Web portal za filmove

Lucić, Dario

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

WEB PORTAL ZA FILMOVE

Diplomski rad

Dario Lucić

Osijek, 2022.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit

Osijek, 19.09.2022.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za diplomski ispit

Ime i prezime Pristupnika:	Dario Lucić
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. Pristupnika, godina upisa:	D-1139R, 13.10.2020.
OIB studenta:	48785772977
Mentor:	Prof. dr. sc. Krešimir Nenadić
Sumentor:	,
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Izv. prof. dr. sc. Alfonzo Baumgartner
Član Povjerenstva 1:	Prof. dr. sc. Krešimir Nenadić
Član Povjerenstva 2:	Robert Šojo, mag. ing. comp.
Naslov diplomskog rada:	Web portal za filmove
Znanstvena grana diplomskog rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak diplomskog rada:	Potrebno je izraditi web portal za ljubitelje filmova. Modelirati i izraditi bazu podataka koja će podržavati web portal. Omogućiti registraciju novih korisnika. Registriranim korisnicima omogućiti vođenje osobne evidencije pregledanih filmova koje registrirani korisnici koji su pogledali taj film mogu ocijeniti i komentirati. Ugraditi funkcionalnost pomoću koje registrirani korisnik može kreirati listu filmova koje želi pogledati. Nakon što je korisnik pregledao film iz liste željenih filmova, pregledani film je potrebno prebaciti u listu pogledanih filmova. Tema rezervirana: Dario Lucić
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 2 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	19.09.2022.

Potvrda mentora o predaji konačne verzije rada:	*Mentor elektronički potpisao predaju konačne verzije.*
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA O ORIGINALNOSTI RADA

Osijek, 27.09.2022.

Ime i prezime studenta:	Dario Lucić
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-1139R, 13.10.2020.
Turnitin podudaranje [%]:	5

Ovom izjavom izjavljujem da je rad pod nazivom**: Web portal za filmove**

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 studenta:

SADRŽAJ

1. UVOD

Glavni cilj ovog rada bio je uspostava potpuno funkcionalne aplikacije koja predstavlja web portal kojeg korisnici mogu koristiti za organiziranje filmova koje su pogledali, dodavali u listu filmove koje žele pogledati kao i ocjenjivati i davati komentare na izabrane naslove. Ovakva aplikacija može biti od velike koristi ljubiteljima filmova jer daje mogućnost organiziranja pogledanih i nepogledanih filmova prijavljenih korisnika. Može im uštediti vrijeme jer imaju spremljene filmove koje žele pogledati na jednom mjestu, bez potrebne ponovne pretrage.

U drugom poglavlju prikazan je pregled sličnih aplikacija koje su uvelike bile inspiracije za aplikaciju koja je predstavljena u radu. Treće poglavlje daje malo detaljniji uvid u tehnologije koje su korištene pri izradi, dijelovi kôda koji su pisani u jezicima u kojim je programirano kao i problemi koji su riješeni. U četvrtom dijelu kroz slike je prikazan izgled aplikacije, odnosno korisničkog sučelja. Opisan je način kako se izvršavaju prijava i registracija, kako izgleda početna stranica, lista filmova koje korisnik želi pogledati kao i način na koji korisnik može dodati ocjenu i komentar. Peto poglavlje služi kao zaključak, kratak osvrt na napravljeno kao i usporedba s prethodno navedenim sličnim aplikacijama.

1.1. Zadatak diplomskog rada

Potrebno je izraditi web portal za ljubitelje filmova. Modelirati i izraditi bazu podataka koja će podržavati web portal. Omogućiti registraciju novih korisnika. Registriranim korisnicima omogućiti vođenje osobne evidencije pregledanih filmova koje registrirani korisnici koji su pogledali taj film mogu ocijeniti i komentirati. Ugraditi funkcionalnost pomoću koje registrirani korisnik može kreirati listu filmova koje želi pogledati. Nakon što je korisnik pregledao film iz liste željenih filmova, pregledani film je potrebno prebaciti u listu pogledanih filmova.

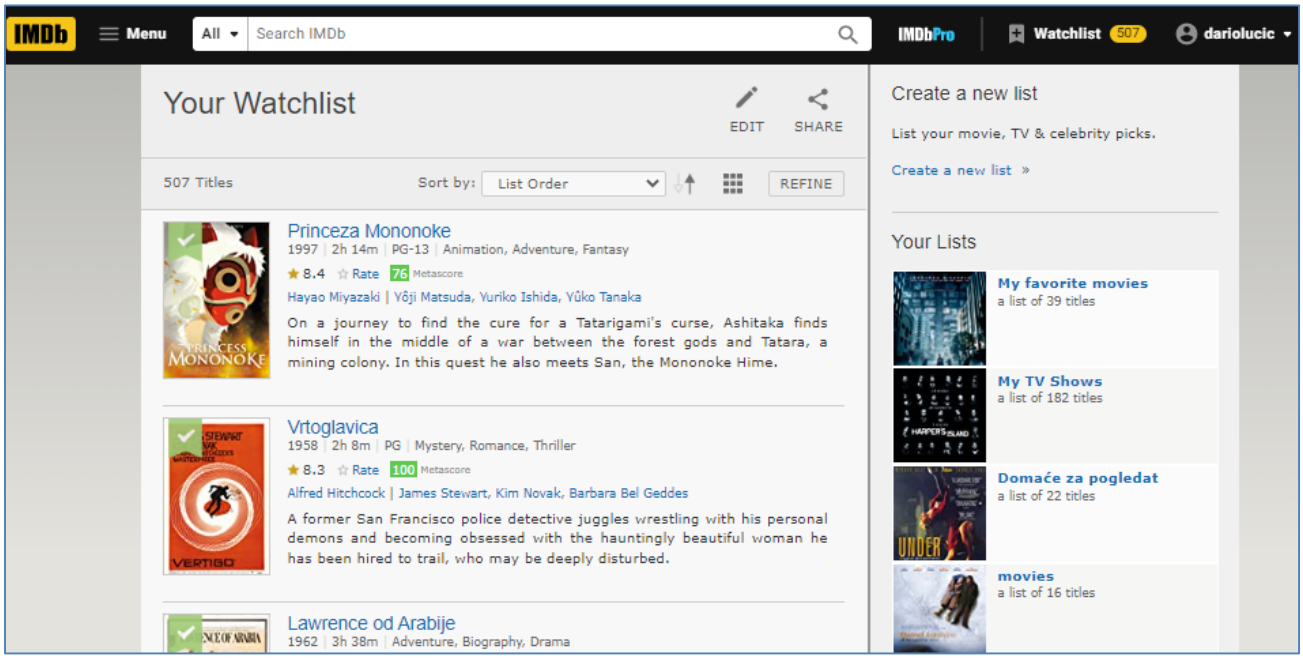
2. PREGLED SLIČNIH APLIKACIJA

U ovom poglavlju opisano je pet različitih, već postojećih internet stranica koje su po funkcionalnostima i po načinu rada dosta slične web portalu koji je izrađen za potrebe ovog rada. Ukratko su opisane mogućnosti koje stranice nude, njihov izgled i sličnosti s onom prikazanom u narednim poglavljima.

2.1. IMDb

IMDb (engl. *Internet Movie Database*) je najpoznatija i najveća baza podataka o filmovima koju možemo pronaći na internetu. Sadrži mnoštvo podataka o filmovima, glumcima, redateljima, producentima, TV serijama kao što ima i sve detaljne podatke o filmskim nagradama kao što su npr. Oscar, Emmy te još mnogo tog. Ovaj portal bio je glavna inspiracija za izradu web aplikacije prezentiranu u radu, što zbog sličnosti u radu što zbog lakoće korištenja. [1]

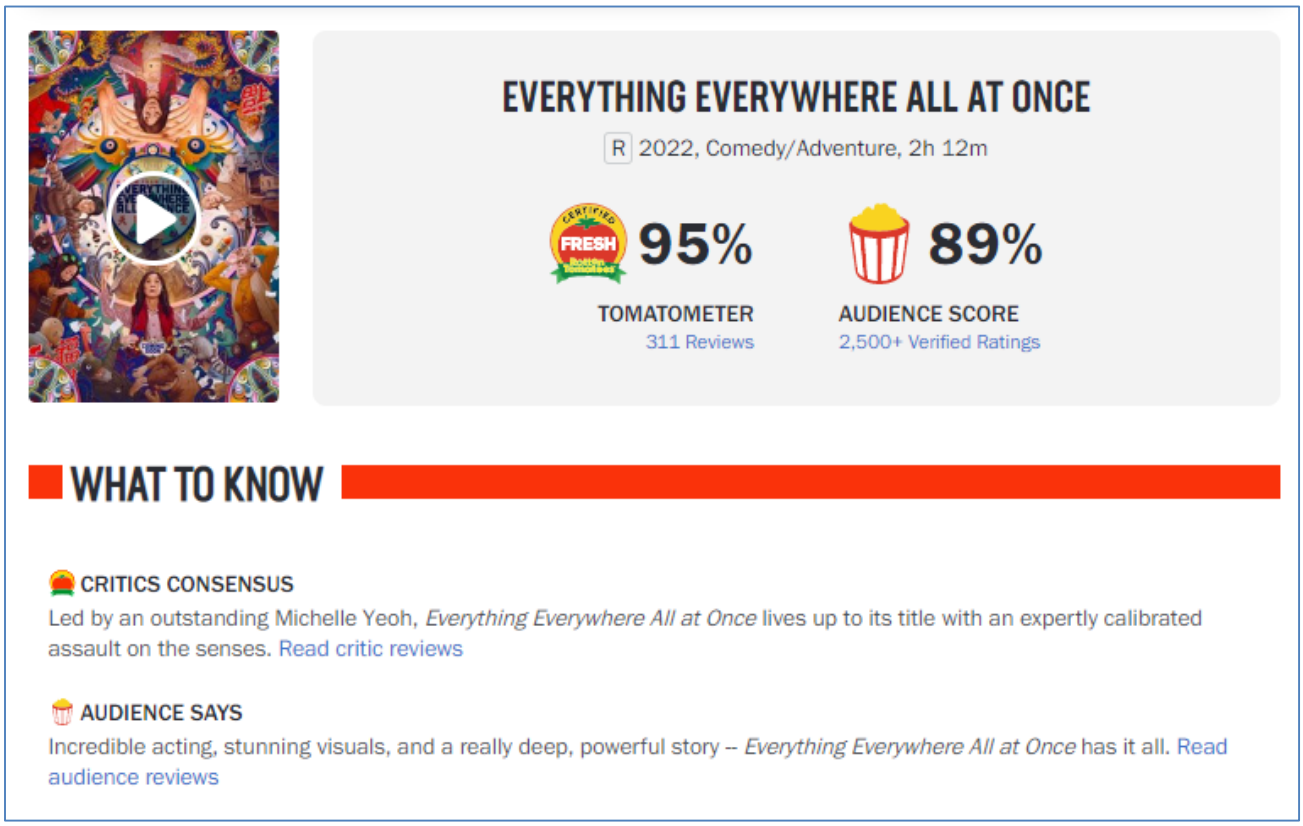
Između ostalog, na portalu postoji i opcija dodavanja filmova na listu za pogledati (engl. *watchlist*) što je prikazano na slici 2.1.



