Web aplikacija za evidenciju i recenziju filmova

Vujić, Aleksej

Undergraduate thesis / Završni rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:852943

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-03-21**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Stručni prijediplomski studij Računarstvo

Web aplikacija za evidenciju i recenziju filmova

Završni rad

Aleksej Vujić

Osijek, 2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1S: Obrazac za ocjenu završnog rada na stručnom prijediplomskom studiju

Ocjena završnog rada na stručnom prijediplomskom studiju

Ime i prezime pristupnika:	Aleksej Vujić
Studij, smjer:	Stručni prijediplomski studij Računarstvo
Mat. br. pristupnika, god.	ARV 28, 24.09.2021.
JMBAG:	0111135842
Mentor:	prof. dr. sc. Krešimir Nenadić
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	izv. prof. dr. sc. Ivica Lukić
Član Povjerenstva 1:	prof. dr. sc. Krešimir Nenadić
Član Povjerenstva 2:	doc. dr. sc. Krešimir Romić
Naslov završnog rada:	Web aplikacija za evidenciju i recenziju filmova
Znanstvena grana završnog rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak završnog rada:	Dati opis zahntjeva i funkcionalnosti koje će imati web aplikacija za vođenje evidencije pregledanih filmova, filmova koje korisnik želi pogledati i ocjenjivanju filmova. Navesti nekoliko postojećih rješenja slične tematike te usporediti izrađenu web aplikaciju s postojećim rješenjima uz navođenje prednosti i nedostataka izrađene aplikacije u odnosu na postojeća rješenja. Projektirati i izraditi bazu podataka koja će se koristiti uz web aplikaciju te opisati postupak izrade baze. Web aplikacija treba imati funkcionalnost registracije korisnika, vođenja evidencije

Datum ocjene pismenog dijela završnog rada od strane mentora:	30.11.2024.
Ocjena pismenog dijela završnog rada od strane mentora:	Izvrstan (5)
Datum obrane završnog rada:	10.12.2024.
Ocjena usmenog dijela završnog rada (obrane):	Izvrstan (5)
Ukupna ocjena završnog rada:	Izvrstan (5)
Datum potvrde mentora o predaji konačne verzije završnog rada čime je pristupnik završio stručni prijediplomski studij:	10.12.2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA O IZVORNOSTI RADA

Osijek, 10.12.2024.

Ime i prezime Pristupnika:	Aleksej Vujić
Studij:	Stručni prijediplomski studij Računarstvo
Mat. br. Pristupnika, godina upisa:	ARV 28, 24.09.2021.
Turnitin podudaranje [%]:	5

Ovom izjavom izjavljujem da je rad pod nazivom**: Web aplikacija za evidenciju i recenziju filmova**

izrađen pod vodstvom mentora prof. dr. sc. Krešimir Nenadić

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis pristupnika:

SADRŽAJ

1. UVOD **1**

1.1. Zadatak završnog rada **1**

2. PREGLED POSTOJEĆIH RJEŠENJA NA TRŽIŠTU **2**

2.1. IMDB **2**

2.2. Rotten Tomatoes **3**

2.3. Letterboxd **4**

2.4. Trakt **5**

2.5. Simkl **6**

3. TEHNOLOGIJE KORIŠTENE U IZRADI APLIKACIJE **8**

3.1. Django **8**

3.2. SQLite **8**

3.3. HTML i CSS **9**

3.4. Bootstrap **9**

4. PRIKAZ POSTUPKA IZRADE APLIKACIJE **10**

4.1. Baza podataka **10**

4.2. Pogledi **13**

4.2.1. Autentikacija korisnika 13

4.2.2. Upravljanje podatcima 15

4.2.3. Upravljanje URL-ovima 18

4.2.4. Predlošci 18

5. PRIKAZ RADA WEB APLIKACIJE **22**

6. ZAKLJUČAK **28**

LITERATURA **29**

SAŽETAK **30**

ABSTRACT **31**

ŽIVOTOPIS **32**

1. UVOD

U ovom je završnom radu dan pregled razvoja programskog rješenja za evidenciju i recenziju filmova. Filmski sadržaj danas predstavlja jedan od najpopularnijih medija i zauzima iznimno velik prostor internetskog prometa. Osim komercijalnih rješenja koje postoje za samo gledanje filmova, tu je još i veliki broj stranica i aplikacija koje su stvorene za interakciju s filmom. One omogućuju korisnicima pregledavanje informacija o filmovima, komentiranje, ocjenjivanje i dodavanje filmova u liste za gledanje. Potreba za ovakvim rješenjima stalno raste zbog količine sadržaja koji korisnici mogu gledati te im nudi mogućnost praćenja što su pogledali, a često im omogućuje i otkrivanje novog sadržaja. Odabir funkcionalnosti koje su uključene u programsko rješenje predstavlja kombinaciju najraširenijih funkcija koje su ugrađene u većinu rješenja koja se bave ovom problematikom. Aplikacija koja je izrađena kao zadatak ovoga završnoga rada, a opisana je u prvom poglavlju, omogućuje korisniku pretraživanje filmova iz online baza podataka uz pomoć API-ja (engl. *Application Programming Interface),* njihovo ocjenjivanje i pisanje recenzija za njih, kao i mogućnost dodavanja filmova na popis filmova koje korisnik nije pogledao, a želi ih pogledati. Korisniku je omogućen pregled detalja o odabranom filmu te pregled vlastite ocjene i komentara kojeg je ostavio za odabrani film. Korisnici mogu vidjeti druge korisnike aplikacije te saznati informacije o njima kao i o filmovima koje su pogledali. Svaki korisnik ima opciju uređivanja svog korisničkog profila i mijenjanja postavki svog računa.

Drugo poglavlje ovoga rada sadrži pregled već postojećih rješenja na tržištu te najbitnijih funkcionalnosti koje ona sadrže i navodi prednosti i nedostatke već postojećih rješenja. U trećem poglavlju su ukratko opisane tehnologije koje su korištene u izradi aplikacije te je objašnjeno zašto su baš one odabrane za izradu aplikacije. Četvrto poglavlje prikazuje postupak izrade same aplikacije kao i objašnjenja značajki aplikacije. Peto poglavlje prikazuje način rada aplikacije i funkcionalnosti koje su dostupne za uporabu. Konačno, u šestom poglavlju dan je zaključak o funkcionalnostima napravljenog rješenja.

1.1. Zadatak završnog rada

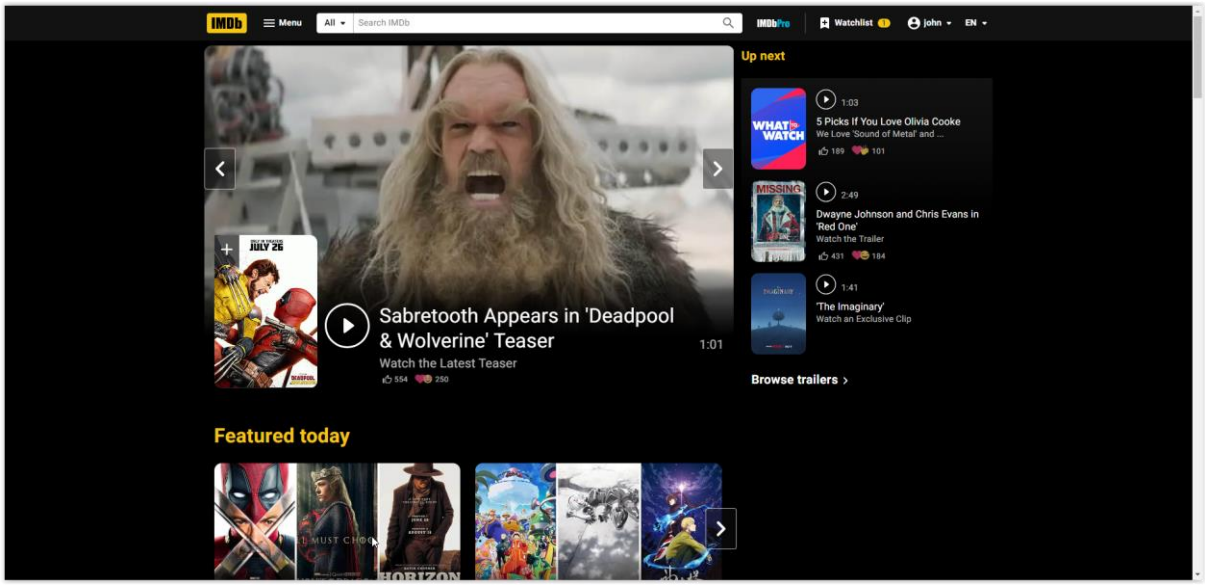
Potrebno je dati pregled funkcionalnosti aplikacije za evidenciju i recenziju filmova. Opisati i izraditi bazu podataka koja će služiti za rad aplikacije. Korisnicima aplikacije omogućiti registraciju i prijavu. Omogućiti pretragu filmova te dodavanje podataka o filmovima na listu pregledanih filmova te filmova koje korisnik želi pogledati. Razviti aplikaciju te objasniti način njezine izrade.

2. PREGLED POSTOJEĆIH RJEŠENJA NA TRŽIŠTU

U ovom poglavlju opisana su neka od najpoznatijih rješenja na tržištu koja rješavaju problem evidencije i recenzije filmova. Većina dostupnih rješenja koja se bave ovom tematikom sastoje se od istih, ili sličnih, funkcionalnosti i mogućnosti, stoga su uzeti u obzir najčešće korišteni pristupi koji se bave recenzijom filmova te su implementirani u programskom rješenju, kao što su mogućnost dobivanja informacija o filmovima, dodavanje filmova u listu pregledanih filmova i filmova koje korisnik želi pogledati te uvid u informacije o drugim korisnicima, poput njihovog profila ili filmova koje su pogledali.

2.1. IMDB

IMDb (engl. *Internet Movie Database*) služi kao internetska baza filmova. Njezine funkcionalnosti predstavljene su na njezinoj početnoj stranici kao što je prikazano na slici 2.1 [1]. Daje prikaz trenutno popularnih filmova, a postavljena je uglavnom tako da funkcionira kao aplikacijsko sučelje koje radi s listama i popisima, kao na primjer popularni top 10 ili top 100.



Sl. 2.1. IMDb stranica [1]

U svojoj izvedbi sadrži brojne funkcionalnosti, ali je njezin osnovni zadatak služiti kao online baza podataka o filmovima i tu svrhu se najčešće i koristi. Njezina baza podataka je vrlo raznovrsna te u sebi sadrži mnogobrojne informacije o samim filmovima poput podataka o redatelju, piscima scenarija, glumcima i dr.

