genre updated successfully',
      });
    } else {
      res.status(401).json({
        error: {message: 'Not authorized!'},
      });
    }
  }); [30]
```

Gore navedeni kod, primjer je ažuriranja cijelog dokumenta žanr. Funkcija ažurira dokument sa zadanim identifikatorom s novim objektom, na kojeg pokazuje genre varijabla.

5.6.4. Brisanje dokumenata

Brisanje dokumenata radi se HTTP DELETE zahtjevom. Ruta namijenjena slanju DELETE zahtjeva, sadržavat će kao parametar identifikator objekta, koji je potrebno obrisati. Identifikator objekta, koji se treba obrisati, definira se kao parametar svake rute, koja upravlja DELETE zahtjevima.

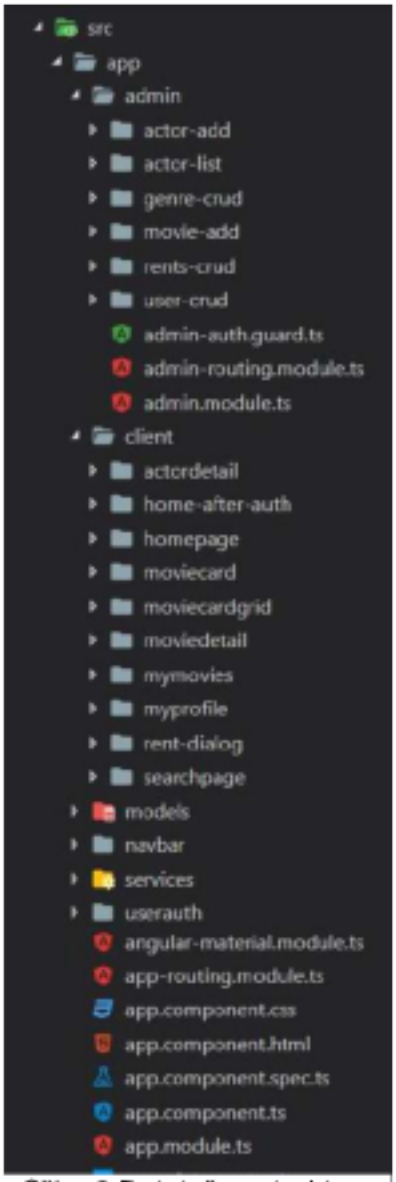
```
Actor.deleteOne({ _id: req.params.id }).then( result => {
    if(result.n > 0) {
      res.status(200).json({
        message: 'Actor deleted successfully',
     });
    } else {
      res.status(401).json({
        error: {message: 'Not authorized!'}
     });
    }}); [30]
```

Kod iznad, primjer je brisanja dokumenta funkcijom deleteOne iz mongoose paketa, iste je funkcionalnosti kao i istoimena funkcija MongoDB sustava.

5.7. Klijentska aplikacija

Klijentska strana aplikacije on-line videoteke je jednostranična aplikacija, implementirana pomoću Angular razvojnog okvira. Angular nudi razvoj aplikacije, koja sama u pregledniku klijenta renderira zaslon bez potrebe za osvježavanjem stranica i dobavljanjem novog statičnog sadržaja s poslužitelja.

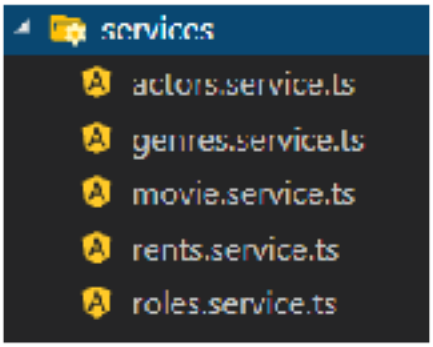
Korisničko sučelje Angular aplikacije implementira se korištenjem modula i komponenti. Komponente su gradivne jedinice aplikacije, one predstavljaju jedan element korisničkog sučelja. Svaka komponenta sadrži klasu komponente, u kojoj se nalazi poslovna logika toga dijela korisničkog sučelja zajedno s HTML (engl. *HyperText Markup Language*) predlošcima, koji definiraju način renderiranja komponente i CSS stilskim datotekama, koje se isključivo odnose na HTML te komponente.



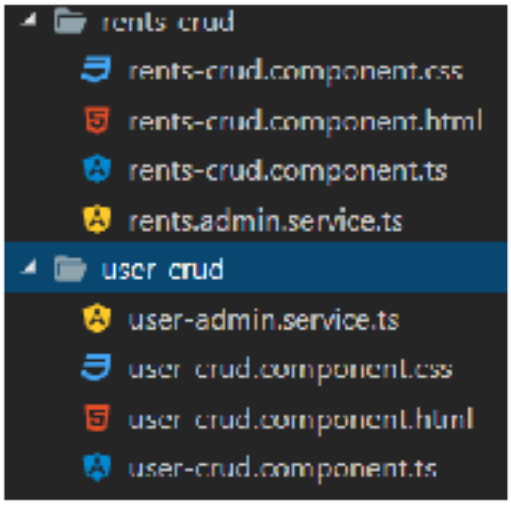
Slika 6 Datotečna struktura projekta klijentske aplikacija

Kao što je prikazano na slici 6, razvijena aplikacija sastoji se od tri modula: glavni aplikacijski modul, admin modul i Angular Material modul. Glavni aplikacijski modul pokreće cijelu aplikaciju i sadrži sve komponente iz *client* direktorija, koje su stvorene za dio aplikacije koji je vidljiv iz pogleda običnog korisnika. Korisničko sučelje kreirano je pomoću paketa Angular Material [31]. Angular Material paket je razvojni okvir za kreiranje modernih i interaktivnih korisničkih sučelja. Sve komponente aplikacija implementirane su pomoću komponenti Anglar Material paketa, stoga u projektu postoji modul zvan Angular Material modul, koji sadrži sve uvoze korištenih Angular Material komponenti sučelja. Treći modul je Admin modul, koji u sebi implementira sve komponente vidljive iz pogleda administratora sustava. Admin modul u preglednik korisnika učitava se na zahtjev (engl. *lazy loading*), što znači da je konfiguriran tako da se klijentu cijela logika tog modula i njegov vizualni dio učitava u preglednik, tek nakon prvog zahtjeva za korištenjem tog dijela aplikacije.

Osim komponenti i modula, Angular razvojni okvir nudi implementaciju servisa. Servisi služe dohvaćanju i slanju podataka na Web API pomoću HTTP klijenta, dostupnog u HttpClient komponenti Angular okvira. Fokus prikaza podataka je na modulima i njegovim komponentama, dok servisi rade s podacima i propagiraju ih kroz sve kompoente koje ih koriste.



Slika 8 Angular servisi glavnog aplikacijskog modula



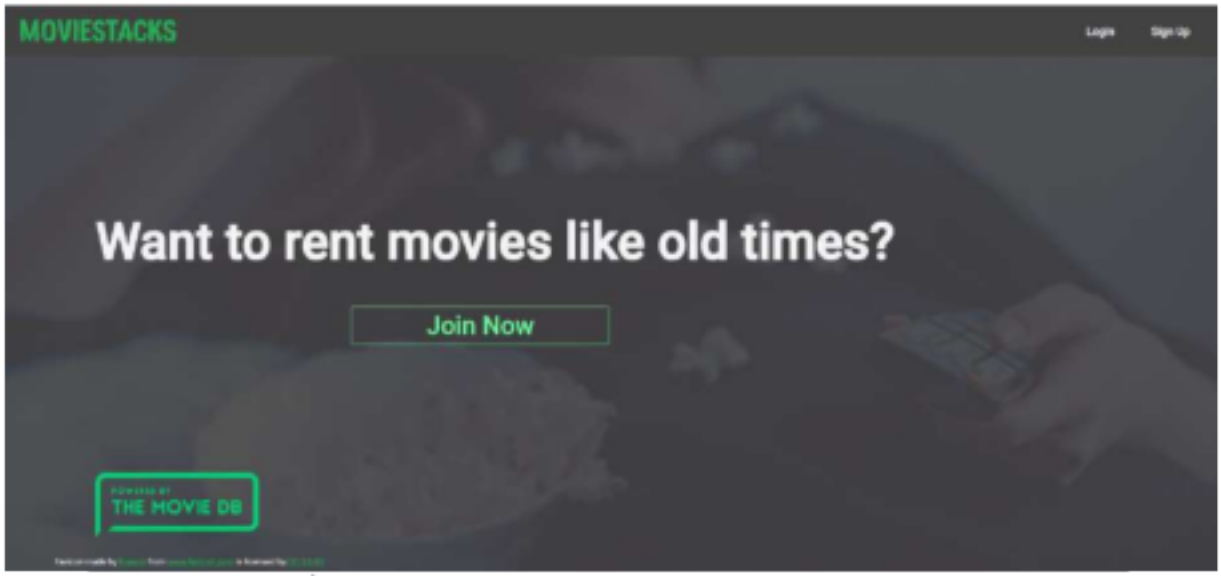
Slika 7 Angular servisi administratorskih modula

Glavni aplikacijski modul koristi podatkovne servise, za koje svaki odgovara jednoj grupi ruta Web API sloja, što znači da postoje podatkovni servisi za glumce, žanrove, filmove, najmove i uloge (Slika 8). Svako dohvaćanje, ažuriranje, brisanje i kreiranje dokumenata baze podataka preko Web API servisa, izvodi se koristeći navedene podatkovne servise.

Administratorska strana aplikacije u sebi sadrži dva dodatna podatkovna servisa vidljiva na slici 7, jedan za operacije na dokumentima najmova filmova, a drugi za operacije nad dokumentima korisnika, ta dva podatkovna servisa implementirana su odvojeno od ostalih, zbog svoje specifičnosti koja je vezana isključivo za administratorski dio aplikacije. Administrator je jedini koji može upravljati najmovima filmova proizvoljno, te imati pristup korisnicima sustava i operacija nad tim dokumentima. Osim specifičnosti uporabe, činjenica je da ovi servisi sadrže programski kod ažuriranja i brisanja korisničkih podataka, koji je zbog njihove odvojenosti od ostalih servisa i direktno implementacijom u admin modul, nedostupan običnim korisnicima što nudi određenu razinu sigurnosti. Kao što je prije navedeno, pogledi i funkcionalnost administratorskog dijela aplikacije, preuzimaju se u preglednik samo na zahtjev korisnika koji je administrator.