Sl. 2.1. *Prikaz korisnikove liste filmova/serija za pogledati na portalu IMDb.*

2.2. Rotten Tomatoes

Još jedan od jako popularnih portala posvećen filmova, s tim da je glavna razlika u odnosu na prethodno spomenuti ta što je ovdje glavni fokus ocjena kritičara koji su pogledali pa naknadno i recenzirali film. Kao što se može vidjeti na slici 2.2., konačna ocjena filma prikazana je u postotku, odnosno koliko kritičara se svidio film. Naravno na portalu je omogućeno i registriranje korisnika i ljubitelja filmova kao i davanje ocjena te komentiranje. Može se vidjeti kako je određen konačni konsenzus kritičara, tj. nekakva zbirna ocjena i ukupni dojam prema odabranom naslovu. [2]



Sl. 2.2. *Sučelje odabranog filma na portalu Rotten Tomatoes.*

Registracija je omogućena i putem Facebook profila te je omogućeno dodavanje raznih filmova i TV serija na listu za pogledati kao što je to bio slučaj i na portalu IMDb.

Ovdje se također mogu dodavati filmovi i TV serije na listu želja, kao što su omogućeni ocjenjivanje i komentiranje naslova.

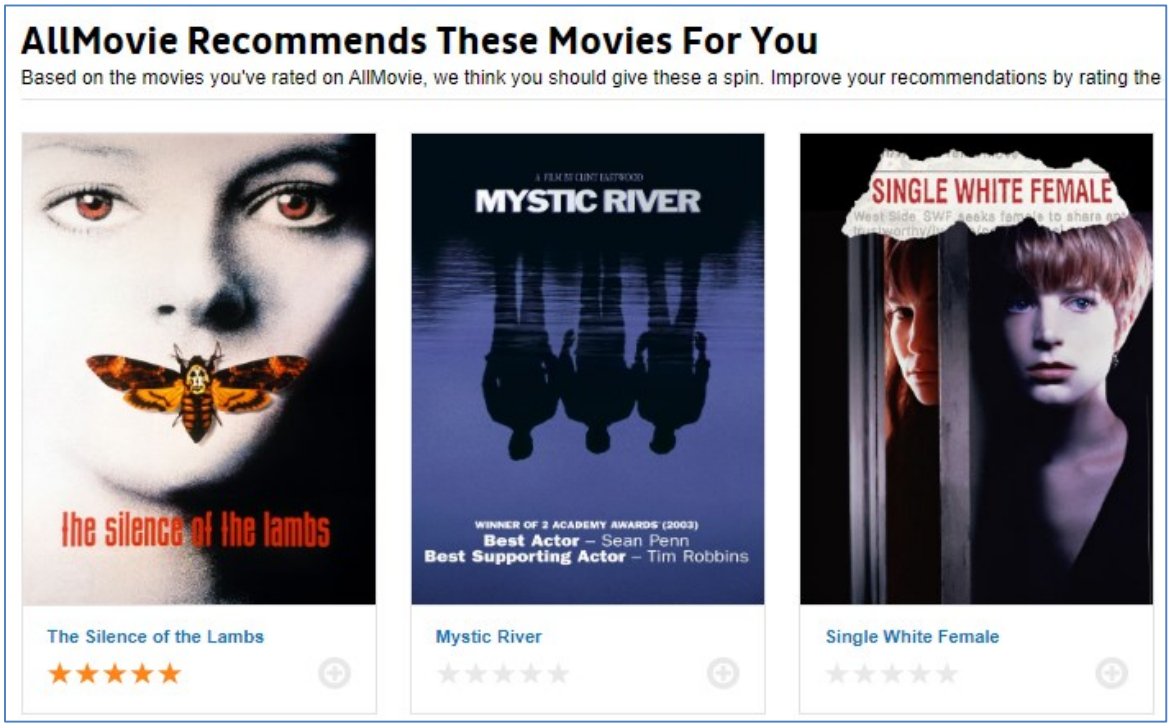
2.3. AllMovie

AllMovie je web portal koji nudi opširne i detaljne izvore gdje se može pronaći nešto više o filmovima, glumcima i redateljima koje korisnik preferira.

Na portalu se može pronaći:

- Recenzije nadolazećih filmova kao i klasika,
- Detaljne informacije o omiljenim filmovima korisnika,
- Novi filmovi u kinima – pretraživanje filmova koji igraju u obližnjim kinima kao i mogućnost kupovanja ulaznica u istim,
- Filmovi dostupni za gledanje – dostupni za iznajmljivanje ili online gledanje,
- Preporuke – cijela sekcija posvećena osobnim filmskim preporukama za svakog korisnika,
- Istraživanje filmova po žanru, atmosferi ili temi. [3]

Kada korisnik unese najdraže filmove ili ih ocjeni par, stranica sa nazivom preporuke (engl. *recommendations*) ispuni se sa sličnim filmovima što je prikazano na slici 2.3.



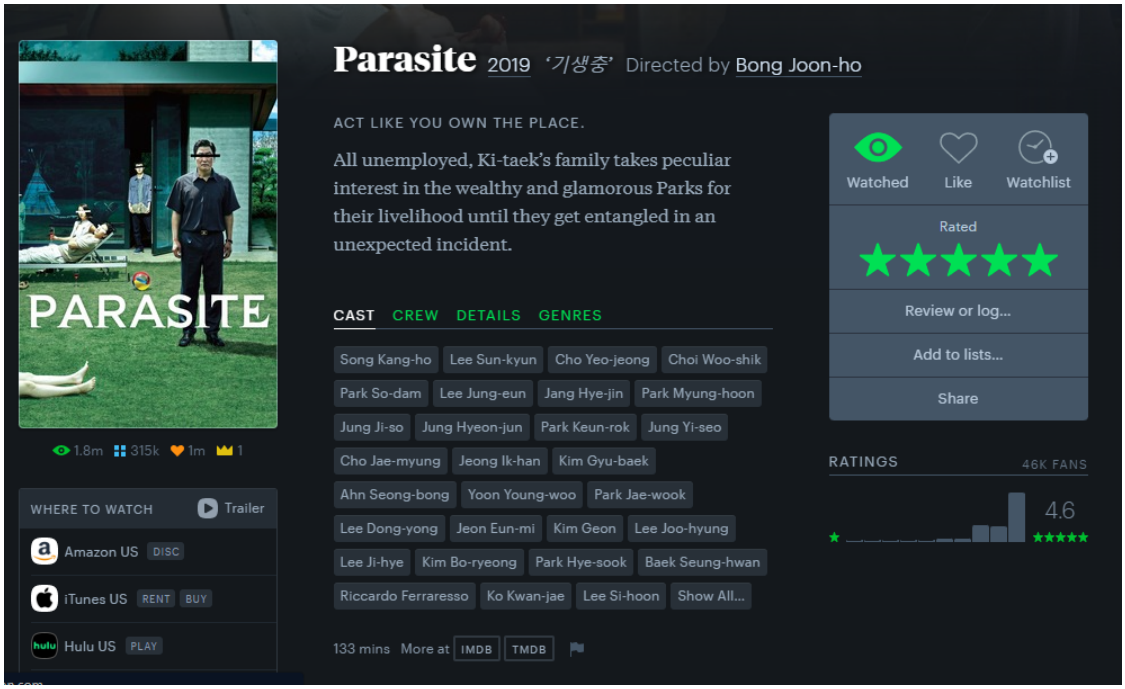
Sl. 2.3. *Izgled sučelja preporuka na portalu AllMovie.*

2.4. Letterboxd

Globalna društvena mreža koja služi korisnicima za razgovore o filmovima kao i za otkrivanje novih naslova. Koristi se kao dnevnik gdje se bilježi i dijele mišljenja o filmovima kako ih korisnici gledaju, ili jednostavno za organizaciju i svrstavanja već pogledanih naslova. Korisnicima je ponuđena opcija i da kreiraju još mnogo lista kao i onu koja sadrži njima najdraže filmove. Kao i u prethodnim primjerima i Letterboxd nudi opciju ocjenjivanja. Jedna od glavnih prednosti ovog portala je mogućnost praćenja svojih prijatelja da bi korisnici mogli vidjeti što su nedavno gledali.

Letterboxd ima i mobilne aplikacije za operacijske sustave iOS i Android, kao i za Apple TV. Ukoliko se korisnici odluče za profesionalnu plaćenu verziju portala omogućuje im se uživanje u aplikacijama bez reklama, kreiranje dodatnih statistika na profilu, opciju za odabir filtriranje onog što je dostupno na njihovim najdražim online servisima, notifikacije o filmovima koji su na listi želja, kloniranje lista ostalih članova i još dosta toga. [4]

Na slici 2.4. prikazan je izgled sučelja odabranog naslova sa portala. Može se primjetiti kako su sve najbitnije informacije prikazane odmah i korisnici ne moraju trošiti puno vremena za traženje nekakvih informacija.



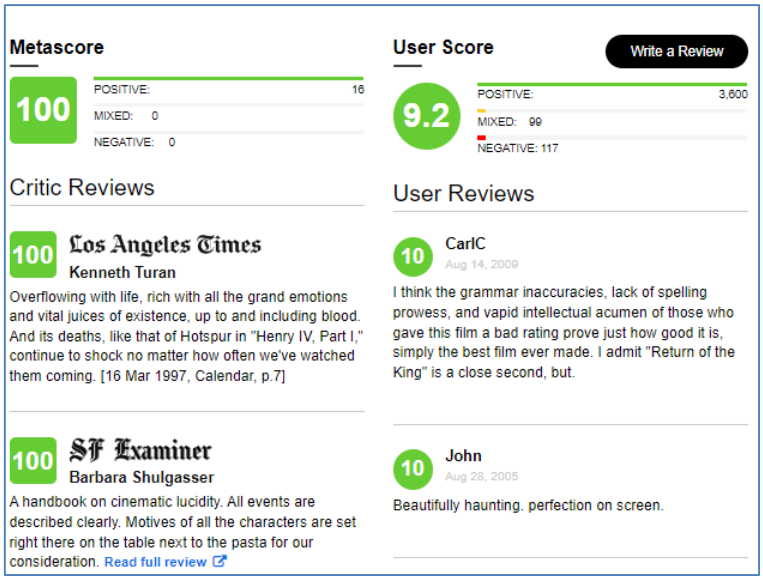
Sl. 2.4. *Izgled sučelja odabranog filma na portalu Letterboxd.*

2.5. Metacritic

Metacritic je još jedan u nizu internet portal posvećen filmovima, ali kao što je to bio slučaj i s portalom Rotten Tomatoes više je orijentiran ocjenama kritičara i stručnjaka. Osim filmova stranica ima komentare stručnjaka i za igre na raznim platformama, TV serije kao i za glazbene albume.

Nastao je kao mala ideja daleke 1999. godine: jedinstvena ocjena koja sažima mnogo recenzija iz svijeta zabave, odnosno filmova i video igara. Portal je započeo s radom 2001. i od tada rastao u kvaliteti te tako destilirao glasove mnogih kritičara u jedinstvenu ocjenu, težinski određenu po respektabilnosti i važnosti kritičara koji je ocjenu dao. Misija portala je pomoći fanovima navedenih hobija da što bolje ulože svoje vrijeme u kvalitetnije sadržaje. [5]

Ukoliko korisnik odabere određeni naslov vidi najbitnije informacije o njemu kao što je to bio slučaj i sa prethodnim portalima. Ispod svakog odabranog naslova nalaze se komentari kritike i ostalih korisnika posebno.



Sl. 2.5. *Izgled komentara kritike i publike na portalu Metacritic.*

Kao što je prikazano na slici 2.5. svaka ocjena i kritike i publike podijeljena je na tri različite grupe, ocjene od nula do 40 označene su crvenom bojom i dodijeljeno im je ime negativna (engl. *negative*). Ukoliko je naslov dobio ocjenu od 40 do 60 ta kritika je označena žutom bojom a ime joj je mješovita (engl. *mixed*) dok je ocjena od 60 do 100 označena zelenom bojom i smatra se pozitivnom ocjenom (engl. *positive*).