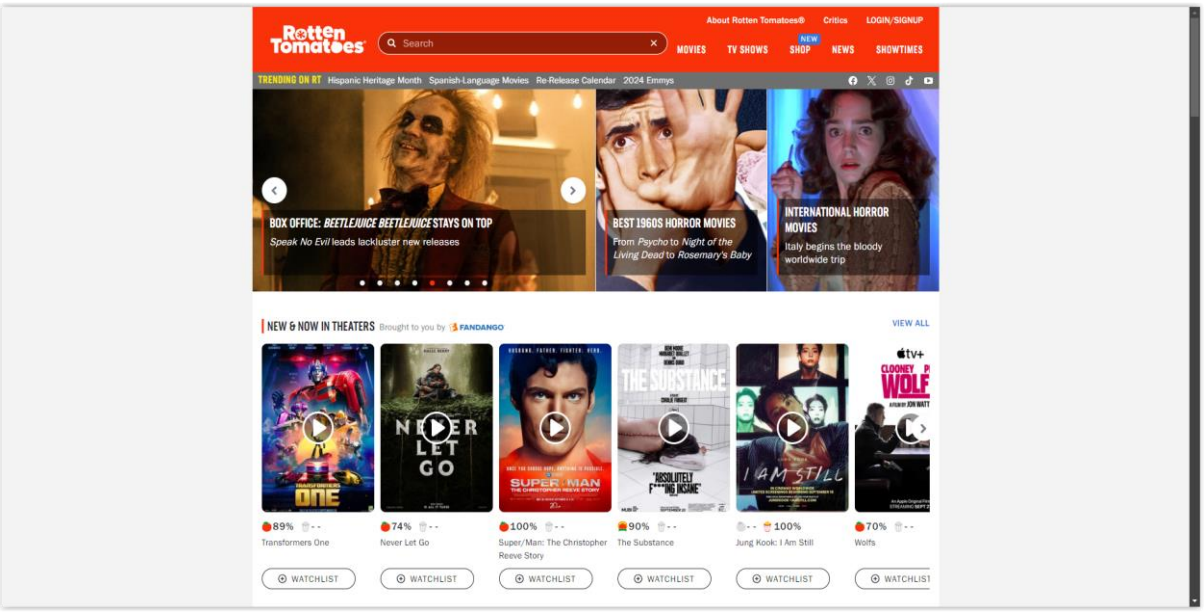
Primarni cilj IMDb-a je imitiranje funkcionalnosti Wikipedije uz sličan način funkcioniranja. Svrha mu je na jednom mjestu objediniti većinu informacija o filmovima kojima svatko može pristupiti. Registrirani korisnici mogu predlagati izmjenu podataka o filmovima. Također, korisnici mogu ocijeniti svaki film ocjenom jedan do deset. Svaki korisnik može dati jednu ocjenu za jedan film, s tim da se svako izmjenjivanje ocjene ne računa kao nova ocjena već se ocjena koju je korisnik već dao mijenja.

Korisnik također može praviti liste filmova po raznim kriterijima te dodavati filmove u popis filmova koje želi pogledati. Također mu je omogućeno ostavljati recenzije na filmove i sudjelovati u anketama.

IMDb-ov sustav za ocjenjivanje je preuzet u izradi aplikacije kao i mogućnost dodavanja filmova u listu pregledanih filmova i filmova koje korisnik želi pogledati. Korisnici ne mogu izmjenjivati informacije o filmovima niti pratiti svoju aktivnost za razliku od IMDb-a. Za API je korišten OMDb (engl. *Open Movie Database)* API koji ima slične funkcionalnosti kao i IMDb-ev API i koristi njegove identifikatore za filmove.

2.2. Rotten Tomatoes

Rotten Tomatoes predstavlja kombinaciju portala za vijesti i aplikacije za vođenje evidencije filmova [2]. Na naslovnoj stranici korisnika dočekuje pregled filmova koji se trenutno prikazuju kao i njihove ocjene uz razne vodiče za gledanje filmova kao što je prikazano na slici 2.2. Na stranici pojedinog filma fokus je primarno na prikazu ocjene toga filma potencijalnim gledateljima.



Sl. 2.2. Stranica Rotten Tomatoes [2]

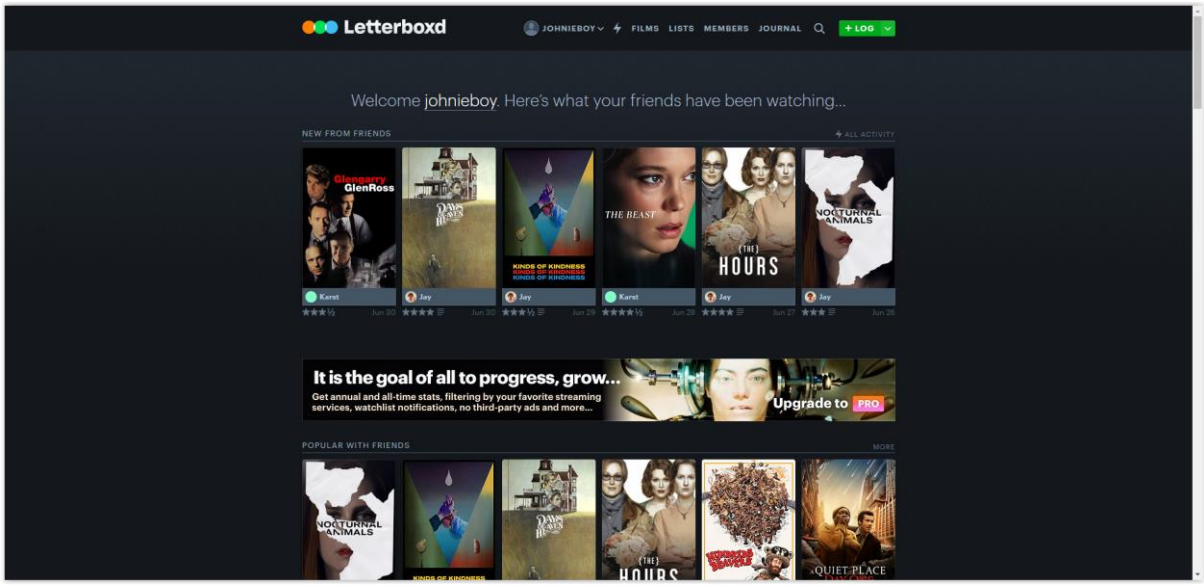
Za razliku od sličnih rješenja, Rotten Tomatoes naglasak stavlja na prikaz dviju ocjena, jedne koju daju korisnici i ocjene koju daju službeni kritičari. Iako na stranici postoje kriteriji po kojima se biraju službeni kritičari, najčešće su to osobe koje se bave filmskom kritikom kao službenim zanimanjem ili su to novinari. Funkcija Rotten Tomatoesa je da imitira filmski vodič za korisnike time što prikazuje osnovne informacije o filmu dok više prostora ostavlja najavama i recenzijama. Korisniku je ostavljeno vrlo malo mogućnosti za upravljanjem svojom evidencijom filmova. Korisnici ih mogu dodati na svoj popis za gledanje te mogu ostaviti ocjenu i recenziju.

Korisnici programskog rješenja mogu ostavljati recenzije i ocjene na filmove kao i na Rotten Tomatoesu, ali ne mogu dobiti pregled recenzija kritičara, što je jedna od najbitnijih značajki Rotten Tomatoesa. Također, korisnici aplikacije ne mogu pregledavati najave o nadolazećim filmovima.

2.3. Letterboxd

Letterboxd se na svojoj internetskoj stranici opisuje kao društvena mreža, ali je uz to naglasak stavljen na vođenje evidencije pregledanih filmova i pronalasku drugih filmova koje bi željeli pogledati [3]. Za razliku od IMDb-a, Letterboxd ne djeluje kao internetska baza podataka o filmovima, a same informacije o filmovima preuzima od TMDb-a (engl. *The Movie Database*).

Letterboxd omogućuje korisnicima pretraživanje filmova i prikazuje im informacije o njima, ali u puno manjem opsegu od IMDb-a. Kada korisnik posjeti stranicu filma, prikazuju mu se filmovi koje su povezani korisnici pogledali te ocjene koje su ostavili, kao što je prikazano na slici 2.3.



Sl. 2.3. Letterboxd stranica [3]

Pronalazak drugih filmova koje bi potencijalno željeli pogledati korisnicima je omogućen uz pomoć prikaza filmova koje su drugi korisnici pogledali te lista koje prikazuju najpopularnije filmove po nekom kriteriju, a koje drugi korisnici izrađuju i uređuju. Drugi korisnici Letterboxda mogu se pratiti kako bi se mogle vidjeti ocjene i recenzije koje su ostavili za filmove koje su pregledali. Uz to, Letterboxd omogućava korisnicima detaljan pregled filmova koje su sami pogledali, recenziju koju su ostavili za taj film i druge funkcionalnosti.

Korisnicima je također omogućen uvid u njihov vlastiti dnevnik koji im prikazuje informacije, poput datuma kad su pogledali film, kad je film izašao, pregled recenzije i ocjene koju su ostavili te im omogućuje uređivanje njihovih ukupnih dojmova o filmu.

Letterboxd pri izradi korisničkog računa od korisnika zahtijeva unos samo adrese e-pošte, korisničkog imena i zaporke. Korisnički profil korisnik može kasnije sam uređivati i dopunjavati te je takav model preuzet u izradi aplikacijskog rješenja.

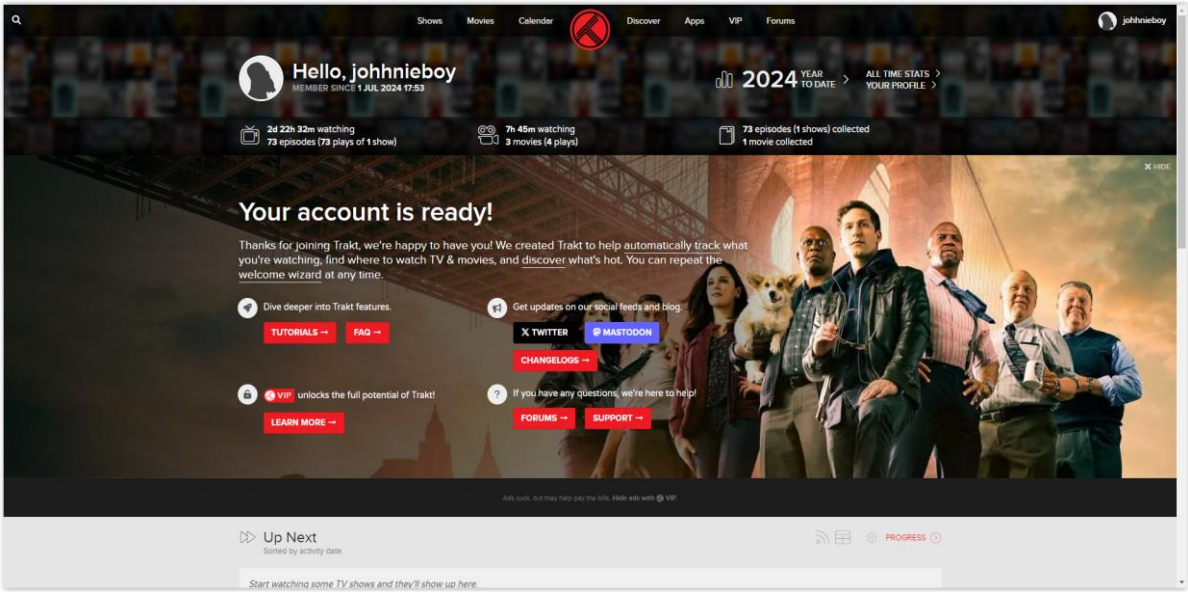
Letterboxd, kao i druge aplikacije, nudi korisniku mogućnost dodavanja filmova u popis pregledanih filmova te dodavanje filmova u popis filmova koje korisnik želi pogledati, no Letterboxd korisniku omogućuje i kreiranje vlastitih popisa, kao primjerice popis najboljih filmova neke godine. Također, nudi informacije o glumcima i redateljima, omogućujući korisniku da i na osnovu tih informacija pronalaze nove filmove koje žele pogledati.

2.4. Trakt

Trakt predstavlja još jedno rješenje za evidenciju pregledanih filmova. Za razliku od drugih stranica, Trakt daje izrazito dobru statistiku o pregledanim filmovima uz mogućnost praćenja i pronalaska drugih filmova i serija koje bi korisnik želio pogledati putem preporuka koje se stvaraju na osnovu filmova koje je korisnik već pogledao, a nudi mogućnost i gledanja sadržaja [4]. Kao i u ostalim stranicama prikazanim u ovom poglavlju, korisnici mogu dodavati filmove na popis pregledanih filmova i filmova koje korisnik želi pogledati, dodavanja u favorite, iako je neke značajke potrebno platiti, na primjer dodavanje privatne recenzije.