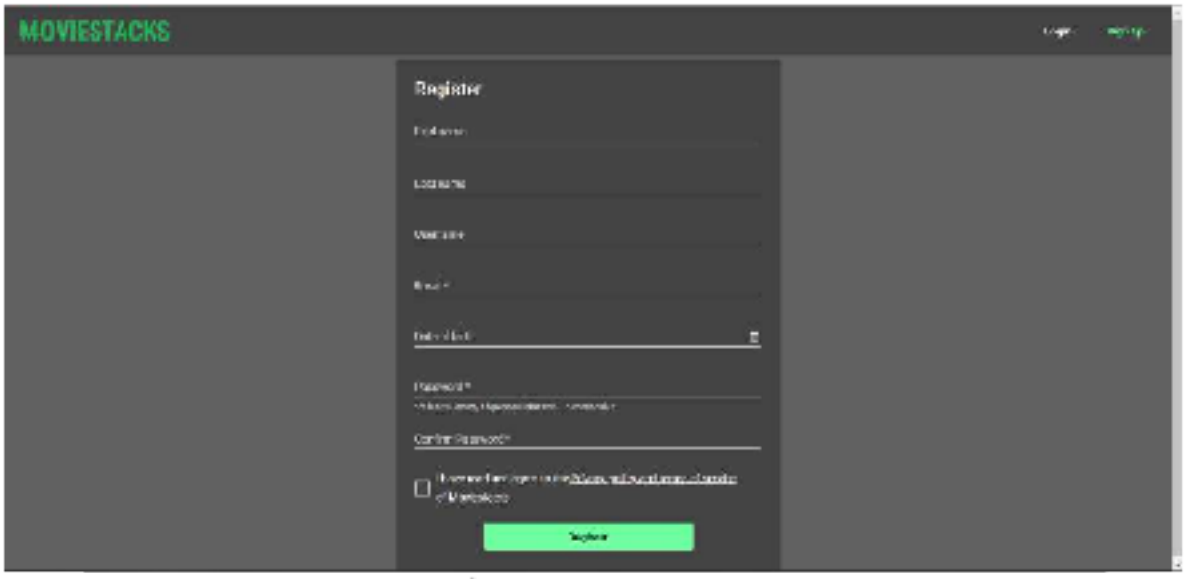
U nastavku će biti objašnjeno kompletno korisničko sučelje klijentske aplikacije iz perspektive običnog korisnika i zatim iz perspektive administratora.

5.7.1.Pogled Korisnika



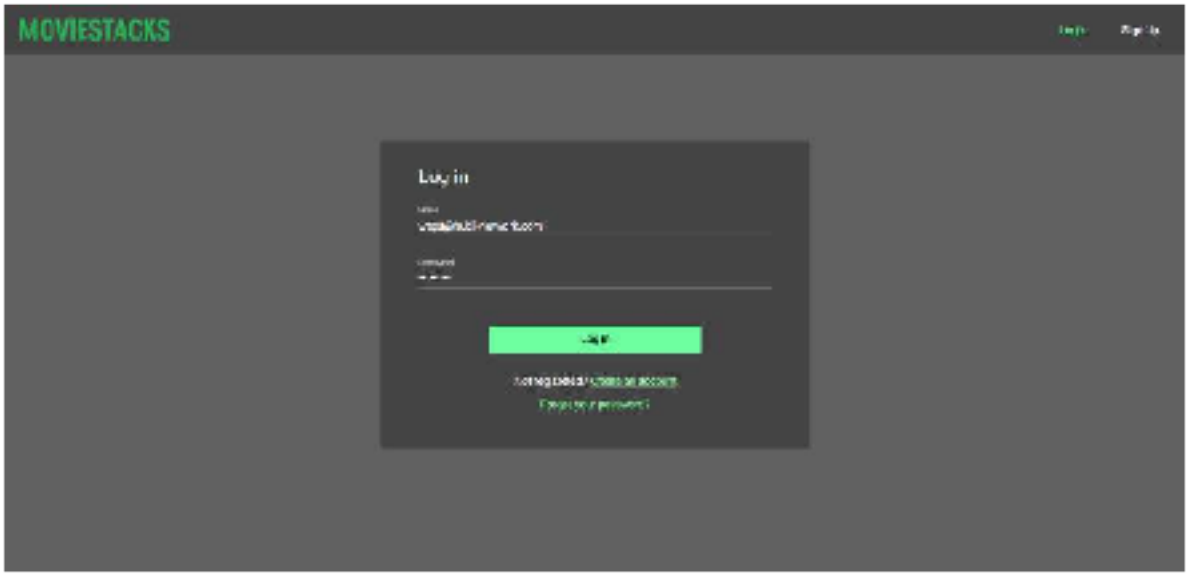
Slika 9 Početna stranica (neregistirani korisnik)

Ulaskom u aplikaciju, neregistrirani korisnik vidi početnu stranicu aplikacije te mu se na vrhu ekrana prikazuje alatna traka koja nudi opciju prijave u aplikaciju (gumb Login) i opcija registracije (gumb Sign Up). Osim gumba za registraciju, moguće je kliknuti gumb Join Now, koji će korisnika preusmjeriti na stranicu za registraciju.



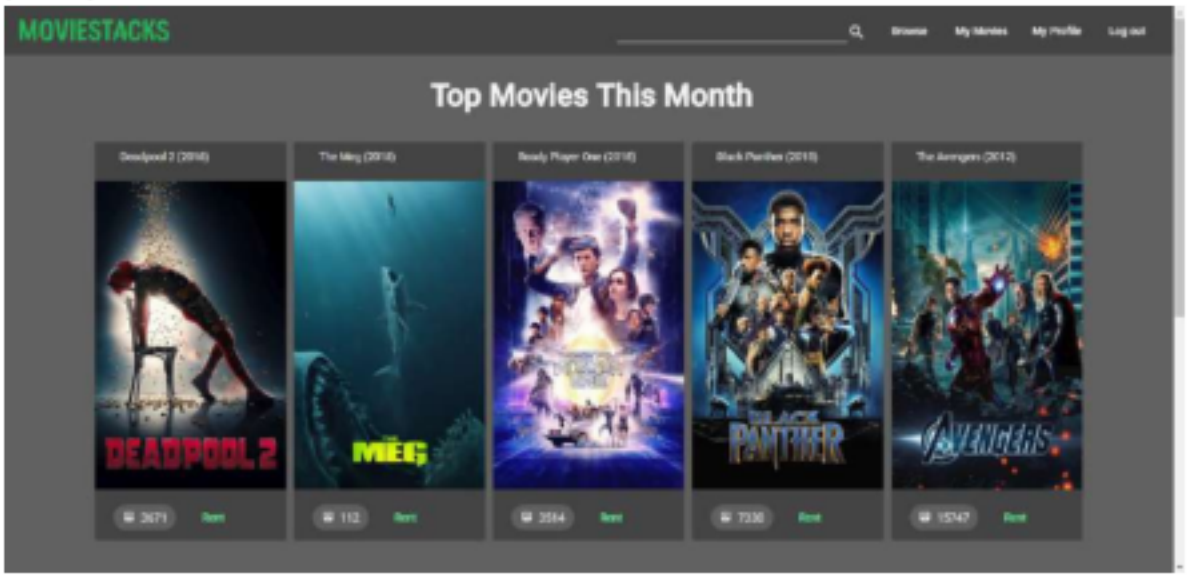
Slika 10 Registracija

Klikom na gumb Sign Up ili Join Now, korisnika se preusmjerava na stranicu registracije prikazanu slikom 10. Korisnik se u sustav registrira popunjavajući svoje osobne podatke (ime, prezime, datum rođenja), navodeći željeno korisničko ime te definirajući lozinku za pristup korisničkom računu. Potvrđivanjem uvjeta privatnosti i korištenja na dnu forme te klikom na gumb Register, korisnika se registrira u sustav. Nakon uspješne registracije, korisniku stiže email s aktivacijskom poveznicom, koju je potrebno stisnuti prije prve prijave u aplikaciju.



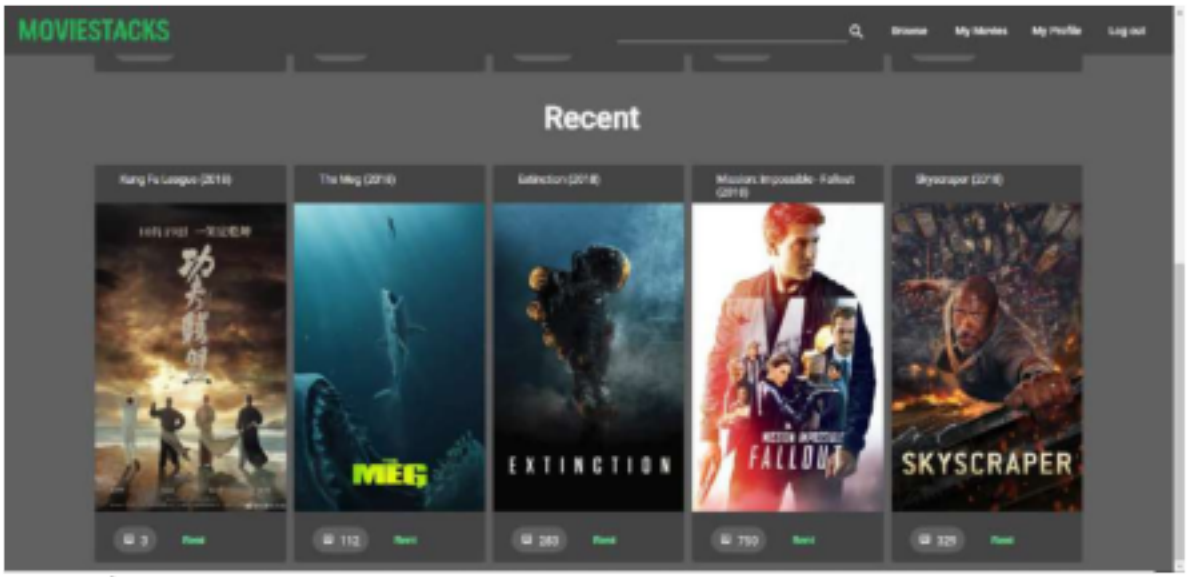
Slika 11 Prijava u aplikaciju

Korisnik se nakon uspješne aktivacije preusmjerava natrag na početnu stranicu aplikacije, i klikom na gumb Log In, otvara se forma prijave u aplikaciju (slika 11). Korisnik unosi svoje podatke, i klikom na Log In gumb (ovog puta na dnu forme), prijavljuje se u aplikaciju.



Slika 12 Početna stranica prijavljenog korisnika (Mjesečna top lista filmova)

Nakon uspješne prijave, korisniku se otvara nova početna stranica, koja sadrži top listu najposuđivanijih filmova (Slika 12) za trenutni mjesec te se otvaraju opcije pretraživanja filmova (gumb Browse), pregleda posuđenih filmova (gumb My Movies) te profila (gumb My Profile) i opcija odjave (gumb Log Out) na alatnoj traci aplikacije.

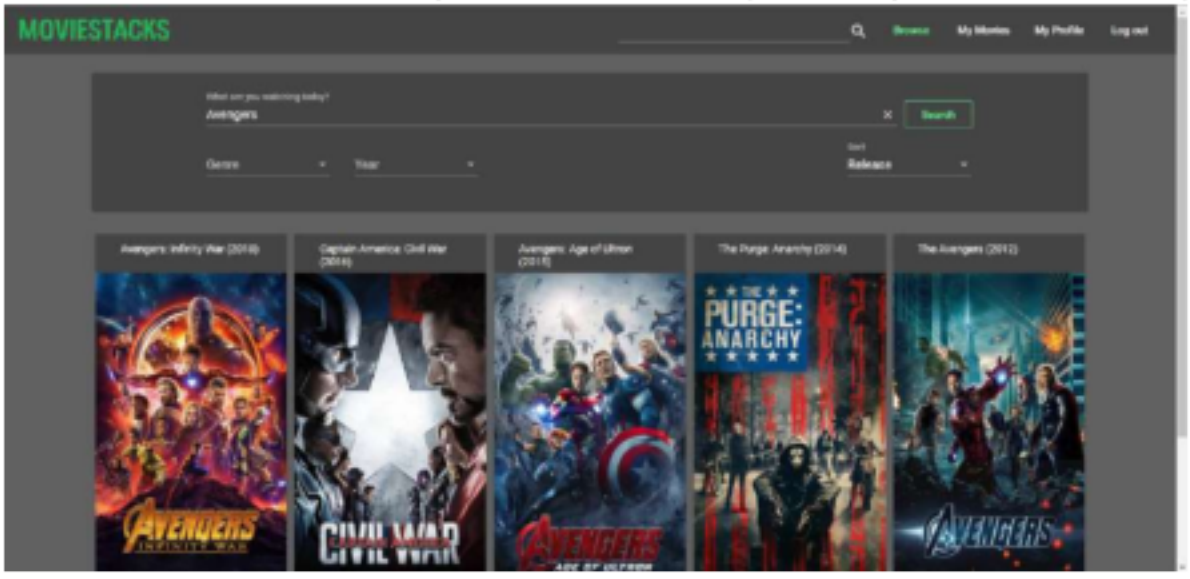


Slika 13 Početna stranica prijavljenog korisnika (5 novoizdanih filmova)

Ispod top liste najposuđivanijih filmova prikazuje se pet najnovijih filmova (slika 13).