3. TEHNOLOGIJE KORIŠTENE U IZRADI APLIKACIJE

Aplikacija je podijeljena na tri dijela koja međusobno komuniciraju jedan s drugim.

Prvi opisani dio je baza podataka. Tu se nalaze tablice iz kojih će aplikacija dohvaćati potrebne podatke ovisno o akciji koju izvršava te spremati ako je to potrebno. Za bazu podataka korištena je PostgreSQL.

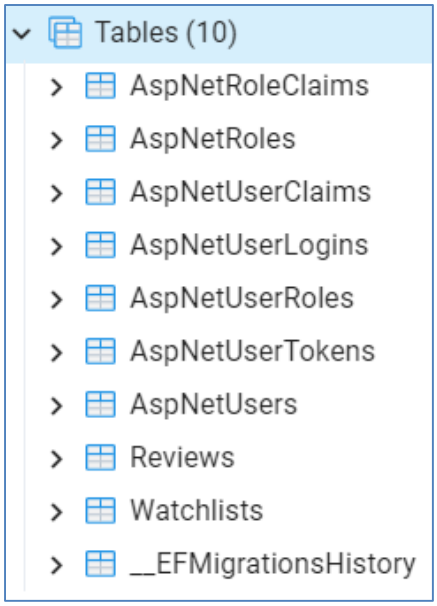
Drugi je poslužiteljski dio (engl. *backend*) koji služi kao srednji sloj između baze podataka i strane klijenta. Korisnik nikada izravno ne komunicira s bazom podataka u ovom slučaju nego svu komunikaciju obavlja poslužitelj. Za programiranje poslužitelja je korištena tehnologija ASP.NET Core Web Api gdje se koristi programski jezik C#.

Treći dio je klijent (engl. *frontend*), odnosno dio gdje se piše kôd s čijim će komponentama korisnici imati interakciju. Za klijentski dio je korišten popularni radni okvir (engl. *framework*) Angular koji koristi jezik za označavanje HTML (engl. *HyperText Markup Language*), stilski jezik CSS (engl. *Cascading Style Sheets*) te programski jezik TypeScript.

3.1. Baza podataka

Kao što je spomenuto u uvodu u 3. poglavlje za aplikaciju je korištena PostgreSQL baza podataka. Takva baza je poznata kao jako moćan sustav objektno-relacijskih baza podataka otvorenog kôda koji proširuje obični SQL te dodaje mnoge značajke koje pohranjuju kompliciranа radna opterećenja podataka na siguran način. Na slici 3.1. prikazane su tablice koje su potrebne za uspješno izvršavanje web portala za filmove. Kao što je spomenuto u početku ovog poglavlja, baza podataka ima izravnu interakciju sa poslužiteljskim dijelom aplikacije. Poslužiteljski dio je taj koji je kreirao vidljive tablice. To je ostvarivo iz razloga što je korišten takozvani prvo-kôd način (engl. *code-first*) koji je detaljnije objašnjen kasnije. [6]

Prvih sedam tablica automatski se kreiraju korištenjem ASP.NET Core Identity dodatka poslužiteljskom dijelu programa koji se brine za autentikaciju i autorizaciju registriranih i prijavljenih korisnika. Tablice recenzije (engl. *Reviews*) i liste želja (engl. *Watchlists*) su tablice koje će spremati podatke o filmovima koje je korisnik pogledao, ocijenio, komentirao te ima želju pogledati.

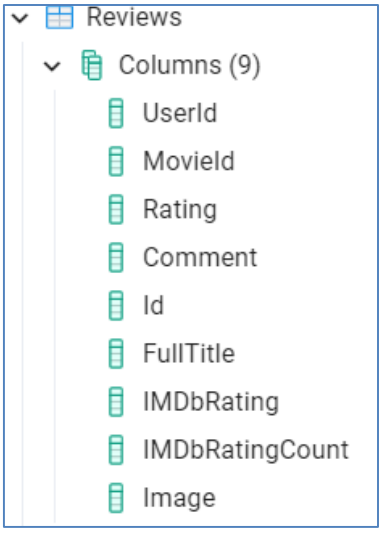


Sl. 3.1. *Tablice korištene za aplikaciju.*

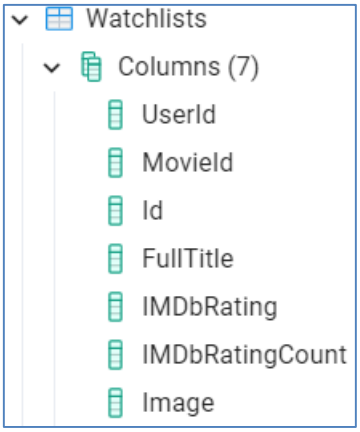
Na slici 3.2. prikazan je izgled tablice recenzije. Tablica sadrži informacije koje su korištene prilikom dobivanja filmova koje je korisnik pogledao, odnosno ocijenio ili komentirao. Korisnik u isto vrijeme ne mora i komentirati i ocijeniti film nego je dozvoljena samo jedna opcija što je objašnjeno kasnije prilikom kreiranja ove tablice.

Tablica sadrži stupce:

- Id – primarni ključ tablice,
- UserId – ID korisnika koji je ocijenio ili komentirao film,
- MovieId – ID filma kojeg je korisnik ocijenio ili komentirao,
- Ocjena (engl. *Rating*) – ocjena koju je korisnik dao odabranom filmu,
- Komentar (engl. Comment) – komentar korisnika na film,
- Ime filma,
- IMDb ocjena,
- Broj ljubitelja filmova koji je film ocijenio,
- Poster filma.



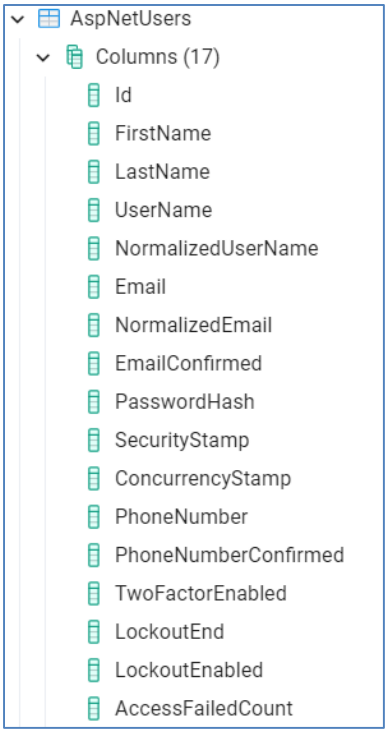
Sl. 3.2. *Tablica recenzije.*



Sl. 3.3. *Tablica liste želja.*

Na slici 3.3. prikazana je tablica liste želja preko koje će se dohvaćati podaci koji će nam reći koje filmove prijavljeni korisnik ima na svojoj listi. Tablica sadrži stupce:

- Id – primarni ključ tablice koji označava određeni zapis u istoj,
- UserId – strani ključ iz tablice korisnici koji označava prijavljenog korisnika,
- MovieId – ID filma kojeg je korisnik dodao na listu,
- Ime filma,
- IMDb ocjena,
- Broj ljubitelja filmova koji je film ocijenio,
- Poster filma.



Sl. 3.4. *Tablica korisnici.*

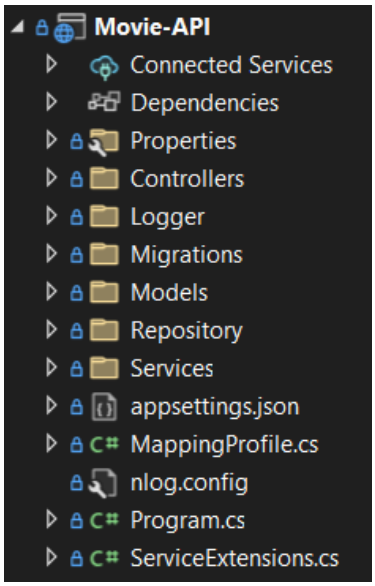
Na slici 3.4. prikazana je tablica korisnici, jedna od tablica koju je automatski kreirao već spomenuti ASP.NET Core Identity nakon što ga se doda u poslužiteljski dio kôda. To je tablica koja služi pri registriranju i prijavi korisnika na aplikaciju. Kod registracije se dodaje novi korisnik u nju, odnosno podaci koji su zahtijevani dok se kod prijave vrši provjera nalazi se li traženi korisnik u bazi te da li unešeni podaci odgovaraju onima u njoj.

3.2. Poslužitelj

Tehnologija korištena za kreiranje i uspostavu poslužiteljskog dijela programa je ASP.NET Core Web API (engl. *Application Programming Interface*). To je tehnologija koja olakšava izgradnju HTTP (engl. *Hypertext Transfer Protocol*) servisa koji doseže velik raspon klijenata, uključujući preglednike te mobilne uređaje, a programski jezik koji se koristi je C#. [7]

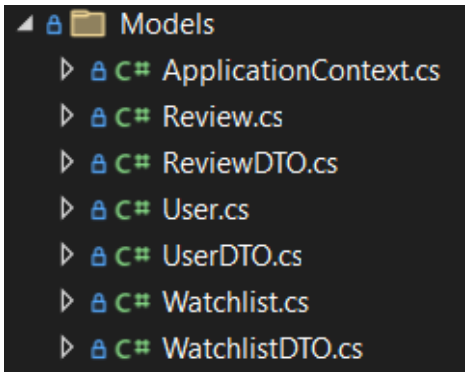
Za izradu aplikacije je korišten takozvani prvo-kôd način (engl. *code-first*) programiranja. Takav pristup omogućuje programerima laganu komunikaciju s bazom podataka bez potrebe pisanja ijedne linije SQL kôda. Potrebno je sastaviti modele tablica koje će predstavljati identičnu tablicu koja će se kasnije migrirati u bazu podataka. Tablice se kreiraju u kontekst klasi koja je prikazana nešto

kasnije. Kreirani modeli se naknadno spajaju na repozitorije gdje se mogu dodati gotove funkcije dohvaćanja, brisanja, unosa te ažuriranja podataka na entitetima.



Sl. 3.5. *Struktura mapa i datoteka u poslužiteljskom dijelu aplikacije.*

Repozitoriji se kasnije dodaju u takozvane kontrolere putem poznatog obrasca ubrizgavanja ovisnosti putem konstruktora gdje se u konačnici vrše REST (engl. *Representational State Transfer*) metode koje onda može konzumirati klijentska strana aplikacije. Na slici 3.5. prikazana je struktura mapa i datoteka u poslužiteljskom dijelu aplikacije. Po strukturi mapa može se primjetiti da je korišteno i logiranje koje će programerima dati uvid na sve REST metode koje korisnici aplikacije izvršavaju te će na taj način lakše moći pronaći potencijalne greške u radu aplikacije.



Sl. 3.6. *Struktura mape modeli.*

Na slici 3.6. mogu se vidjeti modeli, odnosno tablice koje su kreirane u bazi podataka koje su već prikazane. Za svaku tablicu napravljen je i DTO (engl. *Data Transfer Object*), odnosno objekt koji se koristi u kontrolerima pri komunikaciji s klijentskim dijelom aplikacije, tj. s onim tko konzumira metode u kontroleru. Na ovaj način se ostvaruje sigurniji prijenos podataka jer se ne koriste izravno modeli baze podataka.

```
public class Watchlist
{
    [Key]
    [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
    public int Id { get; set; }

    [ForeignKey(nameof(User))]
    public string UserId { get; set; }
    public User User { get; set; }

    [Required]
    public string MovieId { get; set; }

    [Required]
    public string FullTitle { get; set; }

    [Required]
    public string Image { get; set; }

    [Required]
    public string IMDbRating { get; set; }

    [Required]
    public string IMDbRatingCount { get; set; }
}
```

Sl. 3.7. *Klasa lista želja u poslužiteljskom dijelu aplikacije.*

Na slici 3.7. prikazana je klasa lista želja u C# jeziku, odnosno klasa čija su svojstva koristila kao temelj za izradu tablice u PostgreSQL bazi podataka. Iznad Id svojstva mogu se vidjeti dva dekoratora. Prvi govori da je riječ o svojstvu koje će se kreirati kao primarni ključ tablice, dok mu drugi dodjeljuje mogućnost inkrementiranja vrijednosti za svaki novo uneseni podatak.