Trakt stoga to nadoknađuje statistikom o pregledanim filmovima te je na profilu korisnika moguće vidjeti koliko je vremena korisnik utrošio gledajući neki film, kao što je prikazano na slici 2.4, dok je za serije moguće dobiti informaciju koliko je korisniku bilo potrebno vremena da pogleda cijelu seriju. Kada korisnik označi film kao pogledan, Trakt uzima vrijeme trajanja filma koji je korisnik pogledao i zbraja ga s trajanjem drugih pregledanih filmova kako bi prikazao koliko je vremena

korisnik ukupno potrošio gledajući filmove. Također, prikazana je raspodjela pregledanih filmova po žanrovima.



Sl. 2.4. Trakt stranica [4]

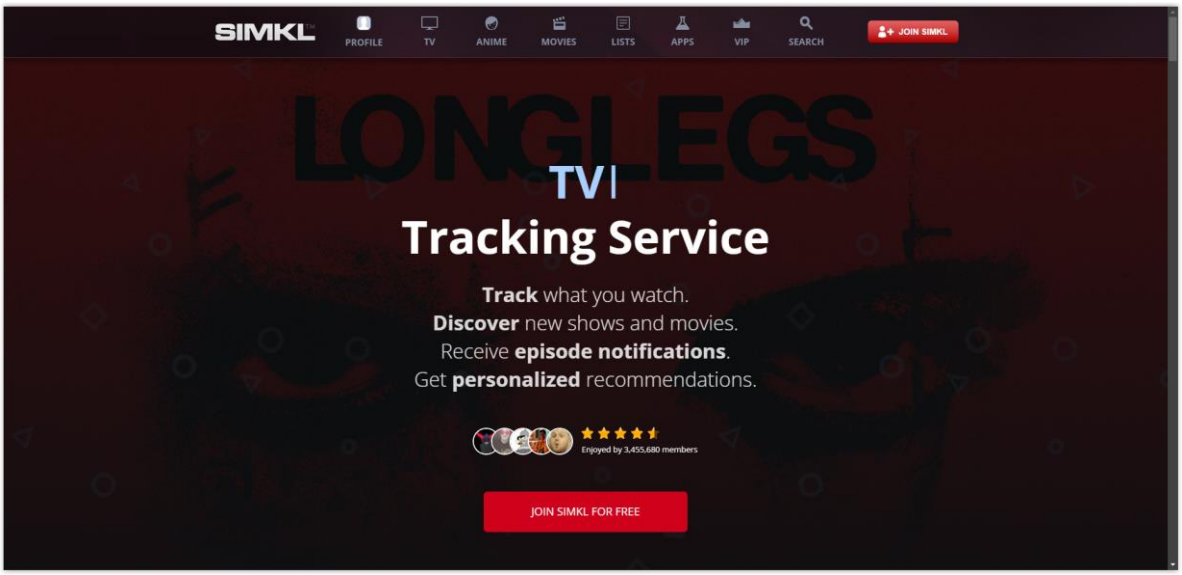
Kao i programsko rješenje, Trakt nudi korisnicima mogućnosti dodavanja filmova u popis filmova koje su pregledali te dodavanje u popis filmova koje žele pogledati, kao i broj pregledanih filmova, no uz to on nudi i puno naprednije mogućnosti poput mogućnosti praćenja ukupnog vremena koje je utrošeno za gledanje filmova te prikaz omiljenih glumaca i redatelja.

2.5. Simkl

Simkl je platforma koja nudi korisniku otkrivanje filmova koje bi željeli pogledati na osnovu filmova koje su već pogledali, vođenje evidencije, kao i još puno drugih značajki [5]. Simkl se od konkurencije pokušava izdvojiti time što u sebi želi objediniti većinu funkcija drugih stranica koje su opisane u ovom poglavlju, pri tom stavljajući naglasak na veliku količinu obavijesti koje korisnik može urediti, što je i oglašeno na njihovoj naslovnoj stranici, kao što se vidi na slici 2.5. Simkl nudi mogućnosti integracije sa streaming platformama poput Kodija i Plexa te na taj način omogućuje korisniku praćenje filmova uz korištenje vanjskih aplikacija.

Kao i u Traktu, korisnicima je dostupna statistika koja pokazuje koliko je filmova korisnik pogledao, koliko je ocijenio, a uz to nudi mogućnost usporedbe svog profila s profilima korisnika na stranici kako bi mogli usporediti što su gledali. Simkl nudi korisnicima sve mogućnosti evidencije pregledanih filmova, kao i programsko rješenje, ali sadrži još neke naprednije

mogućnosti, poput kalendara koji prikazuje izlazak novih filmova i serija te detaljne statistike o pregledanim filmovima, primjerice koliko filmova je pregledano unutar određenog žanra.



Sl. 2.5.Simkl stranica [5]

3. TEHNOLOGIJE KORIŠTENE U IZRADI APLIKACIJE

U ovom poglavlju opisane su tehnologije koje su korištene u izradi programskog rješenja. Objašnjeno je ukratko koje su tehnologije i zašto su odabrane u razvoju ovog programskog rješenja. Za izradu aplikacije korištene su suvremene tehnologije poput Django web okvira, SQLite baze podataka koja obrađuje podatke na poslužiteljskom dijelu, a za prikaz podataka na klijentskom dijelu korišten je HTML (engl. *HyperTextMarkup Language*), CSS (engl. *Cascading Style Sheets*) i Bootstrap.

3.1. Django

Django je web okvir koji se koristi načelima DRY (engl. *Don't Repeat Yourself*) koji naglašavaju korištenje jednom napisanog koda više puta. Pod tim se misli na korištenje već napisanog koda kako bi se obavilo više funkcija u aplikaciji. Django to implementira uz pomoć već gotovih pogleda koji omogućavaju korisniku pisanje više funkcionalnosti korištenjem koda koji je sintaksno sličan. Temelji se na python programskom jeziku te MTV (engl. *Model-Template-View*) arhitekturi koja je zaosnovana na MVC (engl. *Model-View-Controller)* arhitekturi.

Model je u Djangu zadužen za upravljanje podatcima. Pogled korištenjem koda koji je napisan u njemu određuje kojim podatcima krajnji korisnik ima pristup. Za razliku od pogleda, zadaća okvira je prikazati podatke korisniku uz pomoć predložaka i HTML-a. Kako bi se povezali pogledi i predlošci koristi se URL (engl. *Uniform Resource Locator)* konfiguracija. [6]

Django razvojni okvir uzet je za potrebe pisanja aplikacijskog rješenja zbog svoje jednostavnosti i skalabilnosti. On omogućava brzo pisanje koda koristeći već ugrađene poglede i automatski rješava problem rada s bazama podataka pomoću ORM-a (engl. *Object-relational mapper*). On omogućava interakciju s bazom podataka koristeći već ugrađene funkcije umjesto SQL upita.

3.2. SQLite

SQLite je baza podataka koja u sebi sadrži sve pogodnosti SQL (engl. *Structured Query Language)* baze podataka. Korisna je kao rješenje za izradu manjih aplikacija te zapisuje sadržaj baze podataka lokalno na računalu. Uz korištenje Djanga, nema potrebe za administratorom baze podataka sa strane SQLlitea jer tu zadaću preuzima sam Django okvir. [7]

3.3. HTML i CSS

HTML predstavlja prezentacijski jezik za izradu web stranica. Koristi se kao osnovno sredstvo za izradu web stranica. Pomoću HTML-a moguće je odrediti na koji način će se sadržaj na stranici prikazati korisniku, odnosno kako je oblikovan. Na slici 3.1. prikazan je naslov na stranici koji govori da su u nastavku na stranici prikazani podatci o filmu.

```
22        <h1>Podatci o filmu:</h1>
```

Sl. 3.1.Prikaz HTML-a

CSS je jezik koji se koristi za uređivanje HTML elemenata, odnosno omogućuje njihovo stiliziranje te u konačnici upravlja izgledom stranice. Iako je u izradi programskog rješenja primarno korišten Bootstrap, CSS je koristan u slučajevima gdje je potrebno odrediti stil nekog elementa kroz cijeli projekt, a time omogućuje uštedu vremena jer samim tim nije potrebno ručno uređivati elemente na svakoj zasebnoj stranici. Na slici 3.2 prikazan je CSS kod kojim se uređuje *img* element te se sa njim postavlja minimalna širina slike na svakom *img* elementu u aplikaciji. Uz pomoć Django oznake *extends* moguće je primijeniti CSS stil s glavne stranice na sve stranice u aplikaciji gdje je to potrebno.

```
34  img{
35      min-width: 100px;
36  }
```

Sl. 3.2.Primjer CSS-a

3.4. Bootstrap

Bootstrap je popularni okvir koji omogućuje uređivanje internetskih stranica pomoću svojih predložaka. Sadrži veliki broj gotovih predložaka koji se mogu direktno uključiti u kod štedeći vrijeme koje bi bilo utrošeno na uređivanje programskog rješenja. Upravo iz ovog razloga je i odabran za razvoj aplikacije. Na slici 3.3 je prikazano kako je uključivanjem Bootstrapa u HTML elemente moguće upravljati njihovim prikazom.

```
52  <div class="alert alert-warning text-center" role="alert">
53      <br>
54      Krivo korisničko ime ili lozinka!
55  </div>
```

Sl. 3.3. Primjer Bootstrap koda

4. PRIKAZ POSTUPKA IZRADE APLIKACIJE

U ovom poglavlju opisan je postupak izrade programskog rješenja. Razvoj aplikacije objašnjen je uz prikaz programskih blokova i njihovih objašnjenja. Objašnjeno je čemu služi određeni dio koda u aplikaciji, zašto je odabran i na koji način se on uklapa u cjelokupno programsko rješenje.

4.1. Baza podataka

U Django razvojnom okviru model predstavlja ugrađenu funkcionalnost okvira, odnosno predstavlja aplikaciju koja je zadužena za komuniciranje s bazom podataka. Model je klasa, a svaki atribut u njemu ima zadaću predstaviti polje u bazi podataka. Ovakav pristup omogućava korištenje već gotovog sustava za rad s bazom podataka. [8]

Na slici 4.1 prikazan je jedan od modela koji su korišteni u izradi aplikacije.

```
10  class Film(models.Model):
11      user = models.ForeignKey(User, on_delete=models.CASCADE)
12      title = models.CharField(max_length=200)
13      description = models.TextField()
14      release_date = models.DateField(null=True, blank=True)
15      imdb_id = models.CharField(max_length=20)
16      poster_url = models.URLField(blank=True)
17      created = models.DateTimeField(default=timezone.now)
18      rating = models.IntegerField(
19          validators=[MinValueValidator(1), MaxValueValidator(10)],
20          default=5,
21          blank=True,
22          null=True
23      )
24      review = models.TextField(default= "Nije unesena recenzija")
25
26      def __str__(self):
27          return self.title
28
29      class Meta:
30          ordering = ['release_date']
31
```

Sl. 4.1. Prikaz modela Film

Model Film predstavlja stavku filma u aplikaciji. Polje *user* povezan je vanjskim ključem na Djangov model *User*. Vanjski ključ koristi više-na-jedan vezu, a u modelu to bi značilo da jedan film može imati sebi pridruženog jednog korisnika, dok jedan korisnik može imati vezu s više filmova. Polje *title* predstavlja ime filma, u obliku tekstualnog polja s određenom maksimalnom dužinom znakova. Polje *description* predstavlja opis filma, a *TextField* označava kako se radi o dužem tekstualnom polju. Za pohranu vremena koristi se varijabla *release_date* oblika *DateField*, koja može biti prazna i imati vrijednost *NULL* u bazi podataka. URL do postera spremljen je u

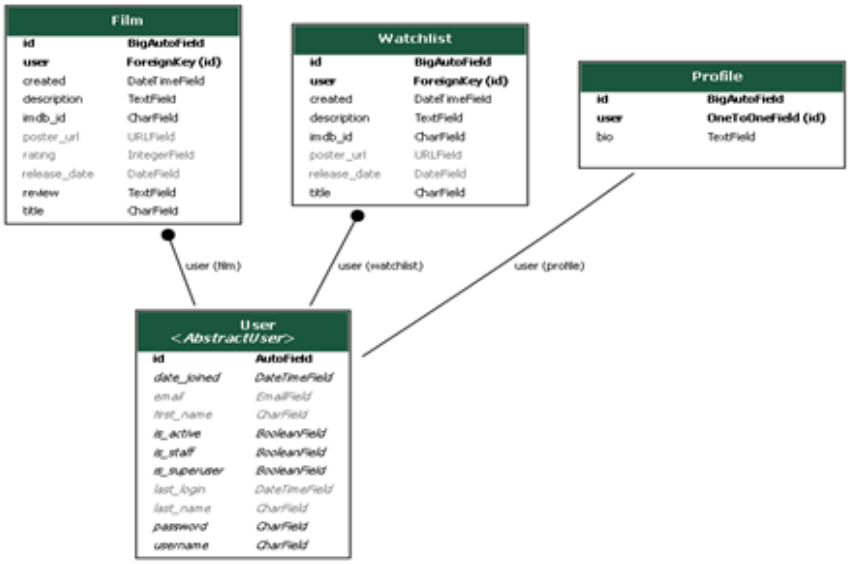
varijabli *poster_url* te također može biti prazan ako ne postoje podatci za određeni film, odnosno URL za njega. Svrha polja *rating* je postaviti vrijednost ocjene, a ako korisnik ne unese ocjenu predefinirana vrijednost je postavljena na pet. Validatori osiguravaju da vrijednost ne može biti manja od jedan i veća od deset. U konačnici, varijabla *ordering* određuje koji zapis iz modela se prvi vraća, u ovom slučaju je to određeno prema datumu izlaska filma. Polja u ovom modelu su mješavina podataka koje korisnik sam unosi i podataka koji se povlače s OMDb-a kroz za to definiran pogled, o čemu će biti riječi dalje u radu.