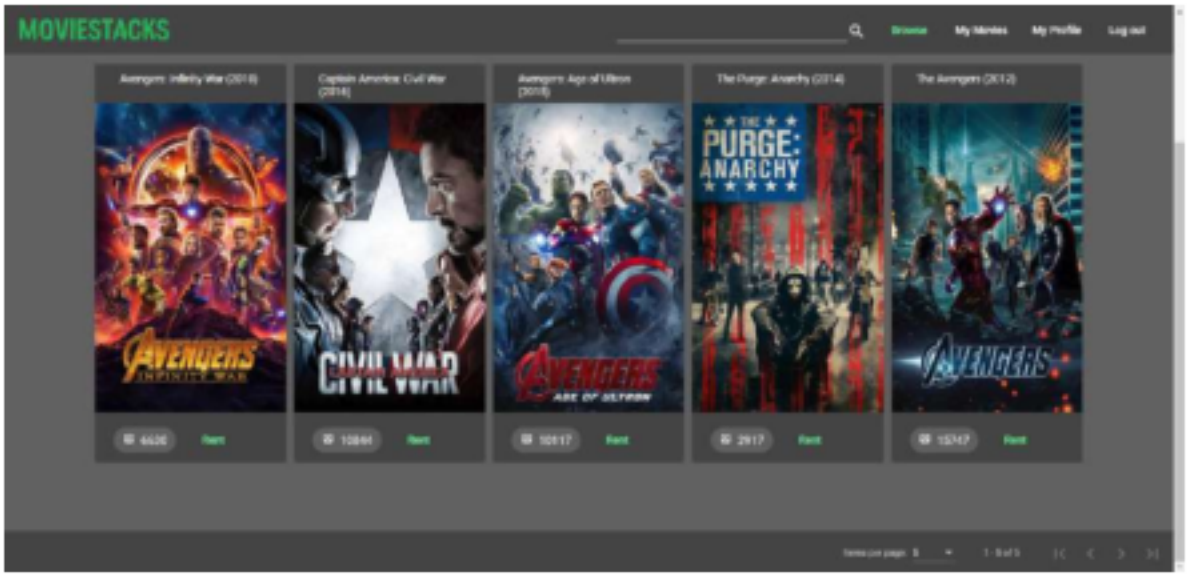
Slika 14 Pretraživanje filmova (prije pretrage)

Klikom na gumb Browse, otvara se pretraživanje filmova. Filmove je moguće pretraživati prema ključnoj riječi. Web API izvršit će pretraživanje pomoću upita na MongoDB tekstualni indeks, definiran na atributima naslova i kratkog opisa filma. Osim ključnih riječi, korisnik može odabrati jedan žanr (padajući izbornik Genre) ili godinu (padajući izbornik Year),



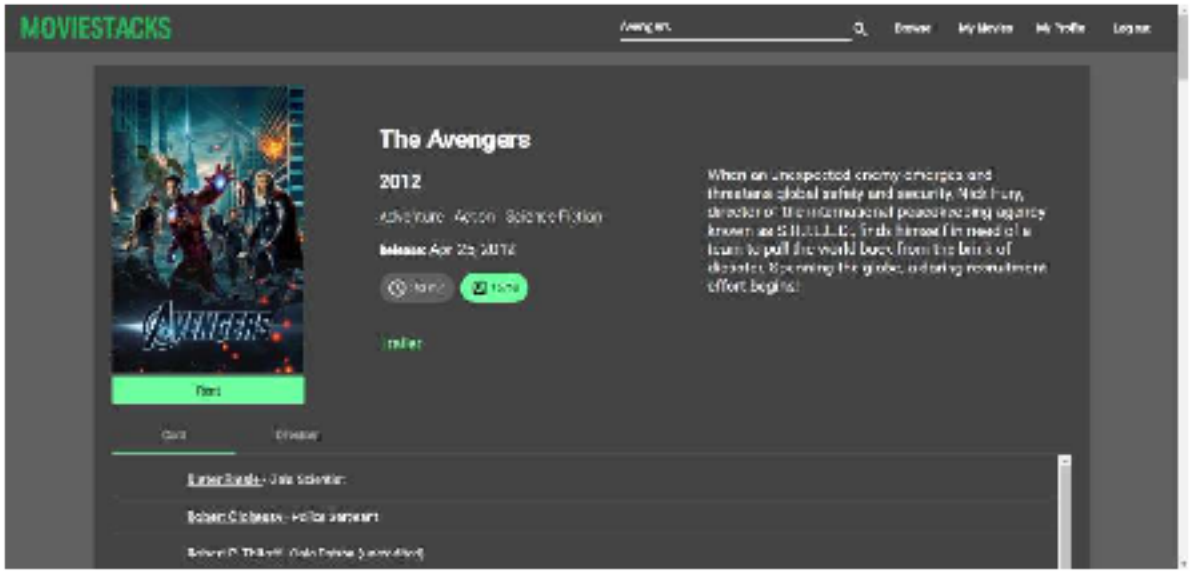
Slika 15 Pretraživanje filmova (nakon pretrage)

koji će biti filter pretraživanja te rezultate pretrage može sortirati prema vremenu izdavanja filma (silazno) ili broju najmova (silazno). Slika 15 prikazuje rezultate pretrage nad ključnom riječi „Avengers“, sortirano silazno prema datumu izdavanja filma.



Slika 16 Rezultati pretrage

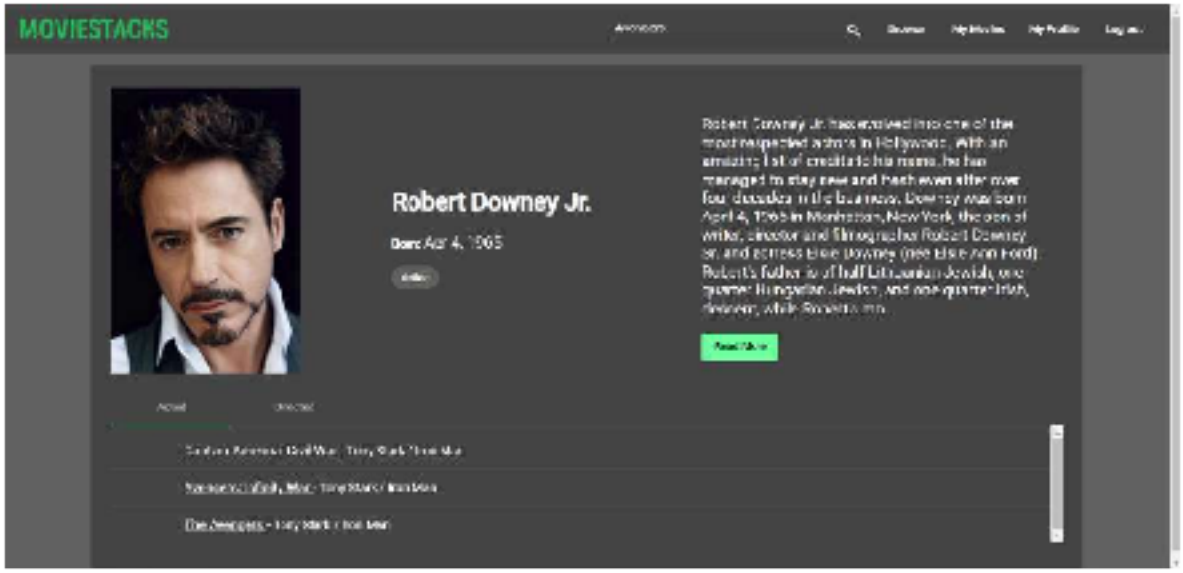
Rezultati pretrage (slika 16) prikazuju se u obliku mreže slika postera filmova, iznad kojih se nalazi naslov filma, a ispod kojih se broj najmova (lijevo) nad filmom na razini cijele aplikacije i gumb Rent (desno), kojim korisnik može iznajmiti film.



Slika 17 Detalji filma

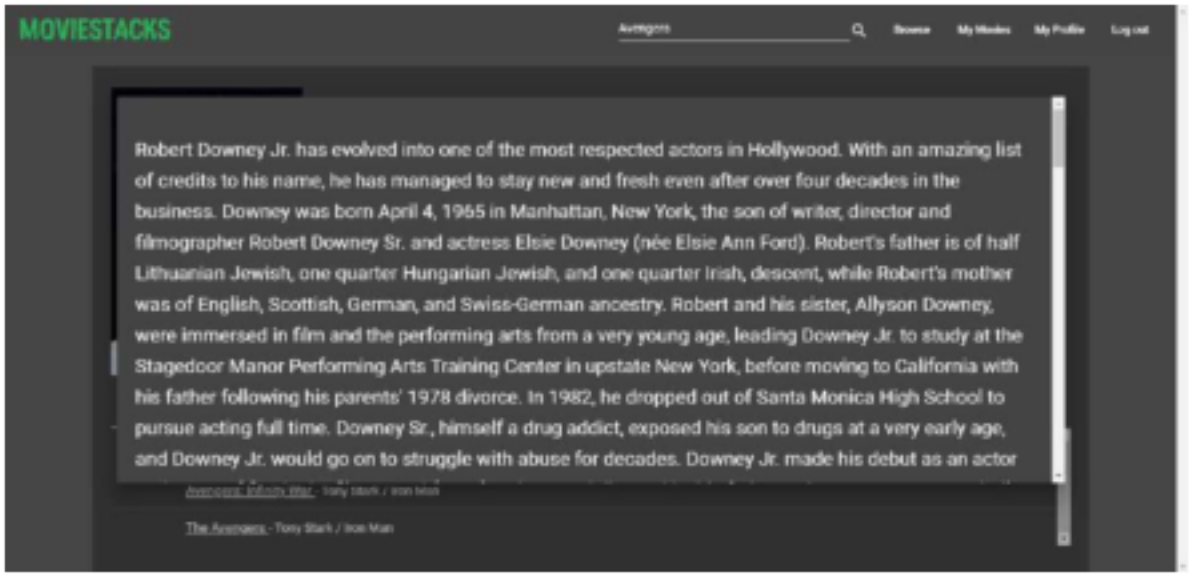
U slučaju da korisnik klikne na jedan od filmova rezultata pretrage, otvara se stranica detalja filma prikazana na slici 17. Stranica detalja filma u svojem gornjem dijelu prikazuje sve podatke filma (naslov, godinu izdanja, žanrove, datum premijere, trajanje, broj najmova, poveznicu na najavu filma i kratki opis radnje filma). Osim podataka, najam film je moguć i s ove stranice klikom na gumb Rent. Ispod podataka filma, prikazuju se dvije kartice, kartica

Cast koja sadrži popis svih uloga filma s pripadajućim glumcima, a kartica Directors sadrži popis redatelja filma. Klikom na neko od imena, korisnika se preusmjerava na stranicu tog glumca/redatelja prikazana na slici 18.