Iznad svojstva UserId postavljen je dekorator koji govori da je riječ o stranom ključu, odnosno o primarnom ključu unutar neke druge tablice odnosno modela. Iznad svojstva MovieId dodan je dekorator koji označava ovaj stupac kao obavezan. Dalje u aplikaciji se javlja greška ukoliko ovaj podatak nije dostavljen.

```
public class Review
{
    [Key]
    [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
    public int Id { get; set; }

    [ForeignKey(nameof(User))]
    public string UserId { get; set; }
    public User User { get; set; }

    [Required]
    public string MovieId { get; set; }

    public int? Rating { get; set; }

    public string? Comment { get; set; }

    [Required]
    public string FullTitle { get; set; }

    [Required]
    public string Image { get; set; }

    [Required]
    public string IMDbRating { get; set; }

    [Required]
    public string IMDbRatingCount { get; set; }
}
```

Sl. 3.8. *Klasa recenzija u poslužiteljskom dijelu aplikacije.*

Na slici 3.8. prikazana je klasa koja je ekvivalent tablici koja se kreirala u bazi podataka. Razlika između ove i prethodno prikazane klase je ta što se ovdje još nalaze svojstva ocjena i komentar. Može se primjetiti kako iznad ova dva svojstva ne postoji dekorator koji predstavlja obavezne stupce. Znakom ? pokraj oznake tipa svojstva označavaju se neobavezna svojstva. Ispod UserId nalazi se i korisnik (engl. *User*) svojstvo aplikacije istoimenog tipa bez kojeg postavljanje stranog ključa u aplikaciji ne bi bilo moguće.

Na slici 3.9. može se vidjeti već spominjana DTO klasa aplikacije gdje su dodana samo svojstva koja će služiti u komunikaciji sa sučeljem.

```
public class ReviewDTO
{
    public int Id { get; set; }

    [Required]
    public string UserId { get; set; }

    [Required]
    public string MovieId { get; set; }

    public int? Rating { get; set; }

    public string? Comment { get; set; }

    public string FullTitle { get; set; }

    public string Image { get; set; }

    public string IMDbRating { get; set; }

    public string IMDbRatingCount { get; set; }
}
```

Sl. 3.9. *DTO klasa recenzije u poslužiteljskom dijelu aplikacije.*

Na slici 3.10. prikazan je već spominjana kontekst klasa. Klasa služi za kreiranje tablica u bazi podataka a dodatno se može upravljati tablicama pisanjem različitih funkcija u zaštićenoj metodi ispod konstruktora. Može se vidjeti kako ova klasa nasljeđuje klasu konteksta identiteta. Na taj način dodaju se automatski kreirane tablice koje se koriste za registraciju korisnika.

Zadnje dvije metode koje su vidljive na slici su metode za kreiranje tablica recenzije i liste želja. U zagradama se dodjeljuju tipovi kakva tablica je potrebna da se napravi. Tu su dodijeljeni modeli koji su prikazani na prethodnim slikama.

```
public class ApplicationContext : IdentityDbContext<User>
{
    public ApplicationContext(DbContextOptions options)
        : base(options)
    {
    }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        base.OnModelCreating(modelBuilder);
    }

    public DbSet<Watchlist> Watchlists { get; set; }
    public DbSet<Review> Reviews { get; set; }

}
```

Sl. 3.10. *Kontekst klasa u poslužiteljskom dijelu aplikacije.*

Kada su u klasu dodane sve potrebne tablice i modeli te kada se dodaju postavke baze podataka u posebnu datoteku, vrše se takozvane migracije gdje se automatski generira SQL kôd i naknadno ažurira baza podataka tako da je programeru maksimalno olakšan posao.

Nakon kreiranja tablica u bazi podataka potrebno je dodati programsku logiku dohvaćanja iz njih, brisanje, ažuriranje i stvaranje novih podataka. Da se ne bi sav kôd pisao u kontroleru i kako bi čitkost bila što veća stvorene su repozitorij klase i sučelja, te se na taj način dobilo i na fleksibilnosti i boljoj prenosivosti podataka. Na slici 3.11. može se vidjeti jedno od kreiranih sučelja, točnije sučelje za tablicu liste želja.

```
public interface IWatchlistRepo
{
    Task<List<Watchlist>> GetWatchlistOfUser(string id);

    Task AddToWatchlist(Watchlist watchlist);

    Task DeleteFromWatchlist(Watchlist watchlist);
}
```

Sl. 3.11. *Sučelje liste želja u poslužiteljskom dijelu aplikacije.*

Sučelje koje će kasnije naslijediti repozitorij liste želja sastoji se od tri potrebne metode, odnosno tri akcije koje su potrebne da budu izvršene na tablici. Prva metoda dohvaća listu želja nekog korisnika, odnosno u ovom slučaju to je prijavljeni korisnik. Metoda vraća listu filmova koje je korisnik dodao na svoju listu želja, odnosno vraća listu modela koji je kreiran za ovu tablicu. Ovoj metodi potrebno je u parametrima predati ID, odnosno ID korisnika koji je prijavljen kako bi se mogli pronaći svi filmovi koji pripadaju njegovoj listi.

Druga metoda služi za dodavanje novih filmova na listu želja prijavljenog korisnika. Kao parametar predaje mu se objekt modela liste želja koji je kreiran. Taj model kao što je prethodno prikazano sadrži ID prijavljenog korisnika te ID filma koji je dodan u listu želja.

Treća metoda služi za brisanje filmova iz liste želja kojoj se također predaje objekt liste želja koji sadrži ID prijavljenog korisnika koji briše film iz liste te ID filma.

```
public class WatchlistRepo : IWatchlistRepo
{
    private readonly ApplicationContext _context;

    public WatchlistRepo(ApplicationContext context)
    {
        _context = context;
    }

    public async Task AddToWatchlist(Watchlist watchlist)
    {
        if (_context != null)
        {
            await _context.Watchlists.AddAsync(watchlist);
            await _context.SaveChangesAsync();
        }
    }
```

Sl. 3.12. *Repozitorij liste želja u poslužiteljskom dijelu aplikacije.*

Na slici 3.12. prikazan je repozitorij liste želja koji nasljeđuje kreirano sučelje prethodno prikazano. Podrazumijeva se da repozitorij onda sadrži i sve metode koje su prikazane u sučelju. U konstruktoru klase ubrizgava se prethodno kreirani kontekst koji komunicira s bazom podataka i traženim tablicama. Za potrebe ove klase bila je potrebna komunikacija s tablicom liste želja.

Prikazana je metoda dodavanja novog filma u listu. Metodi je dodijeljena ključna riječ koja je označava asinkronom, što je dobra praksa kada je riječ o akcijama na bazi podataka. Kao što je objašnjeno u sučelju kojeg klasa nasljeđuje, metoda prima objekt liste želja koji će se dodavati u kontroleru. Prvo se vrši provjera postoji li kontekst i veza s bazom te ukoliko postoji, u tablicu se asinkrono dodaje objekt iz parametra, odnosno ID korisnika koji je akciju pokrenuo te ID filma kojeg ima namjeru pogledati. Nakon ovog podaci se spremaju u bazu. Još jednom se može primjetiti kako je ASP.NET tehnologija, odnosno radni okvir entiteta (engl. *entity framework*) programeru maksimalno olakšao posao komunikacije s bazom jer nema potrebe za pisanjem SQL kôda nego za sve postoje već unaprijed definirane metode koje su u praksi korištene mnogo puta.

Na jako sličan način obavlja se i brisanje filmova iz liste želja i dohvaćanje liste, te metode koje su potrebne u repozitoriju za tablicu recenzije.

```
[HttpGet("{id}", Name = "WatchlistOfUser")]
public async Task<IActionResult> GetWatchlistOfUser(string id)
{
    try
    {
        var watchlist = await _watchlistRepo.GetWatchlistOfUser(id);

        if (watchlist == null)
        {
            _logger.LogError($"Watchlist of user with id: {id}, hasn't been found in db.");
            return NotFound();
        }
        else
        {
            _logger.LogInfo($"Returned watchlist of user with id: {id}");
            var watchlistResult = _mapper.Map<IEnumerable<WatchlistDTO>>(watchlist);
            return Ok(watchlistResult);
        }
    }
    catch (Exception ex)
    {
        _logger.LogError($"Something went wrong inside GetWatchlistOfUser action: {ex.Message}");
        return StatusCode(500, "Internal server error");
    }
}
```

Sl. 3.13. *Kontroler liste želja u poslužiteljskom dijelu aplikacije.*

Na slici 3.13. prikazan je kontroler liste želja, odnosno metoda za dohvaćanje liste želja prijavljenog korisnika. U konstruktoru klase dodan je objekt tipa repozitorija liste želja, na sličan način što je prikazano u istom repozitoriju gdje se dodavao kontekst.

Iznad definiranja željene metode, potrebno je dodati HTTP metodu koja će se obavljati. Postoje četiri glavne metode:

- POST – služi za dodavanje novih podataka,
- GET – služi za dohvaćanje postojećih podataka iz baze,
- DELETE – služi za brisanje podataka iz baze,
- PUT – služi za ažuriranje postojećih podataka.

Nakon definiranja metode kao one koja će dohvaćati podatke iz baze, definira se povratni tip i postavlja parametar. Metoda će od vanjske aplikacije koja koristi ovaj API primati ID od korisnika. Isti ID predaje se kao parametar u metodu koja je kreirana u repozitoriju. Ukoliko ne postoji takav ID u bazi vraća se status koji prikazuje pogrešku te se putem prethodno spominjanog logiranja zapisuje poruka koja programerima govori koja akcija je izvršena.