Slika 4.2 prikazuje model *Profile* koji predstavlja korisnikov profil i povezan je s modelom *User* pomoću *OneToOneField*, što predstavlja vezu jedan-na-jedan. Ovakav odnos osigurava da svaki *User* ima samo jednu *Profile* instancu. Ovaj model se koristi kako bi se dopunio već postojeći model *User* koji je ugrađen u Django. Model *User* ima već predefinirana polja, a kako bi se proširio savjetuje se korištenje novog modela koji će biti povezan s *User* modelom.

```
49  class Profile(models.Model):
50      user = models.OneToOneField(User, null=True, on_delete=models.CASCADE)
51      bio = models.TextField(max_length=400, default="Korisnik još nije unio svoje podatke.")
52
53      def __str__(self):
54          return str(self.user)
55
56      class Meta:
57          ordering = ['user']
58
```

Sl. 4.2. Prikaz modela Film

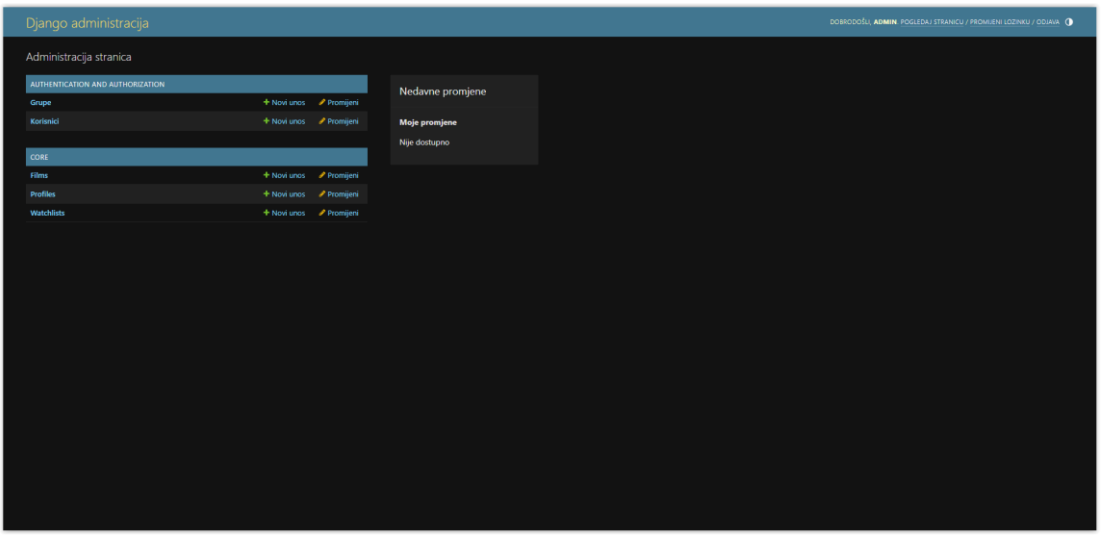
Na slici 4.3 prikazani su svi modeli koji se koriste u aplikaciji. Model *Watchlist* sličan je modelu *Film*, a služi kao zaseban model u kojem korisnik može pohraniti filmove koje želi pogledati.



Sl. 4.3. Prikaz sheme modela

Važna stavka je stvaranje super korisnika. Django u sebi već sadrži administratorsko sučelje, prikazano na slici 4.4, koje omogućava administratoru upravljanje podatcima u aplikaciji. Ova značajka je vrlo korisna jer uklanja potrebu za stvaranjem administratorskih ovlasti u aplikaciji. Korisnik koji ima atribut *is_staff* postavljen na *True* posjeduje pristup administratorskom sučelju, dok svi ostali korisnici koji nemaju ovu ovlast ne mogu pristupiti administratorskom sučelju.

Kako bi se osiguralo prikazivanje administratorskog sučelja u produkcijskom okruženju potrebno je pokrenuti komandu *collectstatic* koja će sakupiti sve potrebne datoteke za prikaz. Kao pomoć u prikazivanju statičkih datoteka u programskom rješenju korištena je python biblioteka Whitenoise, koja omogućuje prikazivanje statičkih datoteka lokalno. [9]



Sl. 4.4. Prikaz Django administratorskog sučelja

4.2. Pogledi

U Djangu pogledi služe kako bi povezali podatke koji se serviraju korisniku s podatcima na poslužiteljskom dijelu. Funkcioniraju tako da uzimaju zahtjev koji je poslao korisnik i sukladno zahtjevu vraćaju odgovor. Pogledi mogu biti u obliku funkcija, o kojima će biti riječi kasnije, i u obliku klasa. *Class-based views* su pogledi koji mogu naslijediti ugrađene Django klase, a posebice su važni zato što postoji znatan broj generičkih pogleda koji obavljaju najčešće funkcionalnosti koje su potrebne u razvoju aplikacija.

Na slici 4.5 prikazan je jedan takav pogled. Radi se o pogledu *MovieList* koji nasljeđuje dvije klase; *LoginRequiredMixin*, i *ListView*. *LoginRequiredMixin* efektivno zahtijeva da korisnik mora biti prijavljen ako želi pristupiti stranici. *ListView* je pogled koji prikazuje podatke iz modela. U ovom se pogledu nalazi funkcija *get_context_data* koja podatke sprema u python rječnik pod imenom *context*. Potom se pomoću ključeva spremaju podatci u rječnik. Prva dva ključa uzimaju podatke iz *Film* modela, a uz pomoć ključa *reviews* se preuzimaju iz modela *Watchlist*.

Ovaj pogled ujedno služi i kao naslovna stranica aplikacije i zadužen je za prikaz svih filmova koje je korisnik označio kao pogledane.

```
182  class MovieList(LoginRequiredMixin, ListView):
183      model = Film
184      context_object_name = 'movies'
185
186
187      def get_context_data(self, **kwargs):
188          context = super().get_context_data(**kwargs)
189          context['movies'] = context['movies'].filter(user=self.request.user)
190          context['count'] = context['movies'].filter(title__isnull=False).count()
191          context['reviews'] = Watchlist.objects.all()
192
193          return context
194
```

Sl. 4.5. Prikaz pogleda *MovieList*

4.2.1. Autentikacija korisnika

Registriranje i odjavljivanje korisnika također koristi poglede temeljene na klasi koji nasljeđuju generičke klase.

Klasa *CustomLoginView*, prikazana na slici 4.6, zadužena je za registraciju korisnika. Nasljeđuje klasu *LoginView* koja u sebi sadrži varijablu *form. Form* varijabla je objekt *AuthenticationForma* te ona u sebi sadrži formu koja provjerava je li unijeto ispravno korisničko ime i zaporka. Forme koje su definirane u pogledima funkcioniraju preko formi na HTML datotekama. Stoga, kada se HTML datoteka učita poziva se GET metoda, a kada korisnik šalje podatke, to radi preko POST metode. *Fields* varijabla govori da su sva polja modela iskorištena. Nakon uspješnog prijavljivanja korisnik biva odveden na URL koji je definiran u *get_success_url* funkciji. U slučaju da se korisnik nije uspio uspješno prijaviti, izbacuje mu se poruka i vraća ga se ponovno na stranicu. Odjava korisnika se obavlja preko *LogoutViewa.*

Klasa *RegisterView* djeluje na sličan način, osim što nasljeđuje *FormView* koji prikazuje formu. Nadalje, naznačuje se da se koristi forma *CreateUser* koja je forma koja je napisana u forms.py datoteci. U njoj se nalaze polja; *username, email, first_name, last_name, password1, password2*. Kada se forma prikaže korisniku, ta polja se nalaze na formi i potrebno ih je popuniti kako bi došlo do uspješne registracije.

```
205  class CustomLoginView(LoginView):
206      template_name = 'core/login.html'
207      fields = '__all__'
208      redirect_authenticated_user = True
209
210      def get_success_url(self):
211          return reverse_lazy('film_list')
212
213      def form_invalid(self, form):
214          messages.error(self.request,'Neispravno korisničko ime ili lozinka.')
215          return self.render_to_response(self.get_context_data(form=form))
216
217
218
219  class RegisterView(FormView):
220      template_name = 'core/register.html'
221      form_class = CreateUser
222      redirect_authenticated_user = True
223      success_url = reverse_lazy('film_list')
224
225      def form_valid(self, form):
226          user = form.save()
227          if user is not None:
228              login(self.request, user)
229          return super(RegisterView, self).form_valid(form)
230
231      def get(self, *args, **kwargs):
232          if self.request.user.is_authenticated:
233
234              return redirect('film_list')
235          return super(RegisterView, self).get(*args, **kwargs)
236
```

Sl. 4.6. Prikaz pogleda za registriranje i odjavljivanje

4.2.2. Upravljanje podatcima

Za prikaz podataka koriste se generički pogledi koji su već uključeni u Django. Osim ranije spomenutog *ListViewa* koji prikazuje podatke iz modela, postoji još nekoliko vrsta poput *TemplateViewa*. Ako je cilj prikazati detaljniji pregled podataka, onda se umjesto *ListViewa* može koristiti *DetailView* pogled. *DetailView* se koristi kada želimo prikazati jedan objekt modela i detaljnije prikazati njegova polja.

Slika 4.7 prikazuje primjer *DetailViewa* pod imenom *UserDetail*. Njegova funkcija je prikazati podatke iz svih modela. Kako bi ispravno dohvatio koji model profila i filma je vezan za određenog korisnika, koristi funkciju *get()*. Kako bi pogled znao o kojem se korisniku radi, koristi se *get_object()* funkcija koja uzima primarni id korisnika te ga upravo preko id-a spaja sa modelima *Profile i Films* .