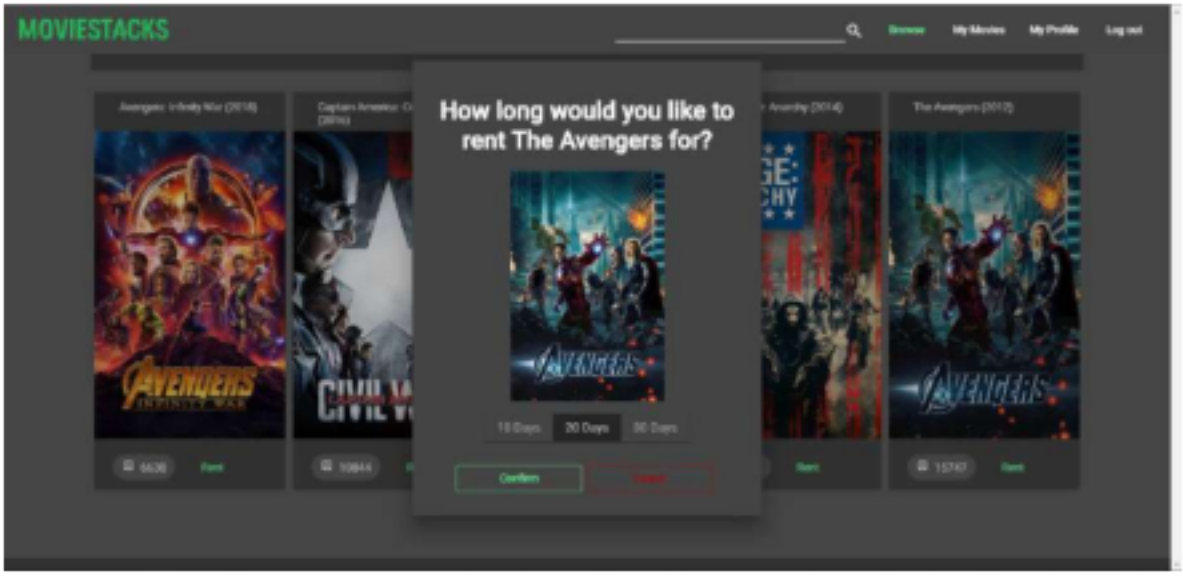
Slika 18 Detalji glumca

Stranica detalja glumca na gornjem dijelu prikazuje podatke glumca (ime, prezime, datum rođenja, datum smrti, zanimanja i kratku biografiju glumca). Ispod podataka glumca, nalaze se dvije kartice, kartica Acted sadrži popis svih filmova u kojima je glumac glumio zajedno s pripadajućim imenom uloge i kartica Directed koja sadrži popis svih filmova na kojima je glumac bio redatelj.



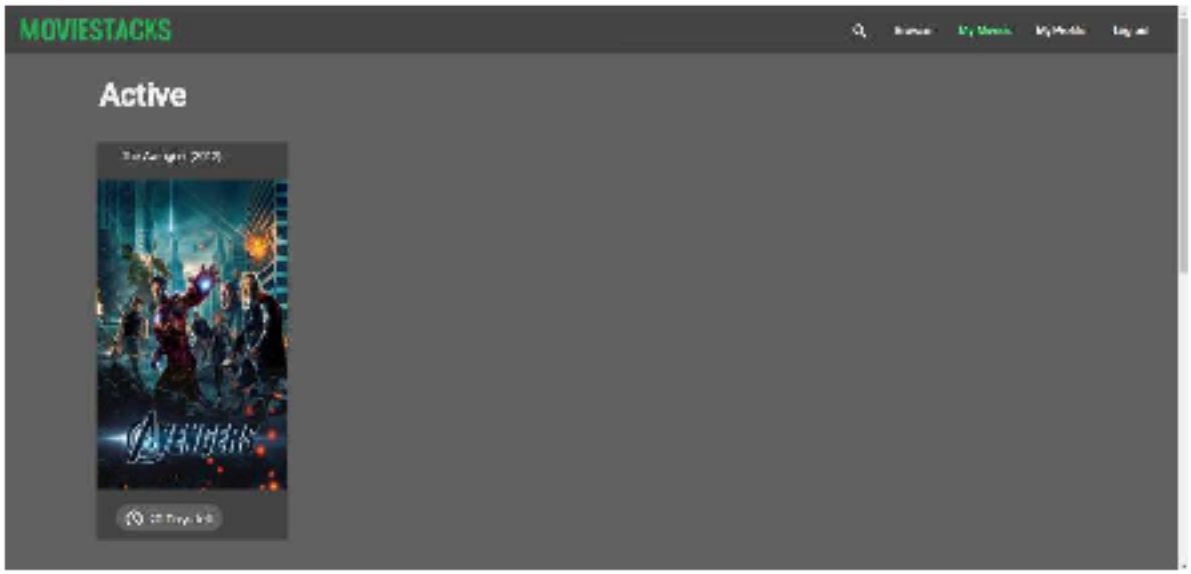
Slika 19 Detalji glumca (Modalni prozor sa punom biografijom)

Slika 19 prikazuje otvoren modalni prozor, čije je otvaranje uzrokovano klikom na gumb Read More na stranici detalja glumca (Slika 18). Modalni prozor sadrži punu biografiju glumca te se zatvara klikom izvan tog prozora.



Slika 20 Najam filma

Na stranicama detalja filma i rezultatima pretrage postoje gumbi s tekstom Rent. Klikom na te gumbe, otvara se skočni prozor najma filma prikazan na slici 20. Korisnik bira jedan od tri ponuđena trajanja filma, te ima mogućnost obustaviti radnju klikom na gumb Cancel ili potvrditi radnju klikom na gumb Confirm. Nakon klika na gumb Confirm, korisnika se preusmjerava na stranicu njegovih iznajmljenih filmova My Movies.



Slika 21 Iznajmljen filmovi (Aktivan najam)

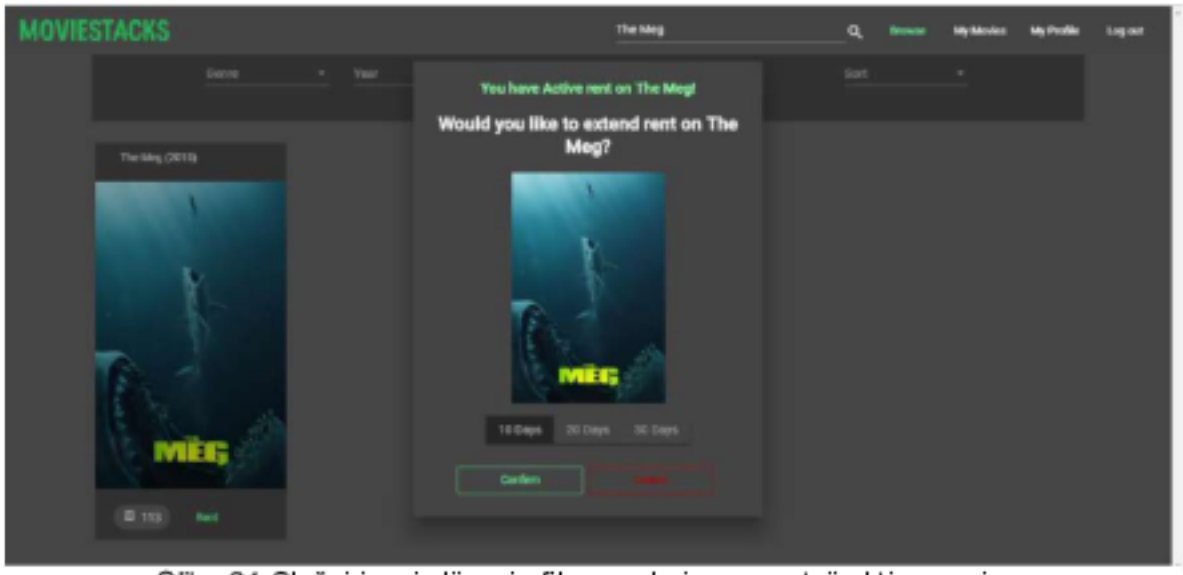
Na stranici My Movies (hrv. Moji Filmovi) na slici 21, korisnik ima uvid u sve filmove koje je iznajmio u aplikaciji. Ukoliko je najam još uvijek važeći, film će biti prikazan u gornjem dijelu stranice ispod naslova Active.

Slika 22 Iznajmljen filmovi (Istekao najam)

Nakon što najam za film istekne, u ovom slučaju 20 dana kao što piše na slici 21, korisniku se film kojem je istekao najam prikazuje ispod aktivnih filmova ispod naslova Expired (Slika 22).

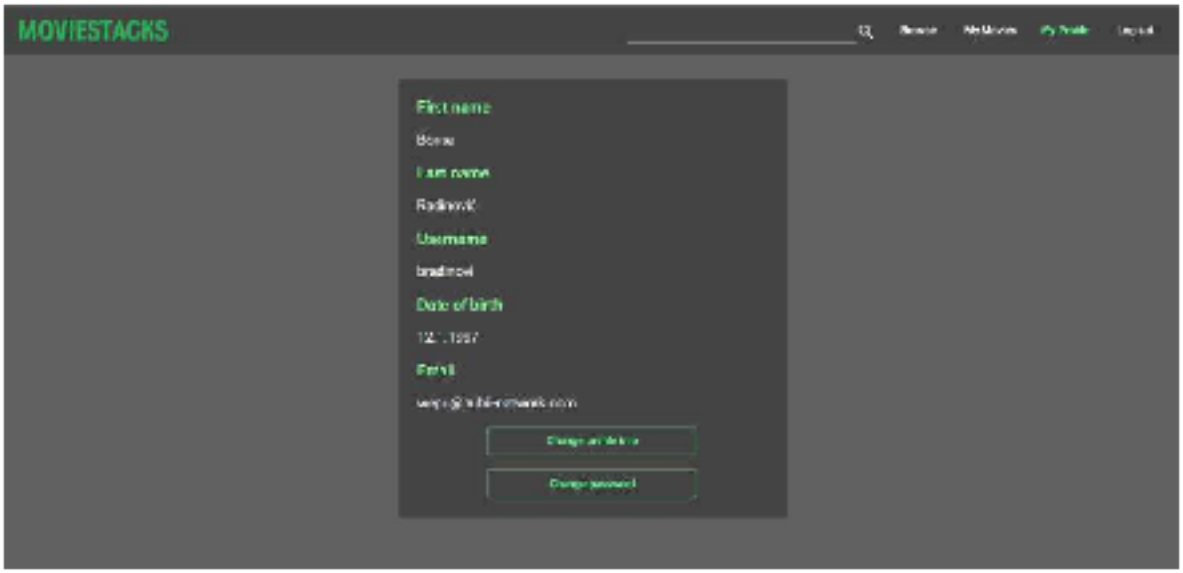
Slika 23 Brza pretraga

Alatna traka nudi opciju brze pretrage prema ključnoj riječi. Nakon unosa ključne riječi, korisnika će aplikacija preusmjeriti na stranicu pretrage (Slika 15) i prikazati filmova nađene za zadanu ključnu riječ.