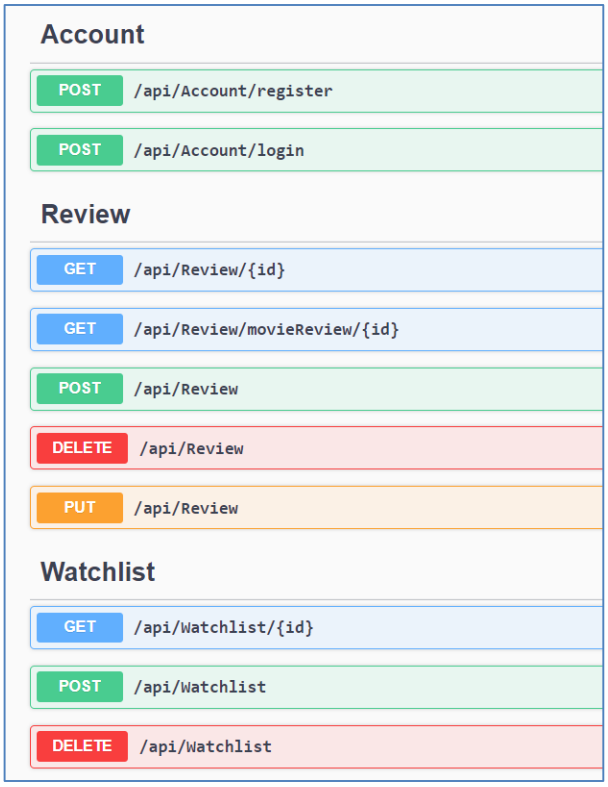
Ukoliko je akcija izvršena po planu korisnik dobija listu filmova koje je dodao na listu želja u DTO tipu objekta. Na ovaj način se može vidjeti prednost DTO-a gdje se korisniku šalje točno onakav objekt koji programeri to žele. Na slici se može primjetiti kako je dodana i provjera gdje se hvataju pogreške, odnosno interne greške unutar servera.

```
public async Task<string> CreateToken()
{
    var signingCredentials = GetSigningCredentials();
    var claims = GetClaims();
    var token = GenerateTokenOptions(signingCredentials, claims);

    return new JwtSecurityTokenHandler().WriteToken(token);
}

private JwtSecurityToken GenerateTokenOptions(SigningCredentials signingCredentials, List<Claim> claims)
{
    var jwtSettings = _configuration.GetSection("Jwt");
    var expiration = DateTime.Now.AddMinutes(15);

    var token = new JwtSecurityToken(
        issuer: jwtSettings.GetSection("Issuer").Value,
        claims: claims,
        expires: expiration,
        signingCredentials: signingCredentials
        );

    return token;
}
```

Sl. 3.14. *Kreiranje tokena za autorizaciju korisnika u poslužiteljskom dijelu aplikacije.*

Na slici 3.14. prikazan je dosta bitan korak koji se koristi prilikom prijave korisnika na portal. Tokeni funkcioniraju na sličan način kao sesije, te omogućavaju prijavljenom korisniku da posjećuje određene stranice aplikacije ukoliko ima dozvolu za to. Može se vidjeti kako se u postavkama tokena dodjeljuje vrijeme trajanja istog te još neke opcije potrebne za kreiranje. Ukoliko je korisnik uspješno prijavljen na portal ima 15 minuta (u ovom slučaju) do isteka tokena te će se morati ponovno prijaviti. Trajanje tokena moguće je postaviti neograničeno.

Na slici 3.15. prikazane su spominjane metode koje se izvršavaju na tablicama baze podataka. Može se primjetiti kako se na tablici recenzije izvršavaju dvije metode dohvaćanja podataka iz iste. Jedna služi za dohvaćanje svih komentara i ocjena korisnika na određene filmove dok druga sadrži podatke za samo jedan film koje će biti prikazane na stranici istog. Dvije metode iste akcije ne smiju sadržavati isti put odnosno link do nje.



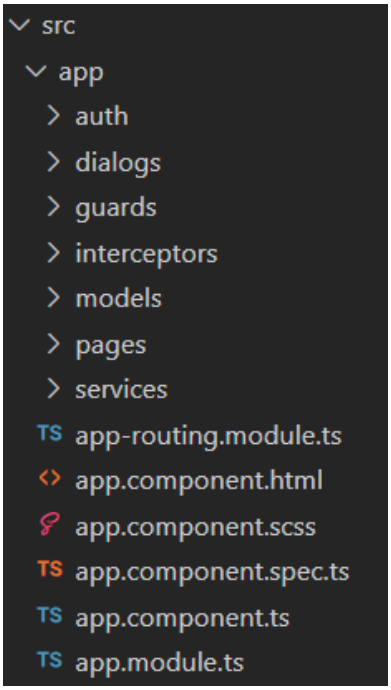
Sl. 3.15. *Metode korištene na tablicama baze podataka.*

3.3. Klijent

Za izgradnju korisničkog sučelja i klijenta koji će konzumirati podatke koje šalje poslužitelj korišten je radni okvir Angular. Angular je razvojna platforma koja je izgrađena u programskom jeziku Typescript. Kao platforma uključuje:

- Skup alata za razvojne programere koji pomaže u izgradnji, razvoju, ažuriranju i testiranju kôda,
- Zbirke integriranih biblioteka koje pokrivaju velik raspon različitih značajki, uključujući usmjeravanje, upravljanje formama, komunikaciji između klijenta i poslužitelja i drugo,
- Komponentni okvir za izgradnju skalabilnih aplikacija za internet. [8]

Na slici 3.16. prikazana je struktura aplikacije kreirane u Angularu. Napravljene su posebne mape za autentikaciju, modele objekata koje koristi aplikacija (recenzija, lista želja), servise koji će dohvaćati podatke sa poslužitelja, stranice aplikacije kako bi se bolje organiziralo i snalazilo u njoj.



Sl. 3.16. *Struktura mapa i datoteka u klijentskom dijelu aplikacije.*

Ispod mapa prikazana je glavna komponenta aplikacije. Svaka komponenta sadrži TypeScript datoteku za pisanje logike komponente, HTML datoteku za pisanje kôda sučelja, datoteku za testiranje komponente te datoteku za stiliziranje. Tu se još nalazi i glavni datoteka za usmjeravanje puteva (engl. *paths*) aplikacije koji je prikazan na slici 3.17.

```
const routes: Routes = [
  {
    path: 'auth',
    canActivate: [LogedInGuard],
    component: LoginComponent,
    loadChildren: () => import('./auth/auth.module') /
    .then(m => m.AuthModule),
  },
  {
    path: 'register',
    canActivate: [LogedInGuard],
    component: RegistrationComponent,
    loadChildren: () => import('./auth/auth.module') /
    .then(m => m.AuthModule),
  },
  {
    path: 'pages',
    canActivate: [AuthGuard],
    canDeactivate: [PosGuard],
    loadChildren: () => import('./pages/pages.module')
    .then(m => m.PagesModule),
```

Sl. 3.17. *Dio datoteke za usmjeravanje u klijentskom dijelu aplikacije.*

U kôdu se može vidjeti kako postoje i zabrane koje ne dozvoljavaju korisnicima koji nisu prijavljeni prikazivanje stranica aplikacije. Dio logike u komponenti za prijavu prikazan je na slici 3.18., odnosno metoda koja se okida pritiskom na gumb prijave.

```
async submit() {
   if(this.form.invalid){
     return;
   }
   console.log(this.form.getRawValue())
   const response = await this.http.post("https://localhost:7288/api/Account/login",
    this.form.getRawValue(), httpOptions )
     .subscribe(res  => {
      const token = JSON.stringify(Object.values(res)[0])
      console.log(token);

       localStorage.setItem("token", token)
       this.router.navigate(['/pages/home']);
     });
 }
```

Sl. 3.18. *Metoda za prijavu u klijentskom dijelu aplikacije.*

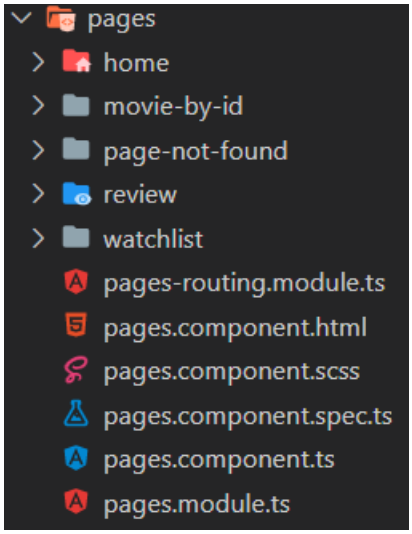
Može se vidjeti kako se prvo izvršava validacija koja provjerava je li korisnik unio ispravnu formu. Ukoliko jest, asinkrono se šalje POST metoda na link koji predstavlja poslužiteljski dio aplikacije koji je prethodno prikazan. Ako je korisnik unio točnu e-mail adresu i lozinku, kao rezultat mu se vraća token koji se sprema u lokalno skladište (engl. *local storage*). Nakon uspješne prijave korisnika se preusmjerava na početnu stranicu aplikacije.

Na slici 3.19. prikazan je HTML kôd za izradu forme za prijavu.

```
<form [formGroup]="form" (submit)="submit()">
  <div id="container" [formGroup]="form">
    <mat-card>
      <mat-toolbar color="primary">
        <span>Login</span>
      </mat-toolbar>
      <mat-card-content>
        <mat-form-field appearance="outline">
          <mat-label>Enter Email</mat-label>
          <input formControlName="email" type="email" matInput placeholder="Enter email" />
          <mat-hint *ngIf="form.touched && form.controls['email'].hasError('email')">Email is required</mat-hint>
        </mat-form-field>
        <mat-form-field appearance="outline">
          <mat-label>Enter Password</mat-label>
          <input formControlName="password" matInput type="password" placeholder="Enter password" />
          <mat-hint *ngIf="form.controls['password'].hasError('password')">Password is required (beetwen 6 and 16
            charachters)</mat-hint>
        </mat-form-field>
      </mat-card-content>
      <mat-card-footer>
        <button (click)="submit()" mat-raised-button color="primary">Login</button>
      </mat-card-footer>
      <a id="loginNav" routerLink="/register">Are you new user? Go to register...</a>
    </mat-card>
  </div>
```

Sl. 3.19. *HTML kôd za prijavu u klijentskom dijelu aplikacije.*

Na slici 3.20. prikazana je struktura mapa i datoteka u mapi stranice (engl. *pages*) gdje su prikazane glavne komponente aplikacije u klijentskom dijelu. Mogu se vidjeti komponenta početne stranice, komponenta gdje je prikazan pojedinačni odabrani film, komponenta ukoliko ne postoji traženi link od korisnika te komponente gdje je prikazana lista želja te lista filmova koje je korisnik već pogledao. Ispod mapa nalaze se datoteke komponente stranice te datoteka koja upravlja rutama nakon prijave korisnika.



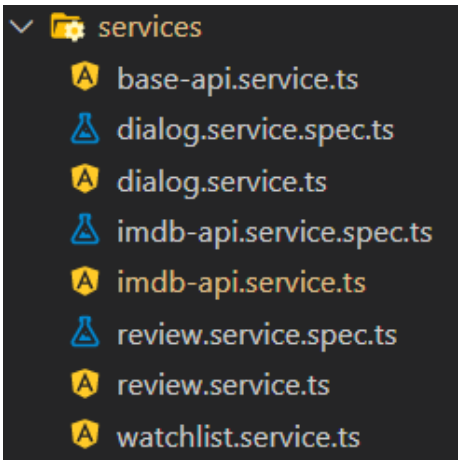
Sl. 3.20. *Struktura mapa i datoteka glavnih komponenata u klijentskom dijelu aplikacije.*

```
const routes: Routes = [
  {
    path: '',
    component: PagesComponent
  },
  {
    path: 'movie/:id',
    component: MovieByIdComponent
  },
  {
    path: 'home',
    component: HomeComponent
  },
  {
    path: 'watchlist',
    component: WatchlistComponent
  },
  {
    path: 'myMovies',
    component: ReviewComponent
  },
];
```

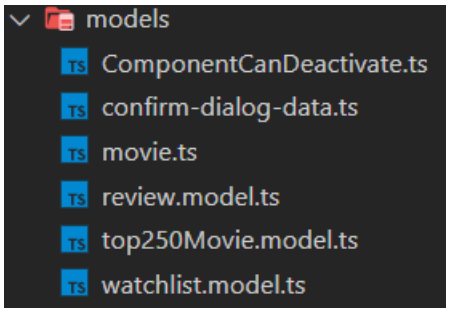
Sl. 3.21. *Dio datoteke za usmjeravanje nakon prijave u klijentskom dijelu aplikacije.*

Na slici 3.21. prikazani su putevi aplikacije nakon prijave korisnika. Može se vidjeti kako svaki put ima svoju komponentu koja ga predstavlja te link kako da se dođe do njega. Prikazane komponente su iste koje su navedene na slici 3.20.

Nakon komponenata, još jedan od najbitnijih koncepata razvijanja aplikacije u radnom okviru Angular su servisi. Radi preglednosti napravljena je nova mapa gdje se svrstavaju isti. Servisi obezbjeđuju komponente raznim podacima i listama podataka određene vrste kojima komponente onda upravljaju te ih prikazuju na korisničkom sučelju. Na slici 3.22. prikazani su servisi koji su korišteni u aplikaciji. Tu se nalazi servis koji dohvaća podatke sa IMDb API-ja (engl. *Application Programming Interface*) kao i servisi koji dohvaćaju podatke sa poslužitelja napravljenog za ovu aplikaciju, odnosno podatke o filmovima sa liste želja, te ocijene i komentare na film korisnika.