Korisniku je omogućeno upravljanje svojim korisničkim profilom tako da ima mogućnost dopunjavati svoj profil, mijenjati svoje korisničke podatke, mijenjati zaporku te izbrisati svoj korisnički račun.

```
288  class UserDetail(LoginRequiredMixin, DetailView):
289      model = User
290      template_name = "core/user_detail.html"
291
292      def get_context_data(self, **kwargs):
293          context = super().get_context_data(**kwargs)
294
295          context["users"] = User.objects.all()
296          context["films"] = Film.objects.all()
297
298          user = self.get_object()
299          context["profiles"] = Profile.objects.get(user=user)
300          context["films"] = Film.objects.filter(user=user)
301
302          return context
```

Sl. 4.7. Prikaz *DetailView* pogleda

Na slici 4.8. prikazana su četiri pogleda koja su zadužena za upravljanje korisnikom i njegovim informacijama. Sva tri pogleda koriste ugrađene poglede kako bi manipulirali podatcima o korisniku i njegovom profilu. Pogled *ChangePasswordView* koristi formu *PasswordChangeForm* koja sadrži polja *old_password*, *new_password* i *confirm_password* kako bi korisnik mogao u formi na HTML obrascu izmijeniti svoju zaporku.

```
297  class UserRegisterEdit(LoginRequiredMixin, UpdateView):
298      model = User
299      template_name='core/edit_postavke.html'
300      success_url = reverse_lazy('film_list')
301      fields = ['username', 'first_name', 'last_name', 'email']
302
303      def get_object(self):
304          return self.request.user
305
306  class EditUserProfile(LoginRequiredMixin, UpdateView):
307      model = Profile
308      context_object_name = "edit_user_profile"
309
310      template_name = 'core/edit_profile.html'
311      fields = ['bio', 'profile_picture']
312      success_url = reverse_lazy('film_list')
313
314  class UserDelete(LoginRequiredMixin, DeleteView):
315      model = User
316      context_object_name = 'delete'
317      template_name = 'core/delete.html'
318      success_url = reverse_lazy('film_list')
319
320  class ChangePasswordView(LoginRequiredMixin, PasswordChangeView):
321      form_class = PasswordChangeForm
322      success_url = reverse_lazy('film_list')
323      template_name = 'core/change_password.html'
324
```

Sl. 4.8. Prikaz pogleda za upravljanjem korisnikom

Kao što je već napisano, drugi su oblik pogleda, osim pogleda zasnovanih na klasama, pogledi u obliku funkcija. Funkcije su prvotno jedino bile dostupne u Django okviru, dok nisu uvedeni pogledi zasnovani na klasama kako bi došlo do uštede vremena pri pisanju koda koji se ponavlja. Prednost ovakvih pogleda je što imaju manje ograničenja od pogleda zasnovanih na klasama i omogućuju pisanje bilo kojeg koda koji je potreban kako bi se izvršio određeni zadatak. Svaka funkcija uzima zahtjev upućen od strane korisnika i onda u svom tijelu izvršava određenu logiku.

Na slici 4.9 prikazana je funkcija *search_movies* koja je zadužena za pretraživanje filmova iz OMDb baze. Na OMDb stranici napisano je da se svi zahtjevi za podatcima šalju na ovu internetsku adresu. U URL je potrebno uključiti jedinstveni *api_key* kako bi se znalo o kojem se korisniku radi, odnosno tko šalje zahtjev te termin pretraživanja. Funkcija preko HTML <*input*> oznake prosljeđuje tekst koji korisnik unese kako bi mogla obaviti pretragu. Postavljen je *try-except* blok kojemu je zadaća izbaciti pogrešku ako se logika u *try* bloku ne uspije izvršiti. Ako je zahtjev uspješan, funkcija odgovor pretvara u JSON (engl. *JavaScript Object Notation*) oblik.

```python
def search_movies(request):
    api_key = '681b14eb'
    search_term = request.GET.get('search', 'guardians')  # Default guardians
    url = f"http://www.omdbapi.com/?apikey={api_key}&s={search_term}"

    try:
        response = requests.get(url)
        response.raise_for_status()
        data = response.json()
        movies = data.get('Search', [])

    except requests.RequestException:
        messages.error(request, "Film nije pronađen.")
        movies = []

    return render(request, 'core/search_movies.html', {'filmovi': movies})

```

Sl. 4.9. Prikaz dijela *search_movies* funkcije

Uslijed neuspješnog izvršavanja vraća se informacija o pogreški i prazna lista. Na kraju, funkcija vraća HTML predložak i sve podatke koje je pokupila iz OMDb baze.

Na slici 4.10 prikazan je dio koda *prikazi_film* funkcije koja je zadužena za prikaz detalja određenog filma te njegovo dodavanje u listu pregledanih filmova i listu filmova koje korisnik želi pogledati.

```python
    if request.method == "POST":
        if "add_viewed" in request.POST:
            form = RatingForm(request.POST)
            if form.is_valid():
                if Watchlist.objects.filter(imdb_id=film_imdb_id).exists():
                    Watchlist.objects.filter(imdb_id=film_imdb_id).delete()

                film, created = Film.objects.get_or_create(
                    imdb_id=film_imdb_id,
                    user=request.user,
                    defaults={"user": request.user, "title": film_title, "description": film_description, "release_date": film_release_date, "poster_url": poster_url,
                    },
                )

                film.rating = form.cleaned_data["rating"]
                film.review = form.cleaned_data["review"]
                film.save()

                messages.success(request, "Uspješno ste dodali ocjenu i recenziju!")
                return redirect(
                    "prikazi", imdb_id=imdb_id
                )

        elif "add_watchlist" in request.POST:
            if Film.objects.filter(imdb_id=film_imdb_id, user=request.user).exists():
                messages.info(
                    request, "Ovaj film se već nalazi u vašoj listi pogledanih filmova."
                )
            elif Watchlist.objects.filter(
                imdb_id=film_imdb_id, user=request.user
            ).exists():
                messages.info(request, "Ovaj film se već nalazi u vašoj watchlisti.")

            else:
                film, created = Watchlist.objects.get_or_create(
                    imdb_id=film_imdb_id,
                    user=request.user,
                    defaults={"user": request.user, "title": film_title, "description": film_description, "release_date": film_release_date, "poster_url": poster_url,
                    },
                )
                messages.info(request, "Uspješno ste dodali film u watchlistu.")

        return redirect("prikazi", imdb_id=imdb_id)
```

Sl. 4.10. Prikaz dijela *prikazi_film* funkcije

Ove funkcionalnosti su napravljene u jednoj funkciji kako bi se koristilo *don't repeat yourself* načelo i kako bi sve funkcionalnosti bile smještene na jednom mjestu. Ovisno koji gumb korisnik klikne, obavlja se jedan od dva uvjeta.

Ukoliko korisnik doda film u listu filmova koje želi pogledati, a film koji želi dodati je već označen kao pogledan, tada mu se prikazuje poruka da se film već nalazi u listi pregledanih filmova. U slučaju da se film, odnosno podatci o filmu već nalazi u modelu *Watchlist* tada korisnik biva obaviješten da se film već nalazi u toj listi, odnosno modelu. Ako ne postoji, pomoću funkcije *.get_or_create* film se zapisuje u model.

4.2.3. Upravljanje URL-ovima

Django URL-ovi se nalaze u *urls.py* datoteci. Njihov zadatak je povezivanje zahtjeva koje korisnik šalje s klijentske strane prema pogledima. Kada se zahtjev pošalje, odabire se onaj pogled koji odgovara traženom URL-u. Slika 4.11. prikazuje sve definirane URL obrasce za programsko rješenje.

```
15  urlpatterns = [
16      path("login/", CustomLoginView.as_view(), name="login"),
17      path("register/", RegisterView.as_view(), name="register"),
18      path("users/", ViewUsers.as_view(), name="users"),
19      path("user/<int:pk>/", UserDetail.as_view(), name="user"),
20      path("edit-profile/<int:pk>", EditUserProfile.as_view(), name="edit_user_profile"),
21      path("edit-postavke/<int:pk>", UserRegisterEdit.as_view(), name="user_register_edit"),
22      path("delete-user/<int:pk>/", UserDelete.as_view(), name="user_delete"),
23      path("change-password/", ChangePasswordView.as_view(), name="password"),
24      path("watchlist/", WatchList.as_view(), name="watch_list"),
25      path("add_to_watchlist/", WatchList.as_view(), name="add_to_watch_list"),
26      path("logout/", LogoutView.as_view(next_page="login"), name="logout"),
27      path("", MovieList.as_view(), name="film_list"),
28      path("movie/<int:pk>/", MovieDetail.as_view(), name="film_detail"),
29      path("delete/<int:pk>/", MovieDelete.as_view(), name="film_delete"),
30      path("delete-watchlist/<int:pk>/", WatchlistDelete.as_view(), name="delete_watchlist"),
31      path("search-movies/", views.search_movies, name="search_movies"),
32      path("film/<str:imdb_id>/", views.prikazi_film, name="prikazi"),
33  ] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Sl. 4.11. Prikaz URL-ova

Prvi dio *path()* funkcije sadrži URL na koji korisnik može pristupiti u web pregledniku. Moguće je dodati dodatne argumente poput *<int:pk>* koji govori pogledu kojoj instanci modela treba pristupiti u slučaju da, primjerice, korisnik želi pristupiti profilu drugog korisnika. Funkcija *as_view()* vraća odabrani pogled, varijabla *name* označava ime tog URL obrasca.

4.2.4. Predlošci

Django predlošci su datoteke koje u sebi sadrže HTML i DTL-a (engl. *Django template language*) koji je posebna sintaksa koja se koristi za prikaz dinamičkog sadržaja. Gotovo svaka HTML

datoteka ima bar dio koda napisan sa DTL-om kako bi mogao prikazati najčešće nekakve podatke. DTL daje mogućnost pisanja i koda za kontrolu tijeka pa je tako moguće prikazivati određene HTML elemente ako je neki uvjet ispunjen. Jedna od stavki koju korištenje predložaka omogućuje je nasljeđivanje svojstava iz jedne HTML datoteke u drugu [10]. Prema slici 4.12 može se vidjeti da određeni HTML dokument nasljeđuje *main.html* dokument, a s njim i elemente *main.html* dokumenta.

```
1   {% extends 'core/main.html' %}
2   {% block content %}
```

Sl. 4.12. Prikaz Django *extends* oznaka

Slika 4.13 prikazuje dio koda koji se koristi u predlošku *film_list.html*. Datoteka predstavlja kombinaciju običnog HTML-a, Bootstrapa i Djangovih oznaka.