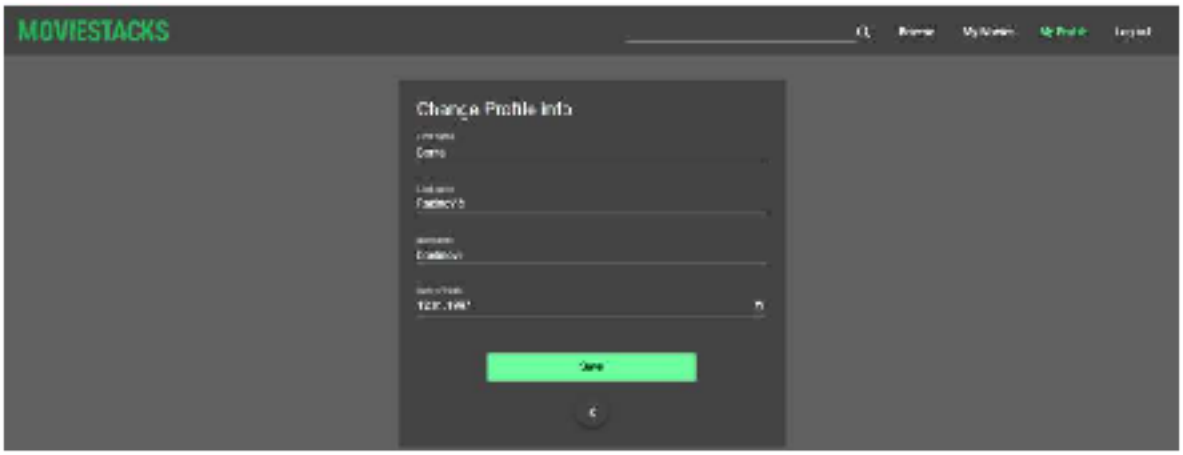
Slika 24 Slučaj iznajmljivanja filma na kojemu postoji aktivan najam

Slika 24 prikazuje slučaj u kojemu je korisnik odlučio iznajmiti film, koji je već iznajmljen, to jest kojemu najam još nije istekao. Kada se dogodi takav slučaj, aplikacija će korisnika obavijestiti o aktivnom najmu i ponuditi produljenje najma, te ako se korisnik odluči produljiti najam inkrementirati duljinu trajanja najma u bazi podataka i samim time produljiti najam filma.



Slika 25 Profil korisnika

Korisnik aplikacije u svakom trenutku ima mogućnost pogledati svoje podatke profila klikom na gumb My Profile na navigacijskoj traci aplikacije. Stranica profila korisnika prikazana je na slici 25, korisnik vidi podatke s kojima se registrirao te mu se nudi opcija mijenjanja tih podataka klikom na gumb Change profile info ili mijenjanja lozinke korisničkog računa klikom na gumb Change Password.



Slika 26 Forma promjene osobnih podataka profila

U slučaju da se korisnik odlučio na promjenu podataka profila, otvara se forma za promjenu osobnih podataka profila sa slike 26. Nakon što korisnik unese željene podatke, klikom na Save sprema promjene i u bazi podataka se mijenjanju ti podaci.

Slika 27 Forma promjene lozinke

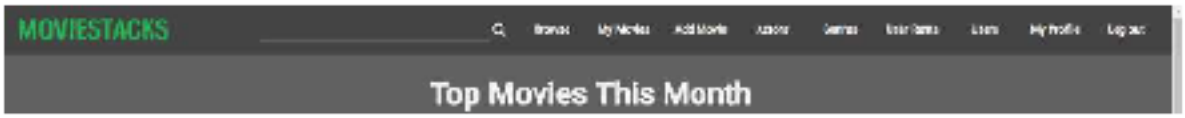
U slučaju da se korisnik odlučio na promjenu lozinke profila, otvara se forma promjene lozinke u kojoj se od korisnika traži da unese trenutnu lozinku profila i zatim unese željenu novu lozinku. Klikom na gumb Change Password korisnik potvrđuje promjenu lozinke i na poslužitelju se mijenja lozinka profila, ako je unesena validana trenutna lozinka.

Slika 28 Zaboravljena lozinka

Poznato je kako korisnici često zaboravljaju lozinke svojih profila, stoga na stranici prijave (slika 11) postoji poveznica Forgot your password, koju korisnik klikne ako je zaboravio svoju lozniku te se otvara forma za osvježavanje lozinke na slici 28. Korisnik unosi email adresu s kojom je registrirao profil, te će zatim Web API generirati nasumični tekstualni niz koji će poslati na email adresu korisnika i ažurirati korisnikov dokument u bazi podataka.

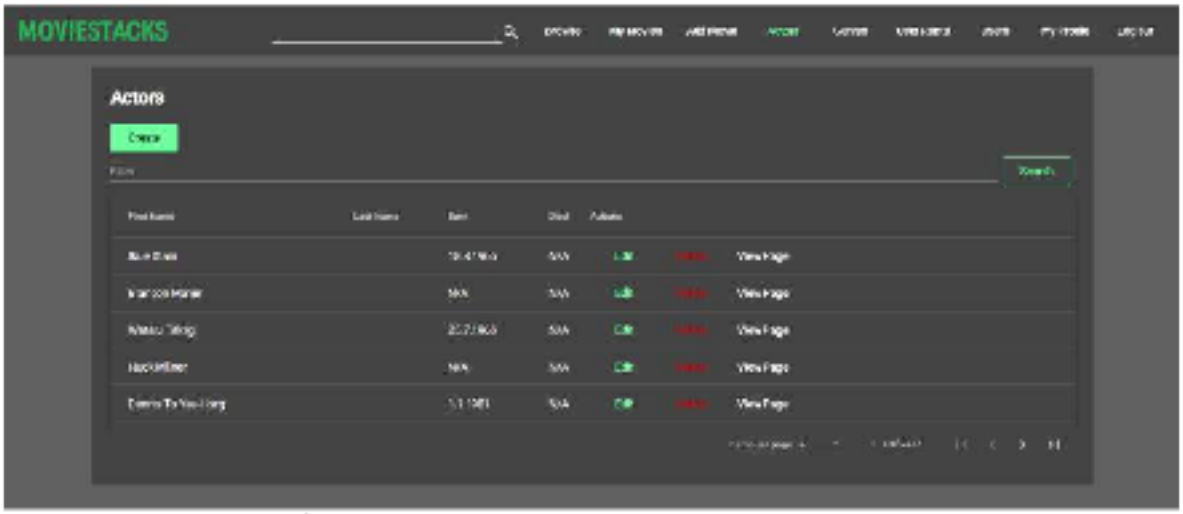
5.7.2. Pogled Administrator

Administratora plikacije ima mogućnost ažuriranja, brisanja i kreiranja svih dokumenata MongoDB baze podataka u pozadini aplikacije, što zahtjeva posebne komponente prikaza pregleda i akcija nad podacima sustava. Bitno je naglasiti kako funkcionalnosti opisane u 5.7.1. Pogled Korisnika vrijede i za administratora.



Slika 29 Navigacijska traka korisnika s administratorskim pravima

Navigacijska traka administratora sadrži pet novih gumba (slika 29). Gumb Actors preusmjerit će korisnika na pregled svih glumaca pohranjenih u bazi podataka, Genres nudi pregled svih žanrova u bazi. Gumbom User Rents administrator dobiva uvid u sve najmove filmova u bazi podataka, gumb Users prebacuje na pogled svih korisnika sustava te gumb Create Movie omogućava stvaranje novog filma u bazi podataka.



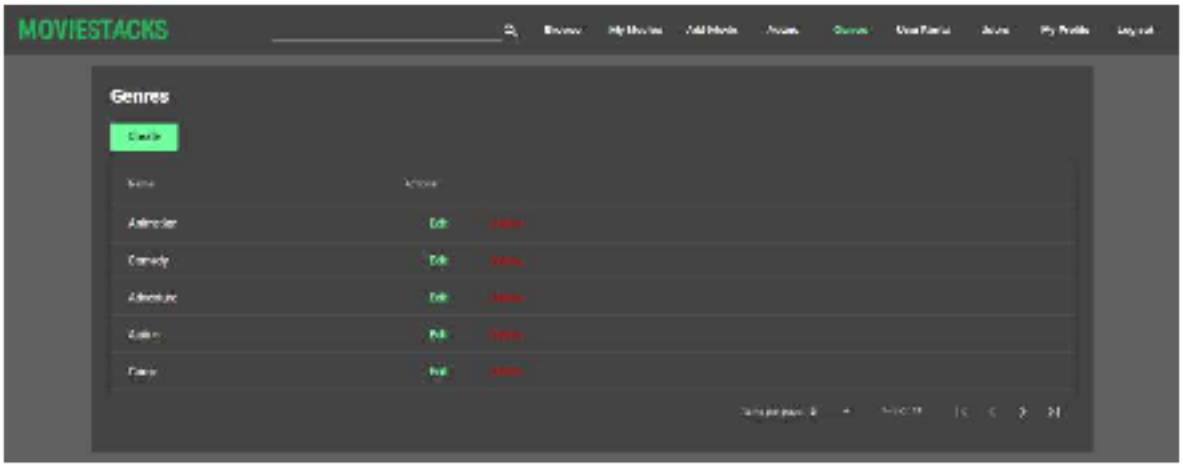
Slika 30 Prikaz svih glumaca u bazi podataka

Klikom na gumb Actor, administrator je preusmjeren na stranicu prikaza svih glumaca u bazi podataka prikazanoj na Slici 30. Administrator za svakog glumca ima mogućnost ažurirati, izbrisati ili otići na stranicu detalja glumca. Iznad tablice nalazi se filter koji, ukoliko ga administrator popuni i klikne, na gumb Search u tablici prikazuje samo one glumce koji odgovaraju ključnoj riječi pretrage. Pretraga traži ključne riječi u imenu i prezimenu glumca, te zatim u tablici prikazuje one glumce koji odgovaraju pretrazi. Osim navedenih opcija na ovoj stranici, klikom na gumb Create adminstrator može kreirati novi dokument glumca i spremiti ga u bazu podataka. Klikom na gumb Delete, korisnika se briše iz baze podataka.

Slika 31 Forma kreiranja glumca

U slučaju da se administrator odluči stvoriti novog glumca u sustavu, klikom na gumb Create sa slike 30, preusmjerava ga se na stranicu s formom za unos novog glumca (slika 31). Nakon završetka unosa podataka i odabrane slike, administrator novog glumca sprema klikom na gumb Save, te ga se tada preusmjerava na stranicu Prikaza svih glumaca sa slike 30.

U slučaju da je administrator na stranici sa slike 30, na jednom od redova tablice, kliknuo na gumb Edit, preusmjerava ga se na istu stranicu s formom na slici 31, ali ovog puta se forma popunjava s postojećim podacima. Nakon željenih izmjena, administrator promjene sprema klikom na gumb Save, kao i kod kreiranja glumca.