Sl. 3.22. *Servisi korišteni u klijentskom dijelu aplikacije.*



Sl. 3.23. *Modeli korišteni u klijentskom dijelu aplikacije.*

Kao što je to bio slučaj i u poslužiteljskom dijelu aplikacije i u klijentskom dijelu je praktično i od velike pomoći koristiti se modelima. Modeli se kao što je bio slučaj i sa servisima nalaze u posebnoj mapi radi lakše organizacije. Kao što je prikazano na slici 3.23. i ovdje se koriste identični onima koji su korišteni na poslužiteljskoj strani, odnosno modeli recenzije i liste želja koji su prikazani kasnije u poglavlju. Uz ta dva modela, još se nalaze i modeli koji se koriste prilikom dohvaćanja podataka sa IMDb API-ja. Na slici 3.24. je prikazan model koji predstavlja podatke dohvaćene sa IMDb API-ja, odnosno model koji će se koristiti prilikom izvršavanja akcija u servisima te prilikom rukovanja podataka u komponentama.

```
export interface Top250Movie {
    id: string;
    rank: string;
    title: string;
    fullTitle: string;
    year: string;
    image: string;
    crew: string;
    imDbRating: string;
    imDbRatingCount: string;
}
```

Sl. 3.24. *Model korišten prilikom rukovanja podataka s IMDb-a.*

Model sadrži podatke o dohvaćenim filmovima a upotreba istog prikazana je kasnije u poglavlju. Na slikama 3.25. i 3.26. prikazani su modeli liste želja i recenzije koji su identični onima sa strane poslužitelja.

```
export interface Review {
    id?: number;
    userId: string;
    movieId: string;
    rating?: number;
    comment?: string;
    fullTitle: string;
    image: string;
    imDbRating: string;
    imDbRatingCount: string;
}
```

Sl. 3.25. *Model recenzije.*

```
export interface Watchlist {
    id?: number;
    userID: string;
    movieId: string;
    fullTitle: string;
    image: string;
    imDbRating: string;
    imDbRatingCount: string;
}
```

Sl. 3.26. *Model lista želja.*

Na slici 3.27. prikazan je već spomenuti IMDb servis gdje se dohvaćaju podaci s njegovog API-ja. Može se vidjeti kako servis sadrži dvije metode. Prva metoda služi za dohvaćanje podataka o filmovima koji su najbolje ocijenjeni, a ideja je da se oni prikažu na početnoj stranici portala. Odgovor koji se dobije od njihovog poslužitelja sprema se kao lista modela koji je prethodno prikazan. U drugoj metodi dobijaju se podaci o filmu kojeg je korisnik odabrao. Metoda prima parametar koji predstavlja ID filma, a povratni tip metode je također model koji je napravljen.

```
export class ImdbApiService extends BaseApiService {

  constructor(http: HttpClient) {
    super(http, 'https://imdb-api.com/en/API/Top250Movies/k_vcqiga20');
  }

  getTopMovies(): Observable<Top250Movie[]> {
    return this.http.get(this.apiRoute).pipe(
      map(response => response as Top250Movie[])
    );
  }

  getMovieById(id: string): Observable<Movie>{
    return this.http.get("https://imdb-api.com/en/API/Title/k_vcqiga20/" + id).pipe(
      map(response => response as Movie)
    );
  }
}
```

Sl. 3.27. *IMDb servis datoteka.*

Također se može primjetiti kako je prilikom dohvaćanja podataka s ovog servisa potrebno proslijediti i ključ u link koji se dobije prilikom registracije na portal IMDb API dokumentacije.

```
export class ReviewService extends BaseApiService {

  constructor(http: HttpClient) {
    super(http, 'https://localhost:7182/api/review/');
  }

  getReviewOfUser(id: string): Observable<Review[]> {
    return this.http.get(this.apiRoute + id).pipe(
      map(response => response as Review[])
    )
  }

  addReview(review: Review): Observable<Review> {
    return this.http.post(this.apiRoute, review).pipe(
      map(response => response as Review)
    );
  }

  updateReview(review: Review): Observable<Review> {
    return this.http.put(this.apiRoute, review).pipe(
      map(response => response as Review)
    );
  }
}
```

Sl. 3.28. *Recenzije servis datoteka.*

Slika 3.28. prikazuje servis recenzija gdje se dohvaćaju i šalju podaci na lokalnog poslužitelja koji je prethodno kreiran. Ovdje se nalaze tri metode, u prvoj se dohvaćaju sve recenzije od prijavljenog korisnika kako bi se mogli dohvatiti filmovi koje je on pogledao, odnosno filmovi koje je ocijenio i komentirao. Dohvaćeni podaci spremaju se kao lista modela recenzije a kao parametar predaje se ID prijavljenog korisnika. Druga metoda obavlja slanje podataka prema poslužiteljskom dijelu, odnosno dodavanje nove ocijene ili komentara u bazu dok treća metoda služi za uređivanje istih. Kao parametar u ove dvije metode se predaje objekt tipa modela recenzije gdje je potrebno da isti sadrži sva obavezna polja.

Na slici 3.29. nalazi se servis koji dohvaća i šalje podatke vezane za listu želja. Također sadrži tri metode a prva ima sličnu ulogu kao prva metoda sa slike 3.28. Metoda dohvaća sve filmove koje je prijavljeni korisnik dodao na listu želja, te se dobijeni odgovor od lokalnog poslužitelja sprema kao lista modela liste želja. Druga metoda služi za dodavanje filmova na listu želja prijavljenog korisnika. Metoda prima objekt modela liste želja koji mora sadržavati sva obavezna polja. Treća metoda služi za brisanje odabranog filma iz liste želja a kao parametar predaje se ID kako bi poslužitelj znao koji red iz baze je potrebno ukloniti.

```
export class WatchlistService extends BaseApiService {

  constructor(http: HttpClient) {
    super(http, 'https://localhost:7182/api/watchlist/');
  }

  getWatchlistOfUser(id: string): Observable<Watchlist[]>{
    return this.http.get(this.apiRoute + id).pipe(
        map(response => response as Watchlist[])
    )
  }

  addToWatchlist(watchlist: Watchlist): Observable<Watchlist>{
      return this.http.post(this.apiRoute, watchlist).pipe(
        map(response => response as Watchlist)
      );
  }

  deleteFromWatchlist(id: number): Observable<Watchlist>{
    return this.http.delete(this.apiRoute + id).pipe(
      map(response => response as Watchlist)
    );
  }
}
```

Sl. 3.29. *Lista želja servis datoteka.*

Na slici 3.30. može se vidjeti dio kôda komponente početne stranice. U konstruktor se ubrizgava prethodno prikazani servis koji dohvaća podatke sa IMDb API-ja. Prilikom učitavanja stranice poziva se dolje definirana metoda koja poziva metodu iz servisa gdje se dohvaćaju najbolje ocijenjeni filmovi. Podaci koji se dobiju se spremaju u listu koja je definirana kao varijabla pri vrhu, dok ista služi pri prikazivanju podataka u tablici u HTML dijelu kôda.

```
displayedColumns: string[] = ['rank', 'title', 'imDbRating', 'imDbRatingCount'];
  top250Movies: Top250Movie[] = [];

  constructor(
    private changeDetectorRefs: ChangeDetectorRef,
    private top250moviesService: ImdbApiService) { }

  ngOnInit(): void {
    this.getAllTop250Movies();

  }

  getAllTop250Movies(){
    this.top250moviesService.getTopMovies().subscribe((data: any) => {
      this.top250Movies = data.items;
      console.log(this.top250Movies);
      this.changeDetectorRefs.detectChanges();

      return data;
    });
  }
```

Sl. 3.30. *Dio kôda komponente početne stranice.*

Na slici 3.31. prikazana je metoda koja se nalazi u komponenti lista želja. Metoda služi za dohvaćanje liste želja za prijavljenog korisnika. Kao što je objašnjeno u servisu koji dohvaća podatke vezane za liste želja od poslužitelja, metodi je potrebno proslijediti ID korisnika u parametru. ID se dobija preko tokena koji se korisniku automatski dodijeli prilikom prijave. Ukoliko je token važeći i nalazi se u lokalnom skladištu kao što je zamišljeno, iz istog se dobiju podaci o korisniku, odnosno njegov ID. Ukoliko metoda dobije podatke oni se spremaju u varijablu kao i u prethodno prikazanoj metodi sa slike 3.30.

```
getWatchlistForUser() {
    var userId;
    const token = localStorage.getItem("token");
    if (token == null) {
      console.log("Token incorrect")
    } else {
      const decodedToken = this.jwtHelper.decodeToken(token);
      var key: string[] = Object.values(decodedToken);
      console.log(key);
      userId = `${key[0][0]}`
      this.watchlistService.getWatchlistOfUser(userId).subscribe((data: any) => {
        this.watchlist = data;
        console.log(this.watchlist);
        this.changeDetectorRefs.detectChanges();

        return data;
      });
    }
  }
```

Sl. 3.31. *Dio kôda komponente lista želja.*

```
<div class="new-class">
  <table id="matTable" mat-table [dataSource]="watchlist" class="mat-elevation-z8">
    <ng-container matColumnDef="title">
      <th mat-header-cell *matHeaderCellDef id="title"> Title </th>
      <td mat-cell *matCellDef="let element" id="title"><img src="{{element.image}}" width="100" />
      <a id="movieTitleLink" [routerLink]="['/pages/movie', element.movieId]"> {{element.fullTitle}} </a> </td>
    </ng-container>
    <ng-container matColumnDef="imDbRating">
      <th mat-header-cell *matHeaderCellDef id="rating"> IMDb Rating </th>
      <td mat-cell *matCellDef="let element" id="rating">
        <mat-icon class="material-icons-outlined" [ngStyle]="{'color':'yellow'}">star</mat-icon>
        <strong>{{element.imDbRating}}</strong>
      </td>
    </ng-container>
    <ng-container matColumnDef="imDbRatingCount">
      <th mat-header-cell *matHeaderCellDef id="imDbRatingCount"> Number of Ratings </th>
      <td mat-cell *matCellDef="let element" id="imDbRatingCount"> {{element.imDbRatingCount | number }} </td>
    </ng-container>
    <tr mat-header-row *matHeaderRowDef="displayedColumns"></tr>
    <tr mat-row *matRowDef="let row; columns: displayedColumns;"></tr>
  </table>
</div>
```

Sl. 3.32. *Dio HTML kôda komponente lista želja.*

Na slici 3.32. prikazano je kako se koriste podaci dobiveni od servisa u HTML komponenti. Za prikaz filmova koje je korisnik dodao u listu želja odabrana je tablica. Može se primjetiti kako je kao izvor podataka za tablicu postavljena varijabla u koju su spremljeni dohvaćeni podaci sa poslužitelja. U tablici su prikazani ime filma uz njegov poster, IMDb ocjena te broj ljubitelja filmova koji je ocijenio prikazani film. Klikom korisnika na određeni film odlazi se na njegovu stranicu gdje se nalazi više podataka o istom.

```
<ng-container matColumnDef="rating">
      <th mat-header-cell *matHeaderCellDef id="myrating"> My Rating </th>
      <td mat-cell *matCellDef="let element" id="myrating">
        <mat-icon class="material-icons-outlined" [ngStyle]="{'color':'yellow'}">
        star</mat-icon> {{element.rating}}/5
      </td>
    </ng-container>
    <ng-container matColumnDef="isCommented">
      <th mat-header-cell *matHeaderCellDef id="isCommented"> Comment </th>
      <td mat-cell *matCellDef="let element" id="isCommented">
        <mat-icon *ngIf="element.comment" class="material-icons-outlined">comment</mat-icon>
      </td>
    </ng-container>
```