```
13      <div class="container-fluid">
14          <br>
15          <h1 class="text-center">Stranica za recenziju filmova</h1>
16          <hr>
17      </div>
18      {% if request.user.is_authenticated %}
19      {% else %}
20      <a href="{% url 'login' %}">Login</a>
21      {% endif %}
22      <div class="header-bar">
23          <div class="text-center">
24              <h1> Dobrodošli {{request.user|title}}!</h1>
25              <br>
26              <h3 style="margin:0">Broj pogledanih filmova: {{count}}</h3>
27          </div>
28      </div>
29      <div class="container-fluid" id="overflow" >
30      <table id="example" class="table table-striped align-middle text-center" style="width:100%" >
31          <thead>
32              <tr class="text-center">
33                  <th scope="col" class="text-center">Ime</th>
34                  <th scope="col" class="text-center">Datum izlaska</th>
35                  <th scope="col" class="text-center">Ocjena</th>
36                  <th scope="col" class="text-center">Recenzija</th>
37                  <th scope="col" class="text-center">Slika</th>
38                  <th scope="col" class="text-center">Pogledaj</th>
39                  <th scope="col" class="text-center">Obriši</th>
40              </tr>
41          </thead>
42          <tbody>
43              {% for film in movies %}
44              <tr>
45                  <th scope="row">{{film.title}}</th>
46                  <td  class="text-center"style="width:15%">{{ film.release_date|date:"Y-m-d" }}</td>
47                  <td  class="text-center">{{film.rating}}</td>
48                  <td  class="text-center">{{film.review}}</td>
49                  <td><img  src="{{film.poster_url}}" alt="" class="img-thumbnail"></td>
50                              <td><a class="btn btn-primary" href="{% url 'prikazi' film.imdb_id %}"><i
51                                  class="bi bi-eye-fill"></i>Pogledaj</a></td>
52                  <td>
53                      <a href="{% url 'film_delete' film.id %}" class="btn btn-danger">
54                          <i class="bi bi-trash"></i>Obriši
55                      </a>
56                  </td>
57              </tr>
58              {% empty %}
59              <br>
60          <div class="alert alert-info text-center d-flex justify-content-center" style="max-width: 500px; margin: 0 auto;"  role="alert">
61              <h6 class="align-middle text-center">Nemate filmova u bazi!</h6>
62          </div>
63              {% endfor %}
64          </tbody>
65      </table>
66  </div>
```

Sl. 4.13. Dio HTML-koda *film_list.html* datoteke

Oznake se pišu sa *{% object %}* i koriste se za izvođenje nekakve logike, kao u bloku gdje se provjerava je li korisnik prijavljen te kako bi se korisnika proslijedilo na *login.html* stranicu ukoliko nije prijavljen. Uz oznake, postoje i varijable koje služe primarno za prikazivanje podataka na stranici i pišu se u obliku *{{variable}}*. U varijabli *{{film.title}}* može se preko varijable *film* dobiti naslov filma. Budući da je pogledu definirano da prikaže sve podatke iz modela Film, pomoću *for* petlje može se proći kroz kolekciju objekata i prikazati svaku stavku posebno. Prednost korištenja Django oznaka je u tome što ovisno o uvjetu postavljenom u njima može se prikazati neki dio HTML-a ako je uvjet ispunjen, dok u suprotnom taj dio koda neće biti prikazan. Uz pomoć *<a>* elementa i oznake za prikaz filma korisnik ima mogućnost posjetiti druge stranice u sklopu aplikacije.

U ovom kodu cilj je prikazati tablicu uz pomoć HTML-a i Django oznaka za svakog korisnika. Na ovaj način na početnoj stranici aplikacije korisnik u tabličnom obliku može vidjeti sve filmove koje je označio kao pregledane uz njihovo ime, datum izlaska, poster, recenziju i ocjenu. Ukoliko korisnik nema filmova u svojoj bazi prikazuje se *div* element u kojem korisnik biva obaviješten kako nema filmova u bazi.

Kako bi se poboljšao pregled tablice, korištena je vanjska biblioteka *Datatables* [11]. *Datatables* je HTML tablica koja uz pomoć Javascripta dodaje dodatne funkcionalnosti tablicama poput pretraživanja, razvrstavanja, postavljanja koliko unosa će biti po stranici i drugih značajki. Odlučeno je koristiti *Datatables* jer se na jednostavan način donese puno funkcionalnosti kako bi se korisniku pružilo ugodnije iskustvo u radu s aplikacijom. Stupci Recenzija, Slika, Pogledaj i Obriši su uz pomoć Javascript koda ograničeni da se ne mogu razvrstavati, što je još jedna od značajki koje *Datatables* biblioteka pruža.

Kako bi se moglo prikazati više podataka iz različitih modela, potrebno je koristiti više varijabli. Prema slici 4.14 varijabla s objektom *user* sadrži atribut *username* koji prikazuje ime korisnika. Budući da se radi o *user_detail.html* predlošku koji je vezan s *UserDetail* pogledom, može se prikazati podatke iz tri modela. Ovaj pristup je koristan kada se želi prikazati više podataka iz različitih modela, kao što je stranica koja prikazuje podatke o pojedinom korisniku.

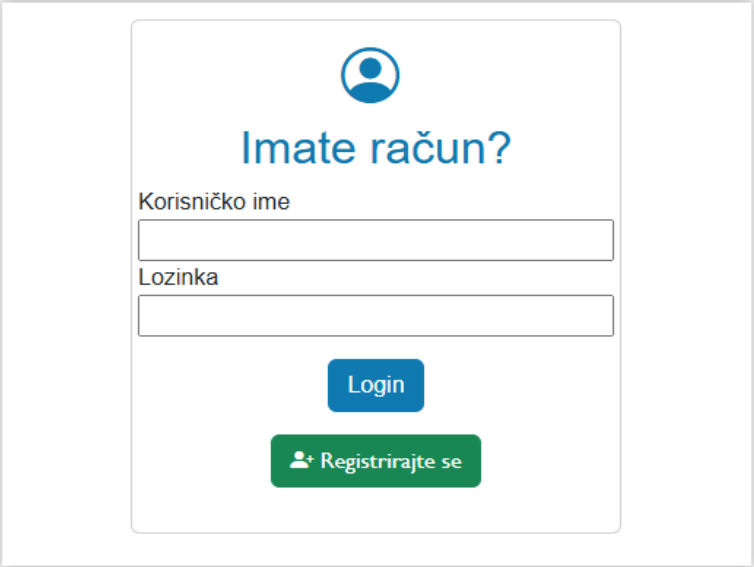
U istoj HTML datoteci koriste se oznake kako bi se dodala logika koja ispituje je li korisnik na svojoj stranici, odnosno na svom *DetailViewu*. Ako jest, korisniku se prikazuju <a> elementi preko kojih korisnik može pristupiti pogledima za upravljanje podatcima.

