Web aplikacija za rezervaciju sjedala u kinu

Štefanac, Karlo

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

Sveučilišni prijediplomski studij Računarstvo

Web aplikacija za rezervaciju sjedala u kinu

Završni rad

Karlo Štefanac

Osijek, 2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1P: Obrazac za ocjenu završnog rada na sveučilišnom prijediplomskom studiju

Ocjena završnog rada na sveučilišnom prijediplomskom studiju

Ime i prezime pristupnika:	Karlo Štefanac
Studij, smjer:	Sveučilišni prijediplomski studij Računarstvo
Mat. br. pristupnika, god.	R4723, 28.07.2021.
JMBAG:	0165091985
Mentor:	izv. prof. dr. sc. Mirko Köhler
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Web aplikacija za rezervaciju sjedala u kinu
Znanstvena grana završnog rada:	**Obradba informacija (zn. polje računarstvo)**
Zadatak završnog rada:	Tema rezervirana za: Karlo Štefanac. Potrebno je napraviti web aplikaciju za upravljanje kino dvoranama. Administrator može dodati predstave i povezati ih s dvoranama u kojima će biti prikazivanje. Korisnici mogu odabrati film i sjedalo u dvorani.

Datum prijedloga ocjene završnog rada od strane mentora:	17.09.2024.
Prijedlog ocjene završnog rada od strane mentora:	Vrlo dobar (4)
Datum potvrde ocjene završnog rada od strane Odbora:	25.09.2024.
Ocjena završnog rada nakon obrane:	Vrlo dobar (4)
Datum potvrde mentora o predaji konačne verzije završnog rada čime je pristupnik završio sveučilišni prijediplomski studij:	26.09.2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O IZVORNOSTI RADA

Osijek, 26.09.2024.

Ime i prezime Pristupnika:	Karlo Štefanac
Studij:	Sveučilišni prijediplomski studij Računarstvo
Mat. br. Pristupnika, godina upisa:	R4723, 28.07.2021.
Turnitin podudaranje [%]:	8

Ovom izjavom izjavljujem da je rad pod nazivom**: Web aplikacija za rezervaciju sjedala u kinu**

izrađen pod vodstvom mentora izv. prof. dr. sc. Mirko Köhler

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

Filmska industrija je jedan od najprofitabilnijih oblika zabave. Globalna zarada od prodaje ulaznica za kinodvorane za 2023. godinu je čak 33,9 milijardi dolara, te je svake godine sve veća i veća. Iz tog razloga potrebno je korisnicima pružiti što jednostavniji i intuitivniji način za kupnju ulaznica za svoje najdraže nove filmove. Gotovo svaki grad danas ima svoju kinodvoranu koja prikazuje najnovije filmove te je za svaku kinodvoranu potrebno omogućiti kupovinu ulaznica putem interneta.

Razvijeniji gradovi i lanci kinodvorana već imaju dobro razvijene proizvode za vođenje kinodvorana, no manji gradovi i kulturni centri nemaju potrebnu za istim rješenjima te oni mogu preuzeti rješenje opisano u ovome radu za svoje potrebe.

Uz uvod, ovaj rad će kroz pet poglavlja prikazati proces izrade web aplikacije za rezervaciju sjedala u kinodvorani. Prvo poglavlje će prikazati već postojeće proizvode koje pružaju slične usluge. Drugo poglavlje će opisati korištene tehnologije potrebne za izradu web aplikacije. Treće poglavlje će prikazati i objasniti način na koji je web aplikacija napravljena. Četvrto poglavlje će prikazati korisničko sučelje te način rada web aplikacije. Peto poglavlje predstavlja zaključak ovog rada.

1.1 Zadatak završnog rada

Zadatak završnog rada je izrada web aplikacije koja će kinodvoranama omogućiti postavljanje termina prikazivanja filmova te će korisnicima omogućiti rezervaciju sjedala u tim kinodvoranama. Vlasnik kinodvorane može definirati dvorane u kojima će se prikazivati filmovi, dodavati filmove koje ima u ponudi te postaviti termine u kojima će se određeni filmovi prikazivati. Korisnik može stvoriti svoj račun, odabrati film koji želi gledati, termin u kojemu ga želi gledati te može rezervirati željeno slobodno sjedalo unutar dvorane. Cilj je omogućiti manjim kinodvoranama (kinodvorane manjih gradova ili privatne kinodvorane) pouzdano sučelje za rezervaciju ulaznica putem interneta.

2. PREGLED PODRUČJA TEME

Ovo poglavlje će obaviti pregled postojećih rješenja koje postižu sličan cilj kao i web aplikacija opisana u ovome radu. Navedena su četiri postojeća rješenja.