Sl. 3.33. *Dio HTML kôda komponente recenzija.*

Na slici 3.33. prikazan je još jedan dio HTML kôda, ovoga puta iz komponente recenzija gdje se prikazuju podaci o filmovima koje je korisnik ocijenio ili komentirao. Ovdje se također koristi tablični prikaz, ali u tablici se još prikazuje i ocjena korisnika na pojedini film te informacija je li korisnik već komentirao isti.

```
addToWatchlist() {
    var watchlist = {
      userID: this.getUserId(),
      movieId: this.movieId,
      fullTitle: this.movieTitle,
      image: this.movieImage,
      imDbRating: this.imdbRating.toString(),
      imDbRatingCount: this.numberOfVotes
    }

    this.watchlistService.addToWatchlist(watchlist).subscribe();
    this.isOnWatchlist = true;
  }
```

Sl. 3.34. *Metoda za dodavanje filma u listu želja.*

Na slici 3.34. prikazan je dio kôda komponente pojedinačnog filma. Ukoliko korisnik odabere gumb za dodavanje filma u listu želja poziva se metoda sa slike. Poziva se metoda iz prethodno prikazanog servisa a predaje joj se objekt čije su vrijednosti polja definirane u komponenti. Na kraju metode mijenja se varijabla koja označava da li se film nalazi u listi želja, te se gumb mijenja iz „dodaj u listu želja“ u „ukloni iz liste“. HTML kôd spomenutih gumbova prikazan je na slici 3.35.

```
<button id="watchlistButton" (click)="addToWatchlist()"
mat-raised-button color="primary" *ngIf="!this.isOnWatchlist">Add to
    watchlist</button>
<button id="watchlistButton" (click)="deleteFromWatchlist()"
mat-raised-button color="warn"
    *ngIf="this.isOnWatchlist">Remove from
    watchlist</button>
<ngx-star-rating class="rating" (ngModelChange)="addUpdateRating()"
[(ngModel)]="this.rating" [id]="'rating'">
</ngx-star-rating>
```

Sl. 3.35. *HTML kôd za dodavanje filma u listu želja i ocjenjivanje filma.*

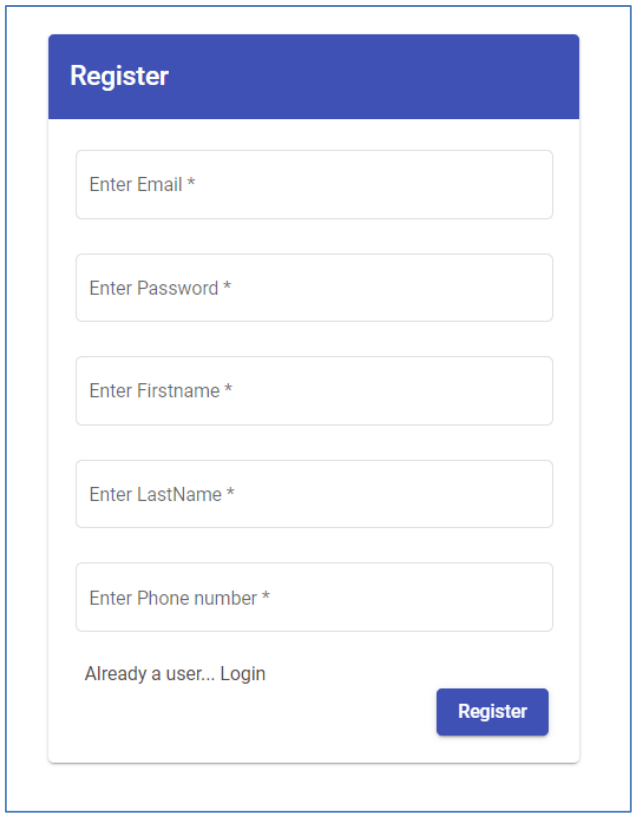
Slika 3.36. prikazuje metodu koja se poziva ukoliko korisnik odabere neki od filmova. Podaci se dohvaćaju iz metode servisa koji je prethodno prikazan a predaje mu se ID filma. Dohvaćeni podaci spremaju se naknadno u varijable definirane na početku komponente te iste služe za prikaz podataka pojedinačnog filma. Na kraju metode pozivaju se metode kojima se provjerava da li prijavljeni korisnik ima film na listi želja te da li ga je već ocijenio.

```
getMovieById(id: string) {
    this.movieService.getMovieById(id).subscribe((data) => {
      this.movieData = data;
      this.movieId = data.id;
      this.movieTitle = data.fullTitle;
      this.movieImage = data.image;
      this.moviePlot = data.plot;
      this.imdbRating = parseFloat(data.imDbRating);
      this.movieRuntime = data.runtimeStr;
      this.numberOfVotes = data.imDbRatingVotes;
      this.genre = data.genres;
      this.director = data.directors;
      this.movieAwards = data.awards;
      this.movieStars = data.stars;

      this.checkIfMovieIsOnWatchlist();
      this.checkIfMovieIsRated();
    })
  }
```

Sl. 3.36. *Kôd za dohvaćanje informacija odabranog filma.*

4. PRIKAZ RADA APLIKACIJE

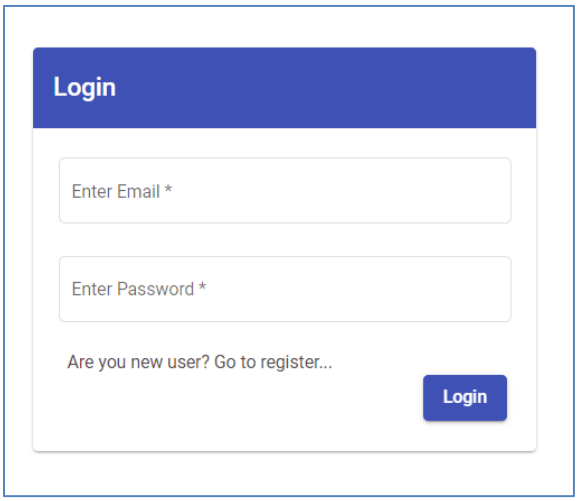
U ovom poglavlju prikazan je rad aplikacije, pojedinih komponenti i funkcionalnosti aplikacije. Izgled je uvelike inspiriran sličnim portalima koji su prikazani u 2. poglavlju. Kao i svaka aplikacija koja uključuje korisnike, osigurani su prijava i registracija. Na slici 4.1. prikazan je obrazac koji se otvara ukoliko korisnik nije registriran.



Sl. 4.1. *Obrazac za registraciju korisnika.*

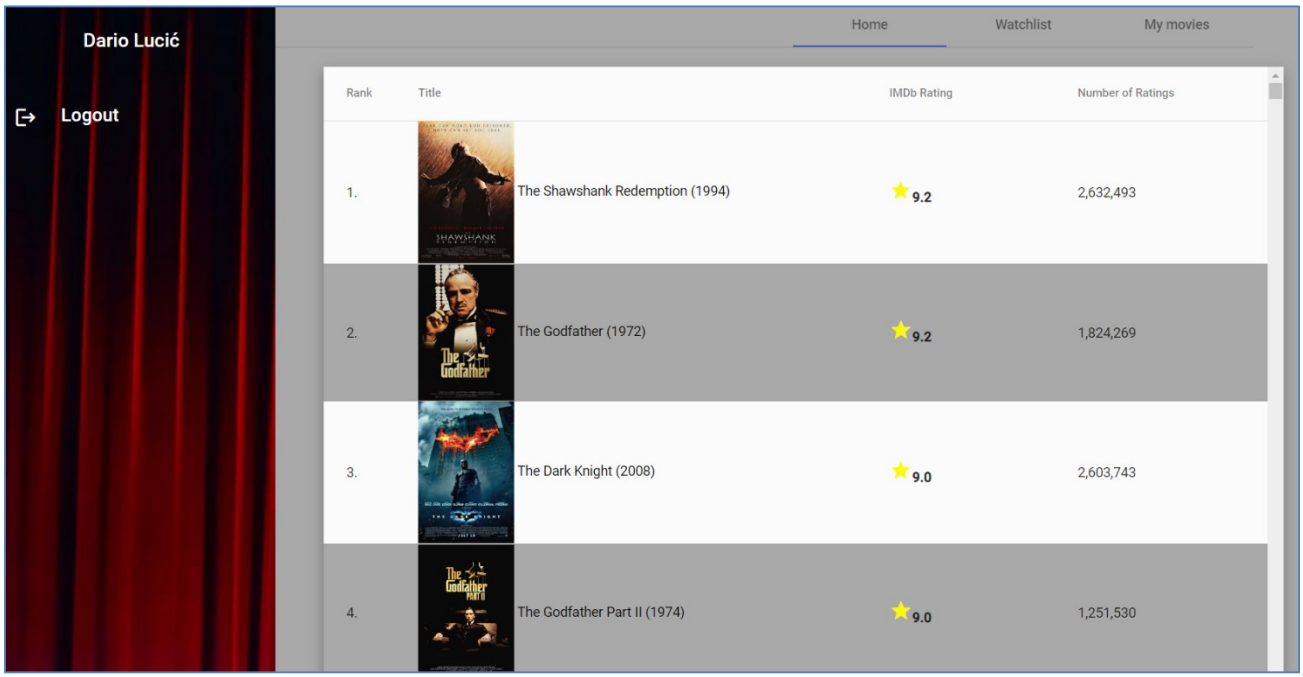
Od osobe se zahtijevaju informacije kao što su e-mail račun, lozinka, ime, prezime te broj telefona kako bi se uspješno registrirao. Ukoliko je osoba već registrirana tj. ima svoj račun na portalu, može odabrati link ispod forme koji ga preusmjerava na stranicu za prijavu.

Ukoliko je osoba već registrirana ili ukoliko odabere gumb za registraciju, preusmjerava ga se na stranicu prijave. Forma za prijavu prikazana je na slici 4.2. Korisnik unosi e-mail te lozinku i ako je prijava valjana šalje mu se token sa poslužitelja te može pristupati stranicama portala. Korisnika se nadalje preusmjerava na početnu stranicu.



Sl. 4.2. *Obrazac za prijavu korisnika.*

Ukoliko se korisnik uspješno prijavio, otvara mu se početna stranica prikazana na slici 4.3. U gornjem lijevom kutu vidi se ime i prezime prijavljenog korisnika te ikona za odjavu.