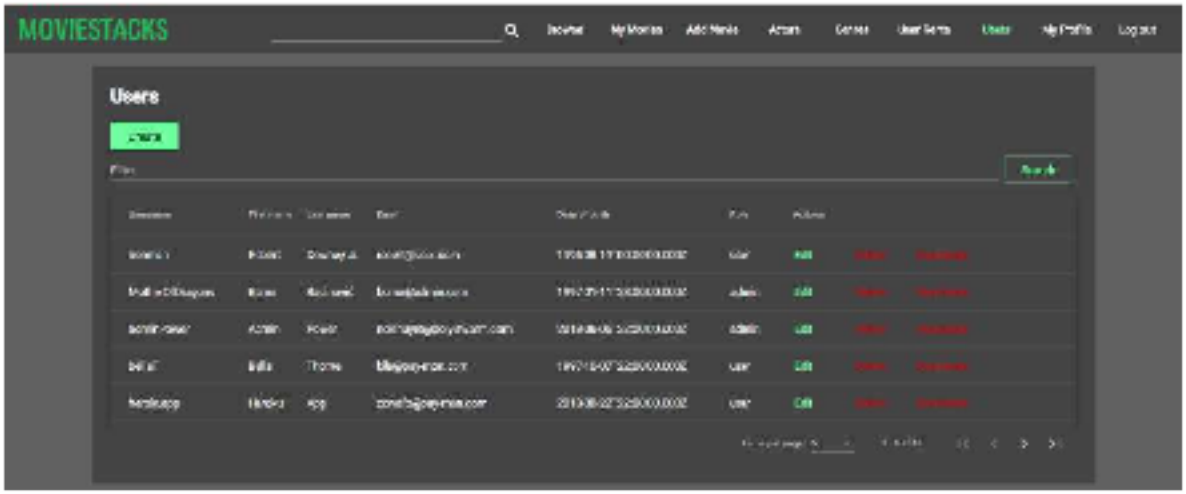
Slika 32 Prikaz svih žanrova iz baze podataka

Kilkom na Gumb Genres, administratoru se otvara tablica prikaza svih žanorva unesenih u bazu podataka, prikazano na slici 32. Administrator na ovoj stranici može stvoriti, ažurirati i izbrisati žanr. Klikom na gumb Delete briše se žanr iz baze podataka.

U slučaju da se administrator odlučio kliknuti na gumb Create za kreiranje novog žanra,

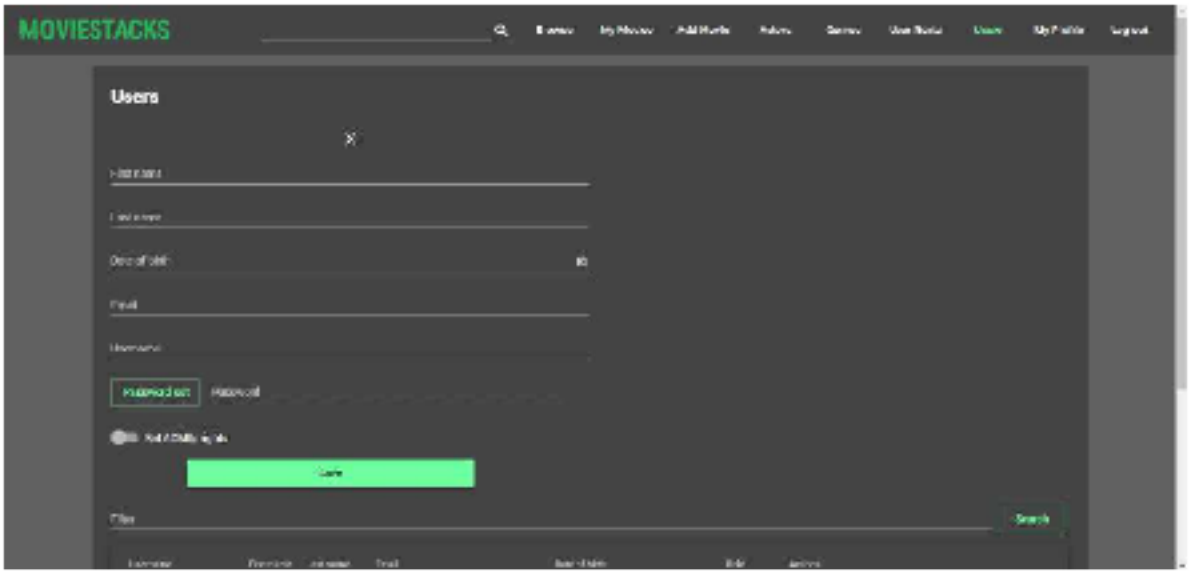
Slika 33 Forma unosa novog žanra

iznad tablice prikaza žanrova otvara se forma za unos novog žanra, kao što je vidljivo na slici 33. Nakon što administrator unese ime novog žanra, klikom na gumb Save sprema novi žanr u bazu podataka. Ista forma samo s popunjenim imenom žanra, otvara se u slučaju klika na gumb Edit u redu tablice prikaza žanra te se ažuriranje sprema klikom na gumb Save. Klikom na gumb Delete u tablici prikaza žanrova, žanr se briše iz baze podataka.



Slika 34 Prikaz svih korisnika sustava

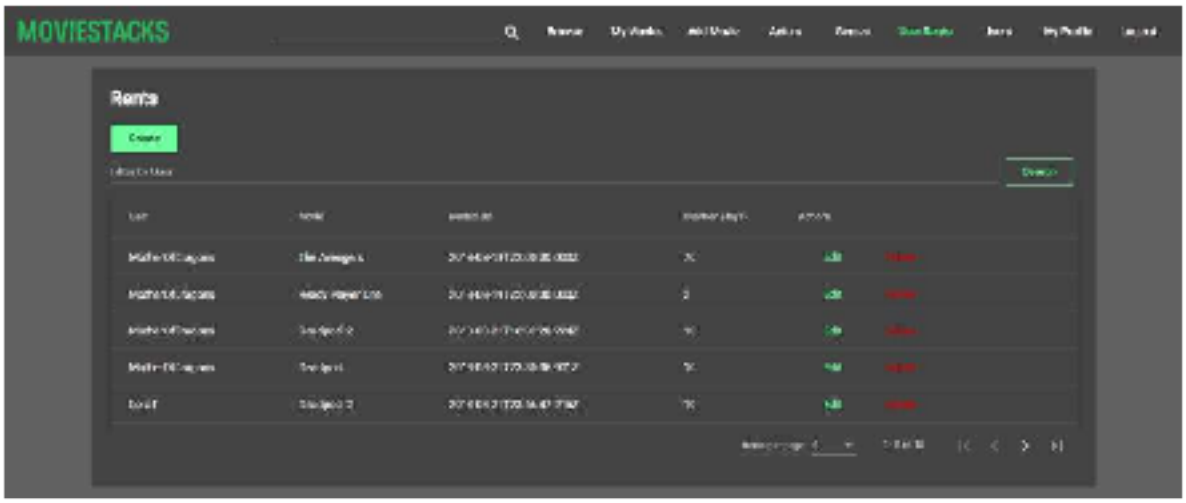
Pregled svih korisnika sustava, administratoru je moguć klikom na gumb Users na alatnoj traci. Svi korisnici prikazani su tablicom na slici 34. Administrator na ovoj stranici može kreirati, ažurirati, izbrisati i deaktivirati korisnika. Ukoliko administrator klikne na gumb Deactivate u redu tablice korisnika, korisnik naveden u tom redu se deaktivira, to jest ne može se prijaviti na sustav do ponove aktivacije. Klikom na gumb Create, otvara se forma za kreiranje novog korisnika prikazana na slici 35. Brisanje korisnika iz sustava, samim time iz baze podataka, vrši se klikom na gumb Delete u redu tablice za prikaz korisnika.



Slika 35 Forma kreiranja novog korisnika

Popunjavanjem podataka i odabirom uloge korisnika u formi na slici 35, administrator nakon klika na gumb Save, sprema novog korisnika u bazu podataka.

U slučaju da je administrator kliknuo na gumb Edit, u redu tablice na slici 34 otvara se ista forma kao na slici 35, samo popunjena. Administrator kod ažuriranja ne mora nužno ažurirati lozinku korisnika, ali ako se odluči ažurirati, i taj podatak gumbom Password set, omogućava polje forme za postavljanje lozinke. Pripadni podaci se nakon klika na gumb Save, spremaju u bazu podataka.

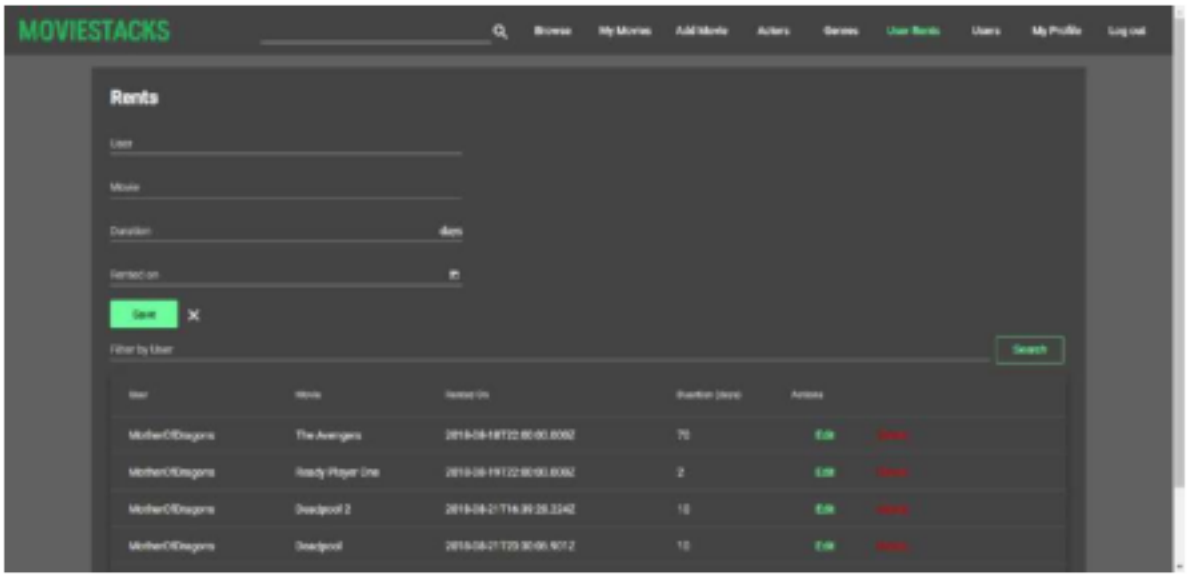


Slika 36 Prikaz najmova filmova svih korisnika

Administrator najmove filmova prati na stranici prikazanoj na slici 36, klikom na gumb User Rents na navigacijskoj traci aplikacije. Prikaz najmova moguće je obaviti prema korisniku. U polju Filter By User administrator početkom pisanja po polju pokreće automatsko dovršavanje polja, koje mu nudi korisnike koji odgovaraju unesenoj vrijednosti, prikaz automatsko dovršavanja vidljiv je na slici 37.