```
15  <div class="d-flex justify-content-center">
16    <div class="card mb-6 col" style="max-width:500px;">
17      <div class="card-header">
18        {{ user.username }} | {{object.first_name}} {{object.last_name}}
19      </div>
20      <div class="row g-0">
21        <div class="col-md-4">
22          <img src="https://robohash.org/{{ user.username }}" alt="Profile Picture" width="200" height="200"
23            class="img-fluid rounded-start">
24        </div>
25        <div class="col-md-8">
26          <div class="card-body">
27            <h5 class="card-title">Email: {{object.email}}</h5>
28            <p class="card-text"><strong>Bio:</strong> {{profiles.bio}}</p>
29            <p class="card-text"><small class="text-muted">Pogledanih filmova: {{films.count}}</small></p>
30          </div>
31        </div>
32      </div>
33    </div>
34  </div>
```

Sl. 4.14. Dio HTML-koda *user_detail.html* datoteke

5. PRIKAZ RADA WEB APLIKACIJE

U ovome poglavlju govori se o načinu rada aplikacije i funkcionalnostima koje ona posjeduje. Nakon pokretanja aplikacije korisnika dočekuje zaslon za prijavu na kojem se može prijaviti ako je već registrirani korisnik, kao što je vidljivo na slici 5.1. Ukoliko korisnik nije registriran, može klikom na gumb *Registrirajte se* otići na stranicu za registraciju. Ukoliko korisnik unese krivo korisničko ime ili zaporku, prikazuje mu se poruka o tomu.

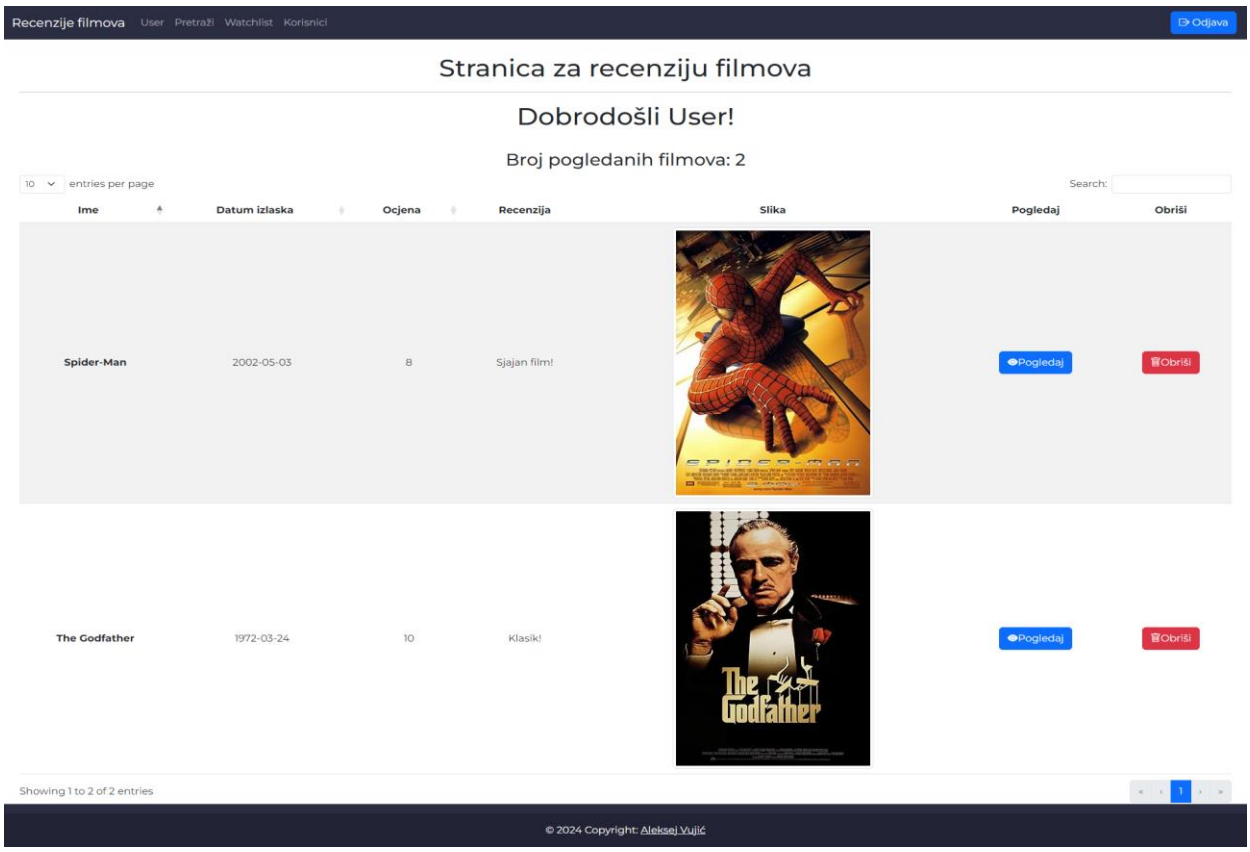


Sl. 5.1. Prikaz stranice za prijavu

Ovisno o već postojećim rješenjima na tržištu, određene aplikacije od korisnika zahtijevaju unos samo korisničkog imena ili adrese e-pošte, dok druge od korisnika zahtijevaju unos dodatnih podataka kao što su ime, prezime i drugo. Kako bi se korisnik uspješno registrirao, od njega se traži unos osnovnih informacija koje su prikazane na slici 5.2. Ostale informacije može unijeti naknadno.

Sl. 5.2. Prikaz stranice za registraciju

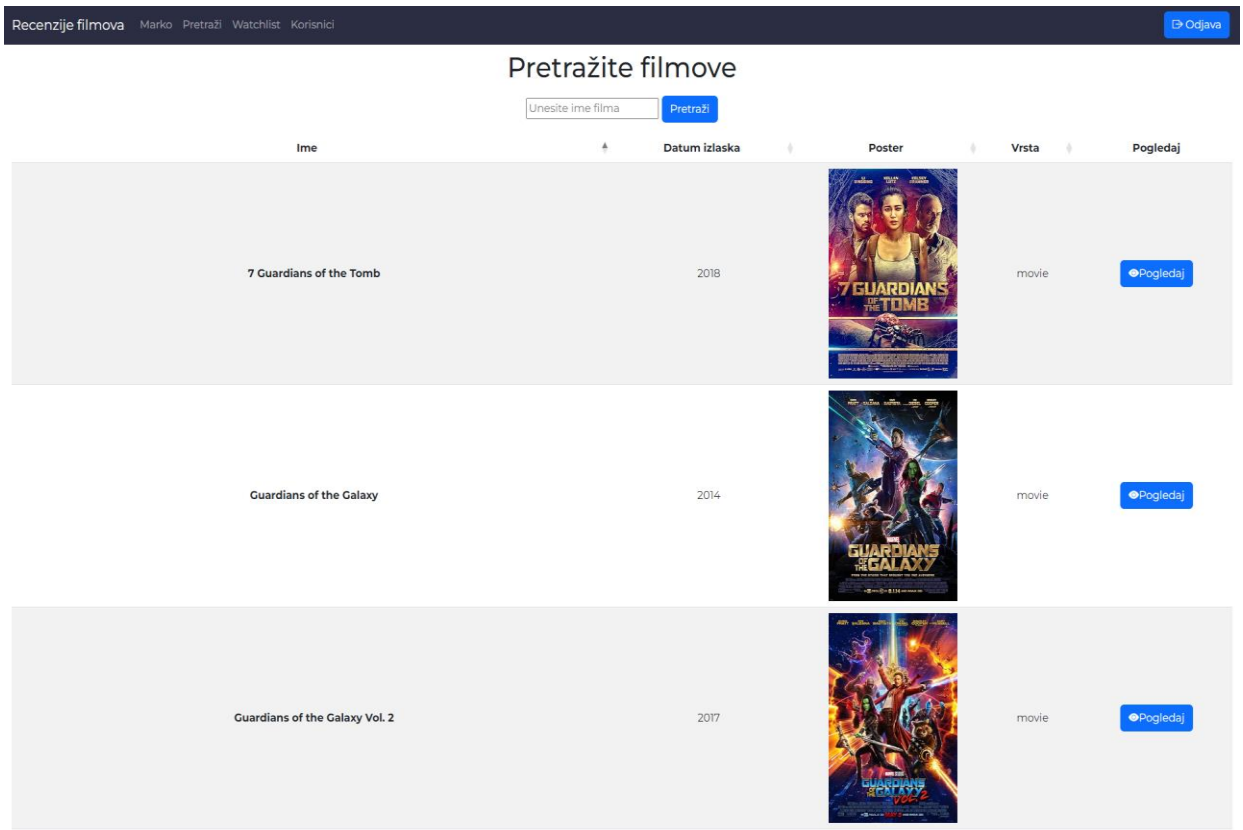
Nakon uspješne registracije, korisnik biva proslijeđen na početnu stranicu aplikacije koja je prikazana na slici 5.3. Kako je aplikacija namijenjena vođenju evidencije pregledanih filmova, korisniku se prikazuju filmovi koje je označio kao pogledane s najbitnijim informacijama o njima. Korisnik također može vidjeti svoju ocjenu i recenziju koju je unio tijekom dodavanja filma u popis pregledanih filmova.



Sl. 5.3. Prikaz naslovne stranice

Na navigacijskoj traci korisnik ima opciju otići na svoj korisnički profil, pretraživati filmove kako bi ih dodao u popis pregledanih filmova ili u popis filmova koje namjerava pogledati te pogledati sve korisnike.

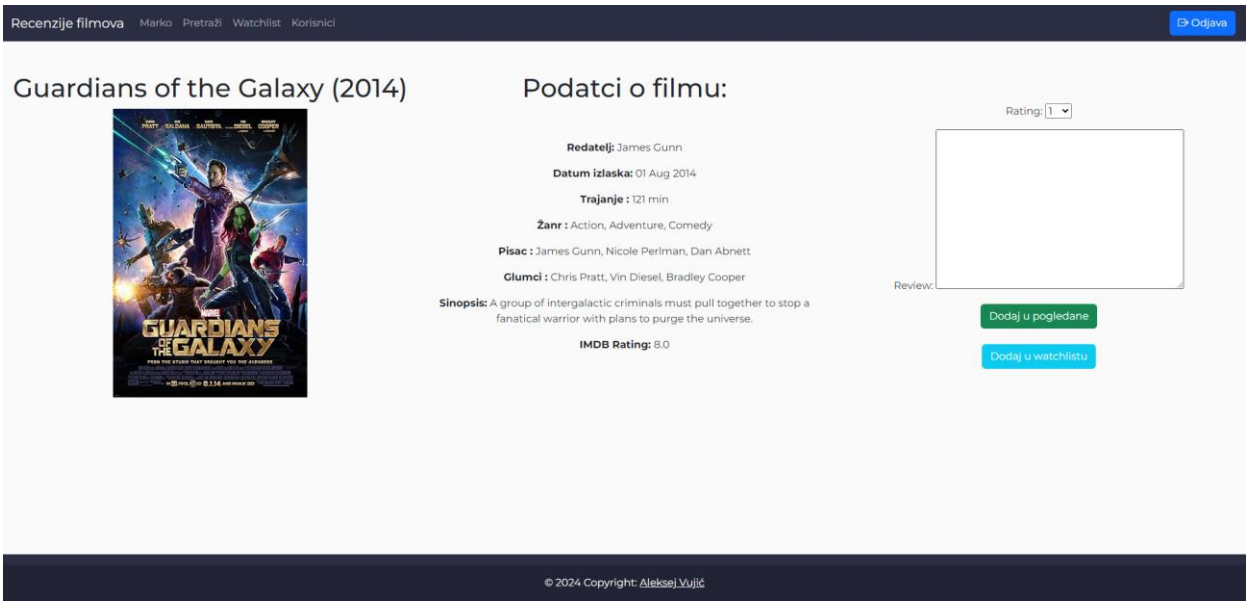
Ako korisnik želi pretraživati filmove, može kliknuti na poveznicu *Pretraži* koja ga odvodi na stranicu za pretraživanje filmova, kao što se može vidjeti na slici 5.4. Na stranici uvijek budu već učitani filmovi, a unosom imena željenog filma i klikom na gumb *Pretraži* korisnik dobiva popis filmova koji odgovaraju traženom pojmu. Na ovoj stranici korisnik može vidjeti ime, datum izlaska, poster i vrstu filma, a klikom na gumb *Pogledaj* otvara mu se zaseban prozor koji pokazuje informacije o tom filmu.



Sl. 5.4. Prikaz stranice za pretraživanje

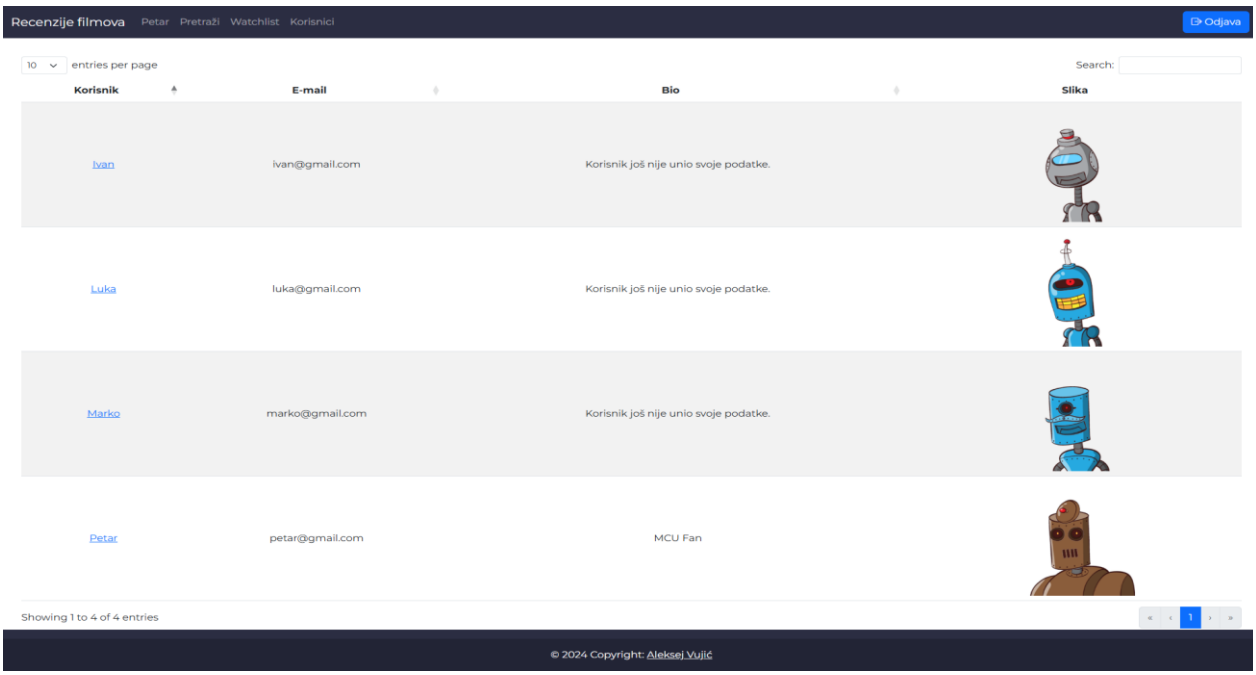
Slika 5.5 prikazuje stranicu pojedinog filma. Ona nudi prikaz osnovnih podataka o filmu kao što su redatelj, datum izlaska, trajanje, žanr, poster i drugo. S desne strane su dva gumba s kojima korisnik ima opciju označiti filmove kao pogledane ili ih spremiti u listu filmova koje želi pogledati. Ukoliko film spremi u listu filmova koje želi pogledati, dobiva obavijest da je film uspješno dodan u prijašnju listu. Ukoliko korisnik klikne na gumb *Dodaj* u pogledane, a da pritom nije dodao ocjenu i recenziju, dobiva upozorenje kako mora popuniti polje. Ako je ispunio polja i

kliknuo *Dodaj u pogledane*, čeka ga obavijest o tome i film više neće biti na stranici filmova koje korisnik želi pogledati, već na naslovnoj stranici.



Sl. 5.5. Stranica za prikaz pojedinog filma

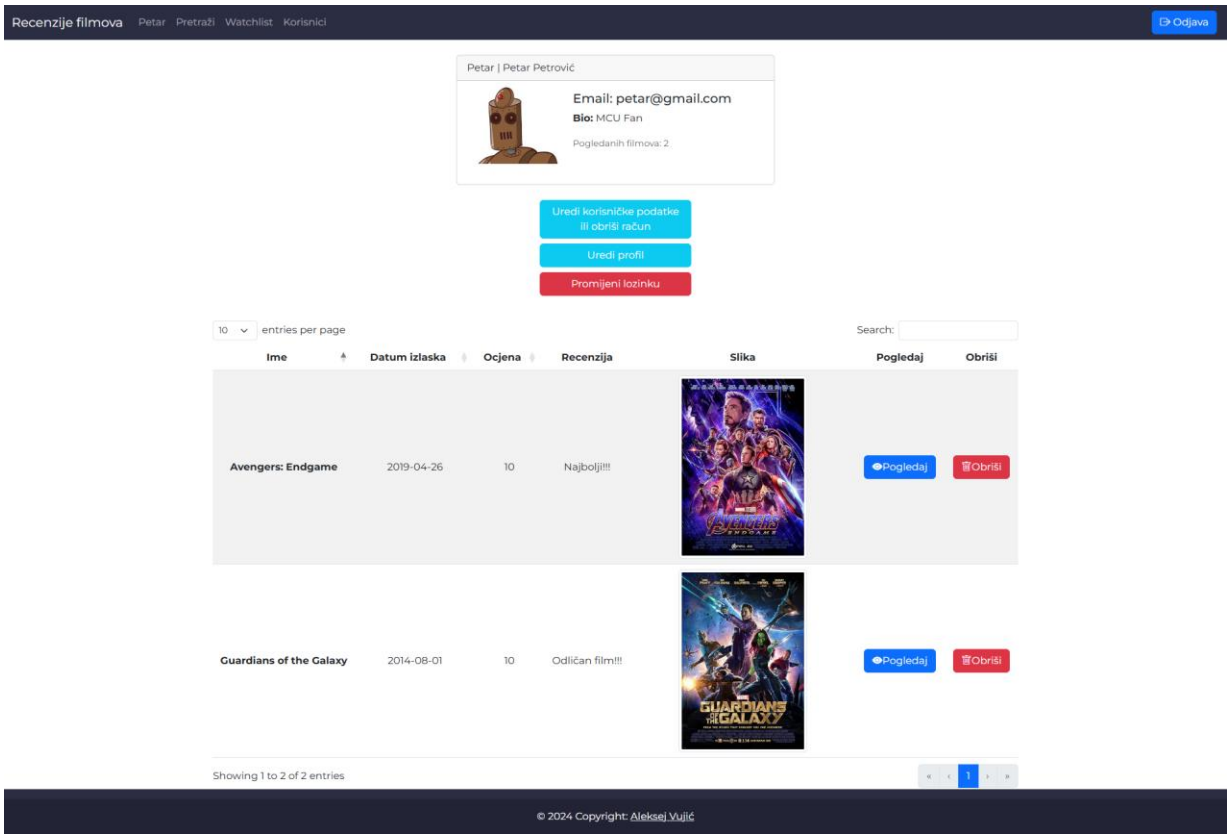
Slika 5.6 prikazuje sve korisnike, a na nju se može dospjeti klikom na poveznicu *Korisnici*. Na ovoj stranici dat je pregled svih korisnika aplikacije s njihovim korisničkim imenom, adresom e-pošte, opisom i slikom. Slika korisnika se automatski generira pomoću njegovog korisničkog imena, a to je učinjeno uz pomoć stranice Robohash [12], koja omogućava stvaranje jedinstvenih slika na osnovu teksta koji joj se proslijedi.



Sl. 5.6. Stranica za prikaz svih korisnika

Moguće je pretraživati korisnike, kao i razvrstavati ih. Klikom na ime drugog korisnika, trenutni korisnik biva odveden na profil tog korisnika.

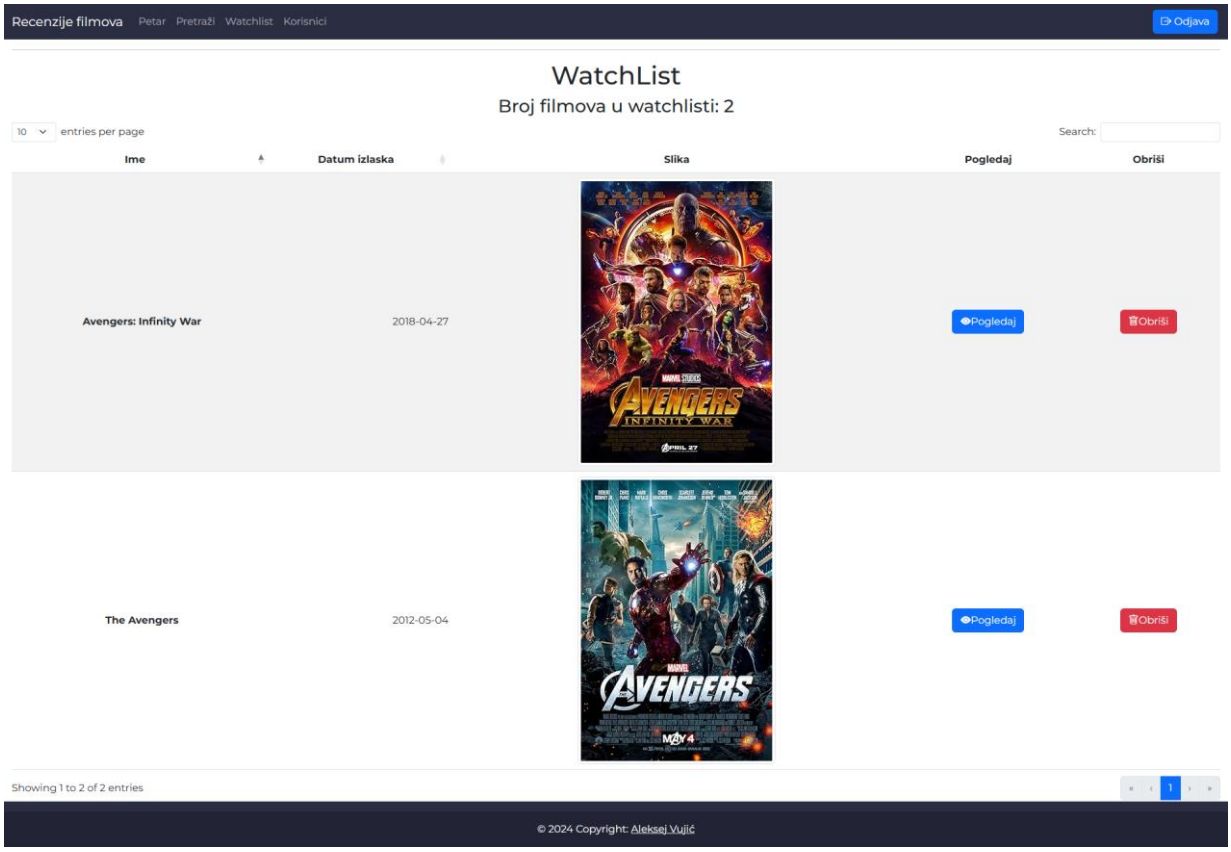
Korisnikovo ime u navigacijskoj traci služi kao prikaz trenutno prijavljenog korisnika, a klikom na gumb biva proslijeđen na svoj korisnički profil, kao što je prikazano na slici 5.7.



Sl. 5.7. Prikaz korisnikovog profila

Na ovoj stranici korisnik može u kartici vidjeti informacije o sebi, svoj opis, adresu e-pošte, ime i prezime. Moguće je dobiti uvid u filmove koje je korisnik pogledao te ih je moguće pretraživati i razvrstavati. U kartici je prikazana korisnikova slika koja je ista kao i na prikazu svih korisnika upravo jer se slika stvara na temelju korisnikovog korisničkog imena. Ako je korisnik na stranici svog profila, prikazuju mu se tri gumba kojima može upravljati svojim profilom. Ako je korisnik na stranici nekog drugog korisnika, ta tri gumba neće mu biti prikazana. U ovom prozoru moguće je dobiti uvid u to koje je filmove odabrani korisnik pogledao te će uz njih u tablici biti prikazani ocjena i recenzija koje je odabrani korisnik ostavio na film. U samoj kartici je prikazana korisnikova slika, korisničko ime, ime i prezime, adresa e-pošte, broj pregledanih filmova te kratki opis o njemu. Ukoliko korisnik želi dobiti prikaz filmova koje želi pogledati, to može učiniti odlaskom na stranicu *Watchlist* koja je prikazana na slici 5.8. Na ovoj stranici prikazuju mu se