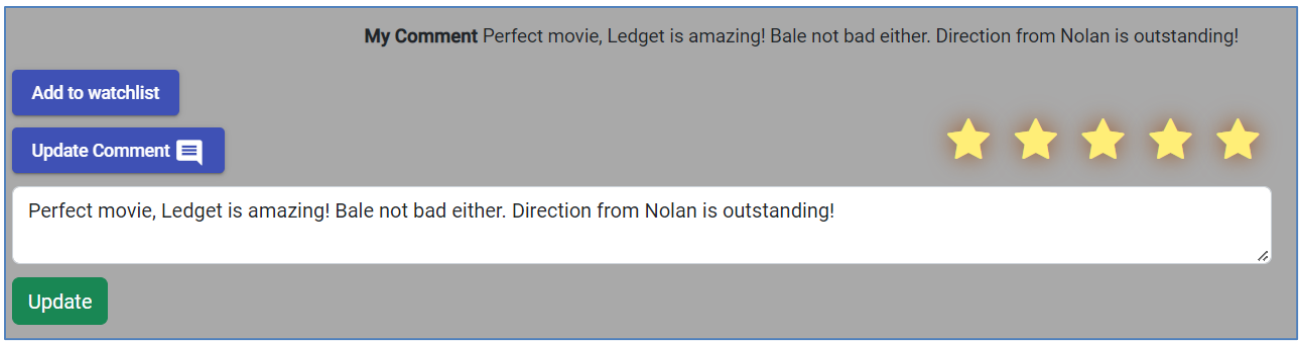
Sl. 4.3. *Početna stranica aplikacije.*

Na početnoj stranici korisniku su ponuđeni najbolje ocijenjeni filmovi sa portala IMDb, podaci koji su dohvaćeni sa API-ja portala. Korisnik može vidjeti rang filma, naslov sa posterom, godinu nastajanja filma, IMDb ocjenu te broj ljubitelja filmova koji je ocijenio prikazani naslov. Na vrhu stranice nalazi se navigacijska traka koja služi za preusmjeravanje na glavne stranice portala te je označeno na kojoj se korisnik upravo nalazi. Ukoliko korisnik odabere neki od naslova otvara mu se stranica istog koja je prikazana na slici 4.4.

Sl. 4.4. *Stranica odabranog filma.*

Na stranici pojedinog filma korisnik može vidjeti osnovne informacije o naslovu: ime, godina, poster, žanrovi, opis, redatelj, glumci, IMDb ocjena, trajanje, broj ljubitelja filmova koji je ocijenilo naslov te nagrade koje je film osvojio. Osim osnovnih informacija koje su dobivene od IMDb-a korisnik može vidjeti ocjenu koju je dao filmu (ukoliko ju je dao), svoj komentar kao i informaciju da li mu se film nalazi na listi želja. Korisniku je ponuđena i opcija ažuriranja komentara i ocjene.

Na slici 4.5. prikazan je izgled dna stranice pojedinačnog filma ukoliko korisnik klikne na gumb ažuriranje komentara.

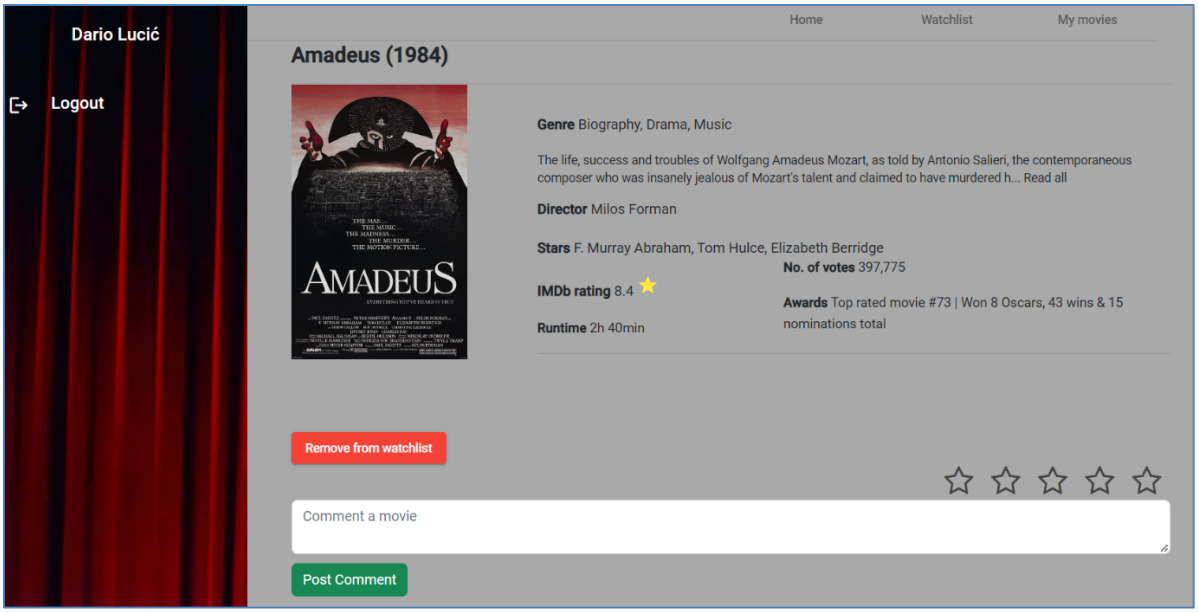


Sl. 4.5. *Ažuriranje komentara na film.*

Ukoliko korisnik doda film na listu želja isti naslov prikazat će mu se na stranici prikazanoj na slici 4.6. Ovdje se nalaze svi filmove koje korisnik želi pogledati. Tu se također nalaze neke osnovne informacije o filmu (poster, naslov, godina, ocjena).

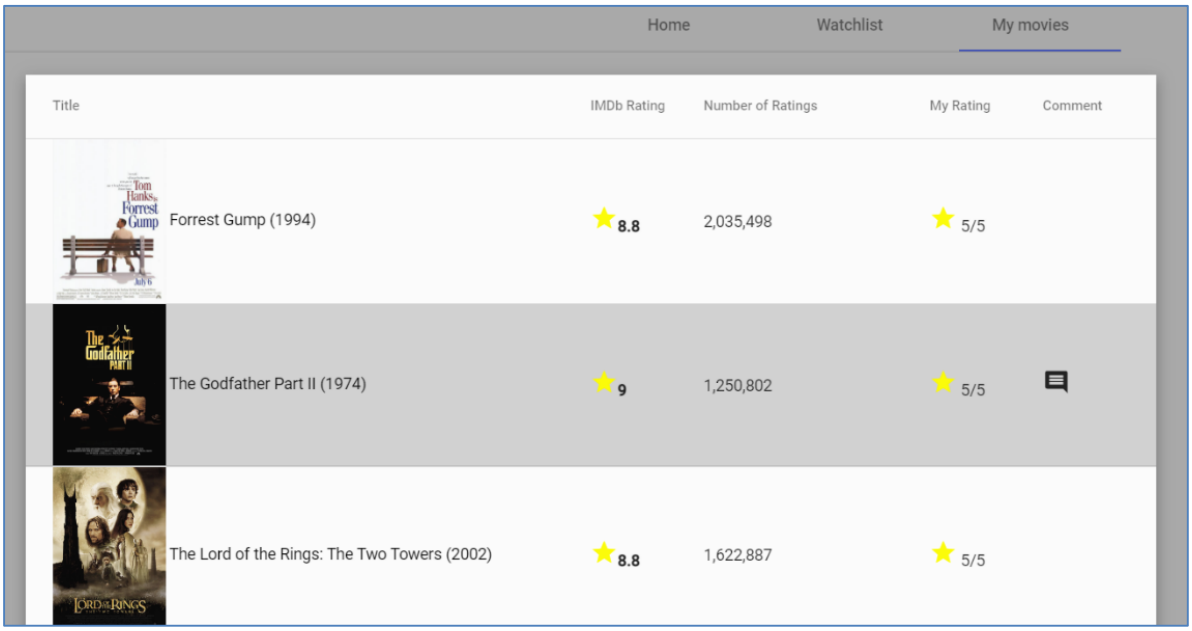
Sl. 4.6. *Izgled liste želja aplikacije.*

Ukoliko korisnik odabere neki od filmova koji se nalazi na listi, otvara mu se stranica koja je prikazana na slici 4.7. Može se vidjeti kako je ovdje sada ponuđena opcija za uklanjanje filma s liste te prazno polje na komentiranje.



Sl. 4.7. *Izgled odabranog filma koji je na listi želja.*

Ukoliko korisnik iz navigacijske trake odabere „Moji filmovi“, tu može vidjeti sve filmove koje je pogledao, odnosno ocijenio ili komentirao što je prikazano na slici 4.8. Uz osnovne, već navedene informacije o filmovima, korisnik ovdje može vidjeti ocjenu koju je dao filmu te da li je film već komentirao.



Sl. 4.8. *Izgled stranice na kojoj su već pogledani filmovi.*

5. ZAKLJUČAK

Cilj rada bio je izrada web portala za filmove gdje će registrirani ljubitelji filmova imati mogućnosti ocjenjivanja, komentiranja kao i dodavanja naslova na listu želja za gledanje. Tehnologije kojima je ostvarena potpuno funkcionalna aplikacija opisana u radu su PostgreSQL za bazu podataka, ASP.NET Core Web API za poslužiteljski dio te radni okvir Angular za klijentski dio te korisničko sučelje.

Ovakva aplikacija od velike je koristi korisnicima prilikom organizacije gledanja filmova za koje su zainteresirani. Kao što je opisano u radu, postoji par sličnih portala koji sadrže jako slične funkcionalnosti prikazane na skoro identičan način. Svi navedeni portali imaju i velik broj korisnika što je još jedan od razloga zašto je ovakav tip aplikacije potreban i koristan u svijetu današnje tehnologije i interneta.

Naravno da postoji i prostora za napredak, mogućnost da se aplikacija napravi još ugodnijom i kvalitetnijom za korisnika što bi se moglo ostvariti dodavanjem dodatnih funkcionalnosti, ali one najbitnije koje sadrži svaka aplikacija sličnog tipa već su ostvarene. Izgradnja aplikacije korištenim tehnologijama vrlo je jednostavna i fleksibilna što je prikazano u radu te dodavanje novih korisnicima zgodnih komponenti ne bi predstavljao velik problem.

LITERATURA

[1] R. Lewis, IMDb [online], Britannica, dostupno na: https://www.britannica.com/topic/IMDb [25.06.2022.]

[2] Skupina autora, About Rotten Tomatoes [online], Rotten Tomatoes, dostupno na: https://www.rottentomatoes.com/about [25.06.2022.]

[3] Skupina autora, About AllMovie [online], AllMovie, dostupno na: https://www.allmovie.com/about [25.06.2022.]

[4] Skupina autora, What is Letterboxd? [online], Letterboxd, dostupno na: https://letterboxd.com/about/faq/ [25.06.2022.]

[5] Skupina autora, About Metacritic [online], Metacritic, dostupno na: https://www.metacritic.com/about-metacritic [25.06.2022.]

[6] Skupina autora, What is PostgreSQL? [online], PostgreSQL, dostupno na: https://www.postgresql.org/about/ [26.06.2022.]

[7] K. Shah, ASP.NET Core 5.0 Web API [online], C# Corner, 2021, dostupno na: https://www.c-sharpcorner.com/article/asp-net-core-5-0-web-api/ [26.06.2022.]

[8] Skupina autora, What is Angular? [online], Angular, dostupno na: https://angular.io/guide/what-is-angular [26.06.2022.]

SAŽETAK

Naslov: Web portal za filmove

Zadatak ovog rada je kreiranje web portala za ljubitelje filmova koji će imati mogućnosti ocjenjivanja filmova, kao i komentiranja te dodavanja naslova na listu želja za gledanje.

Registrirani korisnici imaju ponuđene najbolje ocijenjene filmove te mogu voditi evidenciju o onima koje su već pogledali kao i onima koje žele pogledati. Korisnicima su omogućene najbitnije informacije o svakom filmu (godina izlaska, glumci, redatelji, poster, opis, žanrovi). Aplikacija je ostvarena u tehnologijama PostgreSQL, ASP.NET Core Web API te u radnom okviru Angular.

Ključne riječi: Angular, baza podataka, filmovi, web portal

ABSTRACT

Title: Web portal for movies

The premise of this paper is to create a web portal for movie lovers who will have the ability to rate movies, as well as comment and add titles to the watchlist.

Registered users are offered the highest rated movies and can keep track of the ones they have already watched as well as the ones they want to watch. Users are provided with the most important information about each film (year of release, actors, directors, poster, description, genres). The application was created using PostgreSQL, ASP.NET Core Web API and Angular framework.

Keywords: Angular, database, movies, web portal