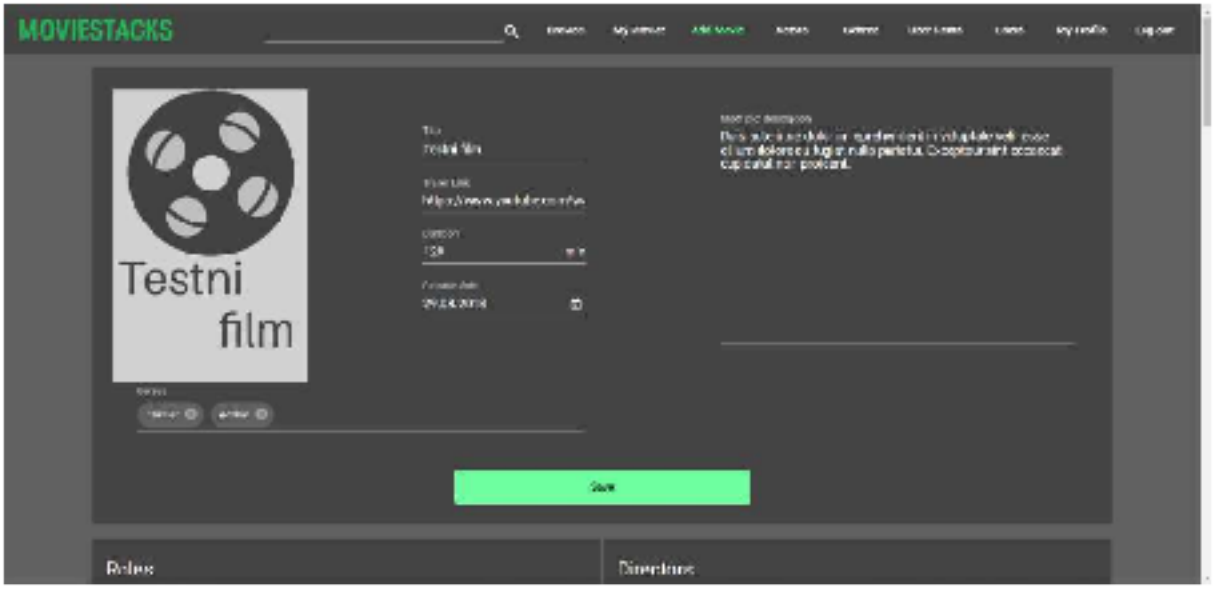
Slika 37 Automatsko dovršavanje polja za odabir korisnika za koji se želi vidjeti najmove

Nakon što administrator odabere korisnika, iz padajućeg izbornika klikom na Search u tablici, dobiva ispis svih najmova tog korisnika.



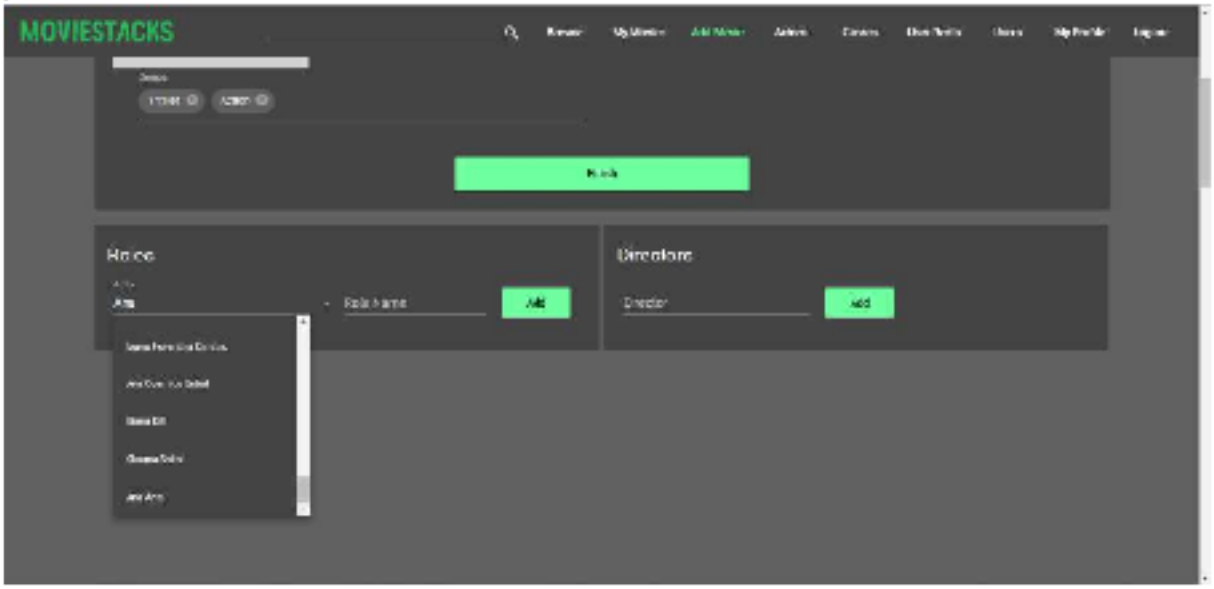
Slika 38 Forma za unos novog najma

Administrator je u mogućnosti stvoriti novi najam filma za bilo kojeg korisnika sustava. Klikom na gumb Create na stranici sa slike 37, otvara se forma za unos novog najma (slika 38). Nakon unosa podataka potrebnih za stvaranje najma, klikom na Save novi najam sprema se u bazu podataka. Ista forma, ali popunjena, otvara se ako administrator klikne na gumb Edit u tablici na stranici slike 37, tom akcijom administrator ažurira taj najam. Nakon što su promijenjeni željeni podaci najma, ažuriranje administrator sprema klikom na gumb Save te se ažuriranje izvršava i u bazi podataka.



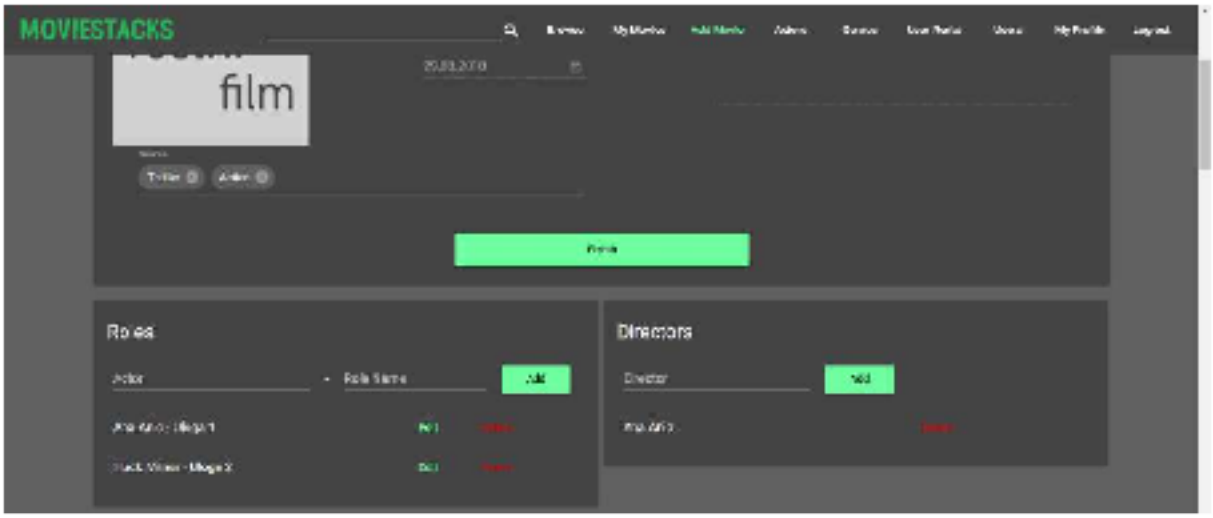
Slika 39 Stranica kreiranja novog filma

Kreiranje filmova moguće je klikom na gumb Add Movie na navigacijskoj traci, nakon čega se otvara forma za unos podataka i slike novog filma (slika 39). Nakon što administrator popuni podatke filma i odabere sliku, potrebno je kliknuti na gumb Save koji sprema podatke novog filma. Nakon te akcije omogućuju se forma za unos uloga filma i redatelja filma prikazane na slici 40.



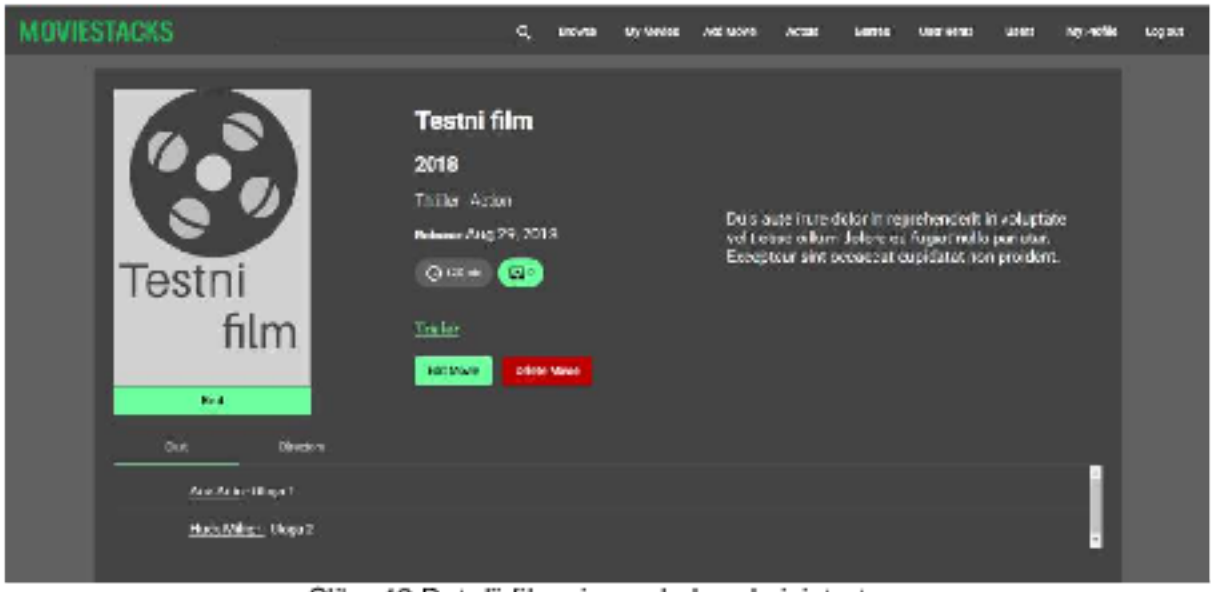
Slika 40 Forme unosa uloga i redatelja filma

Uloge filma unose se formom lijevo na slici 40, potrebno je iz padajućeg izbornika izabrati glumca, koji glumi ulogu i u desnom polju unijeti ime uloge te kliknuti na gumb Add, kako bi se uloga dodala u uloge. Redatelji filma unose se desnom formom odabirom glumca i klikom na gumb Add.



Slika 41 Forme unosa glumca i redatelja nakon unosa

Nakon unosa uloge, one su ispisane ispod forme za unos uloga te ih je moguće ažurirati klikom na gumb Edit i zbrisati klikom na gum Delete, isto vrijedi za formu unosa redatelja. Nakon što je administrator zadovoljan s unosom filma, radnju završava klikom na gumb Finish, što ga preusmjerava na stranicu novokreiranog filma vidljivog na slici 42.



Slika 42 Detalji filma iz pogleda administratora

Stranica prikaza detalja filma, iz pogleda administratora, sadrži gumbe za ažuriranje filma (gumb Edit) i brisanje filma (gumb Delete). U slučaju da administrator klikne gumb Edit, otvara mu se popunjena forma, kao ona na slici 39 i 40, te tamo može ažurirati podatke o filmu, uloge i redatelje. Gumbom Delete administrator film briše iz baze podataka.