filmovi koje želi pogledati, slično kao i na glavnoj stranici. Na ovoj stranici korisniku neće biti prikazana ocjena i recenzija koje je unio, za razliku od naslovne stranice.



Sl. 5.8. Prikaz filmova koje korisnik želi pogledati

6. ZAKLJUČAK

U ovom završnom radu prikazan je razvoj web aplikacije za evidenciju i recenziju filmova. Prikazane su tehnologije koje su korištene tijekom razvoja aplikacije te je ujedno i objašnjeno zašto se pojedina tehnologija koristi. Sve funkcionalnosti koje većina komercijalnih rješenja posjeduje prisutne su i u ovoj aplikaciji. Pozornost je posvećena tomu da aplikacija elegantno izgleda kako bi se izbjegla vizualna prenatrpanost te se težilo postići jednostavnu navigaciju kroz stranicu. Korisnik programskog rješenja može dodavati filmove u svoju listu pregledanih filmova, a filmove koje još nije pogledao može dodati u listu za gledanje. Sve filmove u listi je moguće razvrstavati i pretraživati kako bi se korisniku pružilo jednostavnije i intuitivnije korištenje aplikacije. Moguće je pogledati detalje o traženom filmu kao i pretraživati druge korisnike aplikacije. Korisniku je dana mogućnost upravljanja svojim profilom i postavkama svog računa.

Programsko rješenje je moguće naknadno poboljšati dodavanjem dodatnih funkcionalnosti poput opcije izrade posebnih lista u koje korisnik može dodavati filmove ovisno o određenom parametru koji odabere. Jedna od značajki koje bi se mogle inkorporirati u aplikaciju je mogućnost implementacije kalendara kako bi korisnik mogao imati bolji prikaz filmova koji će izaći te kako bi korisnikovo vođenje evidencije moglo biti još detaljnije.

LITERATURA

[1] IMDb, IMDb.com, Inc., [Online]., dostupno na: https://www.imdb.com/. [15.9.2024].

[2] Rotten Tomatoes, Fandango, [Online]., dostupno na: https://www.rottentomatoes.com/. [15.9.2024].

[3] Letterboxd, Letterboxd Limited, [Online]., dostupno na: https://letterboxd.com/. [15.9.2024].

[4] Trakt, Trakt, Inc., [Online]., dostupno na: https://trakt.tv/. [1.7.2024].

[5] Simkl, Simkl, Inc., [Online]., dostupno na : https://simkl.com/. [15.9.2024].

[6] W. S. Vincent, Django for Beginners; Build Websites with Python & Django, WelcomeToCode, 2022.

[7] D. R. Hipp, SQLite, Hipp, Wyrick & Company, Inc., [Online]., dostupno na: https://www.sqlite.org/index.html. [1.7.2024].

[8] Django Documentation, Django Software Foundation, [Online]., dostupno na: https://docs.djangoproject.com/en/5.1/topics/db/models/. [1.7.2024].

[9] D. Evans, Whitenoise documentation, WhiteNoise Project, [Online]., dostupno na: https://whitenoise.readthedocs.io/en/latest/. [3.11.2024].

[10] L. L. Lazzaro, Ultimate Django for Web App Development Using Python, Delhi: Orange Orange Education Pvt Ltd, 2024.

[11] Datatables, SpryMedia Ltd., [Online]., dostupno na: https://datatables.net/. [15.9.2024].

[12] Robohash, RoboHash.org, [Online]., dostupno na: https://robohash.org/. [1.11.2024].

SAŽETAK

U današnjem svijetu gdje se broj sadržaja koje je moguće konzumirati svakodnevno povećava, javlja se potreba za alatima koji mogu u tome pomoći. Iako mnoga rješenja za problematiku vođenja evidencije i recenzije filmova već postoje, varijacije u detaljima među njima mogu činiti veliku razliku nekim korisnicima. Stoga uvijek postoji potreba za dodatnim rješenjima na tržištu.

Cilj ovoga završnoga rada bio je prikazati postupak izrade programskog rješenja za traženi problem, uz objašnjenje tehnologija koje su korištene, kao i razloge zbog kojih su odabrane. U razvoju aplikacije korištene su tehnologije Django, Bootstrap, HTML te CSS. Aplikacija omogućava korisniku izradu korisničkog računa te prijavu i odjavu. Korisnik može pretraživati filmove i vidjeti podatke o njima. Filmove koje je već pogledao može dodati u listu pregledanih filmova. Filmove koje želi pogledati može dodati u zasebnu listu. Korisniku je dan prikaz drugih korisnika aplikacije i dan mu je pregled filmova koje su pogledali. Svaki korisnik može promijeniti svoje korisničke podatke.

Ključne riječi: aplikacija, Django, evidencija, filmovi

ABSTRACT

Web application for keeping movie notes and reviewing movies

In today's world, where the amount of content that can be consumed increases daily, there is a need for tools that can aid in solving this problem. Although many solutions to the problem of record keeping and reviewing movies already do exist, the variations in detail between them can make a big difference to some users. Therefore, there is always a need for additional solutions on the market.

The goal of this final thesis was to present the process of creating a software solution for the requested problem, along with an explanation of the technologies that were used, as well as the reasons why they were chosen. The application was developed using technologies such as Django, Bootstrap, HTML and CSS. The application allows the user to create a user account, as well as log in and log out. The user can search for movies and see information about them. The user can add movies he has already watched to the list of watched movies. He can add the movies he wants to watch to a separate list. The user is given a view of other users of the application and is given an overview of the movies they have watched. Every user can change their user data.

Keywords: application, Django, tracking, movies

ŽIVOTOPIS

Autor ovog završnog rada, Aleksej Vujić, rođen je u Vukovaru 26.2.1999. Srednjoškolsko školovanje završio je u Gimnaziji Vukovar, opći smjer. Aleksej Vujić trenutno je student treće godine stručnog studija računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek.

_____

Potpis autora