6. Kritički prikaz

Polustrukturirani podaci nude pristup oblikovanju i pohranjivanju podataka koji je fleksibilniji od relacijskog modela. Njihova fleksibilnost proizlazi iz njihove prirode polovične strukturiranosti, koja omogućava prikaz domene podataka na način da svi entiteti domene ne moraju nužno sadržavati iste atribute. Svaka instanca entiteta u polustrukturiranom modelu podataka tretira se kao zasebni objekt, čija je struktura promjenjiva dok kod relacijskih modela to nije slučaj. Kod relacijskog modela za svaki entitet potrebno je definirati relacijsku tablicu, koja sadrži sve atribute entiteta koji kod pohranjivanja instanci entiteta ne moraju postojati, te zbog toga ostaju prazni u relacijskoj tablici što uzrokuje probleme kod upita nad tablicom.

Razvojni programeri i administratori baza podataka često se susreću s domenama promatranja, koje su po prirodi nepotpuno strukturirane pogotovo od pojave weba. Poslustrukturirani podaci oslobađaju ih od striktnog definiranja modela podataka te omogućuju dodavanje ili brisanje atributa podatka, koji će se pohranjivati u bazu tokom samog razvoja. Osim toga, upitni jezici nad poslustrukturiranim podacima jednostavnije su koncipirani, nego oni u relacijskom modelu. U relacijskom modelu kod definiranje modela, dizajneri baza podataka podataka moraju razmišljati o upitima, to jest kako na što efikasniji način pohraniti podatke, kako bi upiti spojeva tablica bili što jednostavniji i vjerno prikazivali domenu za koju dizajniraju bazu podataka.

Pohrana polustrukturiranih podataka u MongoDB izvedena je pomoću u obliku JSON dokumenata. Za razliku od relacijskih podataka koji su limitirani na prikaz redova i tablica, dokumenti u sebi mogu prikazati polja ili čak u sebi imati druge dokumente, zbog ovih osobina dokumenti zapravo podatke prikazuju na isti način, kao i objekti u objektno orijentiranom programiranju. Sličnost prikaza dokumenata i objekata stavlja dokumente u bolju poziciju za pretvorbu u objekte od podataka vraćenih od relacijske baze podataka, podaci vraćeni od strane relacijske baze podataka zahtijevaju kompleksnije parsiranje.

Sustav za upravljane polustrukturiranim podacima MongoDB slobodniji je po pitanju validacije podataka, koji se pohranjuju. Relacijske baze podataka u sebi imaju ugrađene mehanizme validacije podataka, zbog kojih razvojni programer ili dizajner baze podataka mora prilagoditi model podataka, što u konačnici i nije loše jer se osigurava validnost pohranjenih podataka, to jest osigurava njihovu konzistentnost što je bitno kod ACID modela. U slučaju da razvojnom programeru ili dizajneru baze podataka više odgovara BASE model te želi konzistentnost i validaciju uzeti u svoje ruke, pogodnije je koristiti sustav za upravljanje polustrukturiranim podacima MongoDB.

7. Zaključak

Polustrukturirani pristup zapisivanja podataka omogućio je način zapisivanja podataka promjenjive strukture. Notacije zapisa ovog tipa nastale su uzimanjem u obzir prikazivanja takvih podataka, što rješava problem kompleksnosti preoblikovanja podataka iz baze u oblik pogodan prikazu.

Pohranjivanje podataka polustrukturiranog tipa radi se u NoSQL bazama podataka, to jest dokumentno orijentiranim bazama podataka, koje su najpogodnije za polustrukturirane podatke, zbog korištenja notacije pohranjivanja podatka u bazu jednake onima za prikaz polustrukturiranog tipa podataka (XML i JSON). NoSQL baze podataka drže se BASE svojstava, stoga nisu pogodne za sve vrste podataka, pogotovo ne za one koje zahtijevaju garantiranu konzistentnost.

Sustav za upravljanje polustrukturiranim bazama podataka MongoDB, osim mogućnosti upravljanja polustrukturiranim podacima, svojom arhitekturom daje mogućnosti horizontalne skalabilnosti, koja je bitna kod naglog porasta količina podataka na poslužiteljima, što je slučaj kod podataka namijenjenih za web.

Sve u svemu, kod odabira uporabe MongoDB sustava za upravljanje polustrukturiranim podacima, mora se voditi računa o prirodi podataka koji će se pohranjivati. Poželjno je da podaci koji će biti spremani u bazu podataka, što više odgovaraju svojstvima polustrukturiranih podataka, te će se tada uočiti prava prednost uporabe ove vrste baze podataka.

Implementacija aplikacije opisane u ovom radu prikazala je kako iskoristiti MongoDB, na primjeru iz stvarnog svijeta. Razvoj Web API-ja koji odgovore šalje u JSON formatu, uvelike je bio olakšan samom MongoDB prirodom pohranjivanja podataka u JSON formatom. Iste te podatke lako je prebaciti u DOM web preglednika za prikaz korisniku. U implementaciji ima prostora za poboljšanje. Aplikacija bi se mogla proširiti na način da podržava funkcionalnost reprodukcije iznajmljenih filmova, koji bi zahtijevao određen sustav naplate. Navedene funkcionalnosti ne zahtijevaju drastične promjene arhitekture i koda. Komponentna priroda Angulara na klijentskoj strani omogućila bi dodavanje potrebnih prikaza, kao novih komponenti. Što se tiče baze podataka, navedene promjene značile bi dodavanje novih kolekcija, a na poslužiteljskoj strani Web API aplikacije dodale bi se odgovarajuće rute i logike obrade zahtjeva.

Popis literature

[1] W3C, „The history of the Web - W3C Wiki“. [Na internetu]. Dostupno: https://www.w3.org/wiki/The_history_of_the_Web. [Pristupano: 24-kol-2018].

[2] Internet Live Stats, „Total number of Websites - Internet Live Stats“, 2015. [Na internetu]. Dostupno: http://www.internetlivestats.com/total-number-of-websites/#trend. [Pristupano: 24-kol-2018].

[3] S. Abiteboul, P. Buneman, i D. Suciu, *Data on the Web: From Relations to Semistructured Data and XML (The Morgan Kaufmann Series in Data Management Systems)*. San Francisco, California, USA: Morgan Kaufmann Publishers, 1999.

[4] T. Wang Ling, M. Li Lee, i G. Dobbie, *Semistructured Database Design*. Boston,MA ,USA: Springer, 2005.

[5] M. Maleković i M. Schatten, *Teorija i primjena baza podataka*, 1. izd. Varaždin, Hrvatska: Sveučilište u Zagrebu, Fakultet Organizacije i Informatike, 2017.

[6] K. Ruohonen, *GRAPH THEORY*. Ruohonen, Keijo, 2013.

[7] W3C, „Extensible Markup Language (XML) 1.1 (Second Edition)“, 2006. [Na internetu]. Dostupno: https://www.w3.org/TR/2006/REC-xml11-20060816/#sec-origin-goals. [Pristupano: 25-kol-2018].

[8] L. Bassett, *Intorduction to JavaScript Object Notation A To-The-Point Guide To JSON*. North, Sebastopol, CA, SAD: O'Reilly Media, 2015.

[9] B. Smith, *Beginning JSON*. Apress, 2015.

[10] D. Crockford, „RFC 4627 - The application/json Media Type for JavaScript Object Notation (JSON)“, 6427, 2006.

[11] I. E. T. F. (IETF) Bray, Ed., „RFC 7159 - The JavaScript Object Notation (JSON)— Data Interchange Format“, 2014.

[12] ECMA International, „ECMA-262 ECMAScript ® 2018 Language Specification“, 2009.

[13] „A Collective List Of APIs, Go Build Something.“ [Na internetu]. Dostupno: https://apilist.fun/. [Pristupano: 25-kol-2018].

[14] Partha Sarathi, „A deep dive into NoSQL: A complete list of NoSQL databases“, 2014. [Na internetu]. Dostupno: http://bigdata-madesimple.com/a-deep-dive-into-nosql-a-complete-list-of-nosql-databases/. [Pristupano: 25-kol-2018].

[15] G. Vaish, *Getting Started with NoSQL*. Birmingham, UK - Mumbai, IND: PACKT, 2013.

[16] Prof. Dr. Stefan Edlich, „NoSQL Databases“. [Na internetu]. Dostupno: http://nosql-database.org/index.html. [Pristupano: 26-kol-2018].

[17] D. Sullivan, *NoSQL Fore Mere Mortals*. Boston, Massachusetts, SAD: Addison-Wesley, 2015.

[18] K. Banker, P. Bakkum, S. Verch, D. Garret, i T. Hawkins, *MongoDB in Action*, 2. izd. Shelter Islan, NY, SAD: Manning, 2011.

[19] K. Chodorow, *MongoDB The Definitive Guide*, 2. izd. Sebastopol, CA, SAD: O'REILLY, 2013.

[20] MongoDB inc., „Indexes — MongoDB Manual“. [Na internetu]. Dostupno: https://docs.mongodb.com/manual/indexes/. [Pristupano: 27-kol-2018].

[21] MongoDB inc., „db.createCollection() — MongoDB Manual“. [Na internetu]. Dostupno: https://docs.mongodb.com/manual/reference/method/db.createCollection/. [Pristupano: 27-kol-2018].

[22] MongoDB inc., „Query Documents — MongoDB Manual“. [Na internetu]. Dostupno: https://docs.mongodb.com/manual/tutorial/query-documents/index.html. [Pristupano: 27-kol-2018].

[23] M. Schwarzmüller, „Angular & NodeJS - The MEAN Stack Guide“, *Udemy, Inc.*, 2018. [Na internetu]. Dostupno: https://www.udemy.com/angular-2-and-nodejs-the-practical-guide/.

[24] A. Q. Haviv, *MEAN Web Development*, 2. izd. Birmingham, UK - Mumbai, IND: Pack Publishing Ltd, 2016.

[25] Google, „Angular logotip“. [Na internetu]. Dostupno: https://angular.io/assets/images/logos/angular/angular.png. [Pristupano: 02-ruj-2018].
[26] Node.js Foundation, „Express logotip“. [Na internetu]. Dostupno: https://images.g2crowd.com/uploads/product/image/social_landscape/social_landscape_1489710851/express-js.png. [Pristupano: 02-ruj-2018].
[27] LearnBoost Inc., „Mongoose logotip“. [Na internetu]. Dostupno: https://cdn-images-1.medium.com/max/900/1*3F5eonRQqcP35KglajAa8Q.png. [Pristupano: 02-ruj-2018].
[28] Node.js Foundation, „Node.js logotip“. [Na internetu]. Dostupno: https://nodejs.org/static/images/logos/nodejs-new-pantone-black.png. [Pristupano: 02-ruj-2018].
[29] MongoDB Inc., „MongoDB logotip“. [Na internetu]. Dostupno: https://webassets.mongodb.com/_com_assets/cms/MongoDB-Logo-5c3a7405a85675366beb3a5ec4c032348c390b3f142f5e6dddf1d78e2df5cb5c.png. [Pristupano: 02-ruj-2018].
[30] B. Radinović, „Implementacija on-line videoteke“, 2018. [Na internetu]. Dostupno: https://github.com/bradinovi/on-line-videoteka.
[31] Google, „Angular Material“. [Na internetu]. Dostupno: https://material.angular.io/. [Pristupano: 29-kol-2018].

Popis slika

Popis tablica

Prilog

Programski kod implementacije on-line videoteka nalazi se u GitHub repozitoriju na poveznici:
https://github.com/bradinovi/on-line-videoteka.git