2.1 CINEsync

CINEsync je softver kao servis (engl. Software as a Service) rješenje za kinodvorane. Pruža web i mobilne aplikacije za kinodvorane koje se brine za osobnu te online kupovinu ulaznica. Vlasnicima pruža jednostavan način za postavljanje termina te im također pruža prikaz analitike njihovog poslovanja, kao što su prihodi. Također omogućava kupovinu pića i hrane te posebnu

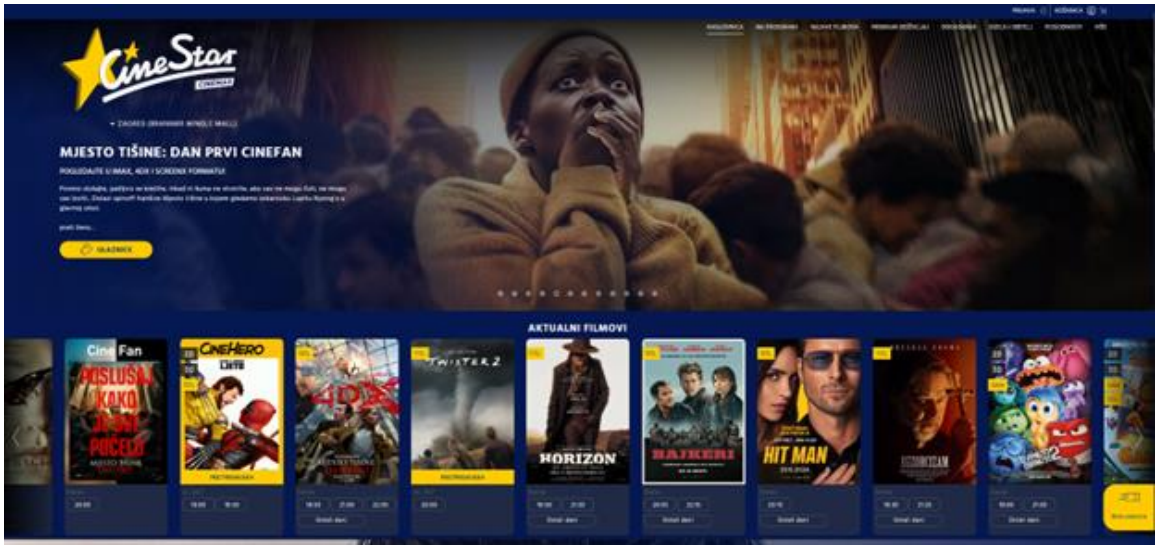
Sl. 2.1. CINEsync

aplikaciju za skeniranje ulaznica koje CINEsync generira. Ovaj projekt je najsličniji web aplikaciji koja je opisana u ovome radu. Znatna razlika je u mogućnosti prikaza analitike poslovanja te u mogućnosti kupovine hrane i pića, što web aplikacija opisana u ovome radu ne omogućuje.

2.2 Cinestar

Cinestar je najveći lanac kinodvorana u Hrvatskoj te je rasprostranjen u 11 gradova. Također je rasprostranjen i izvan Hrvatske. Cinestar ima svoju web aplikaciju za upravljanje svih kina te njihovih posebnih dvorana. Preko glavne stranice je moguće vidjeti sva moguća kina te njima pristupiti i vidjeti sve filmove koji se prikazuju. Na glavnoj stranici odabrane kinodvorane se

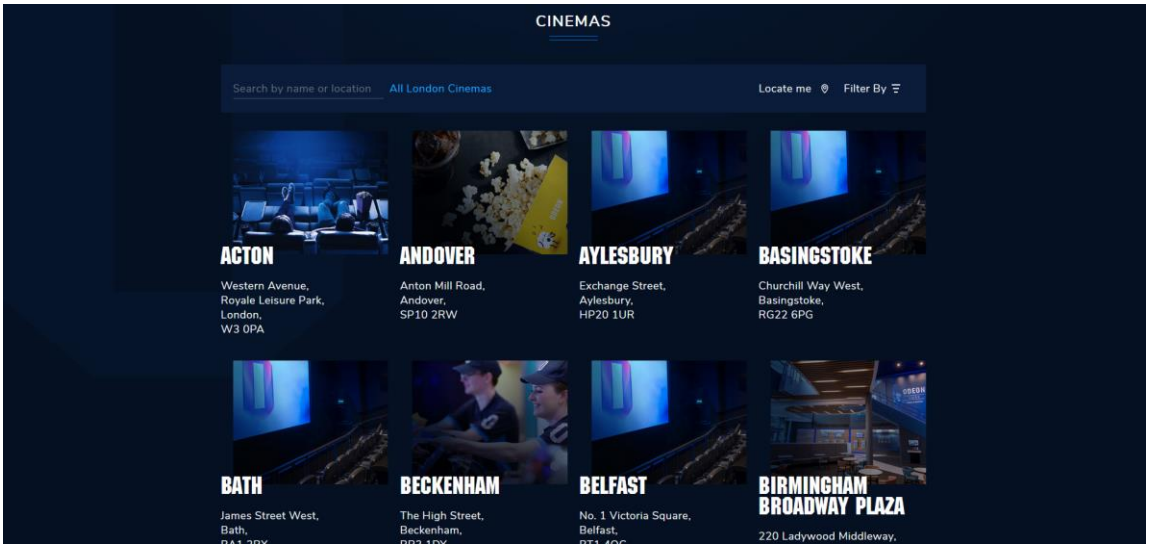
nalaze aktualni filmovi te slobodni termini za današnji dan. Nakon odabira termina prikazuje se zaslon na kojemu su prikazani sva sjedala te prikaz njihove dostupnosti. Ako je sjedalo dostupno možemo ga odabrati te nas odabirom na sjedalo vodi na način plaćanja.



Sl. 2.2. Cinestar

2.3 Odeon

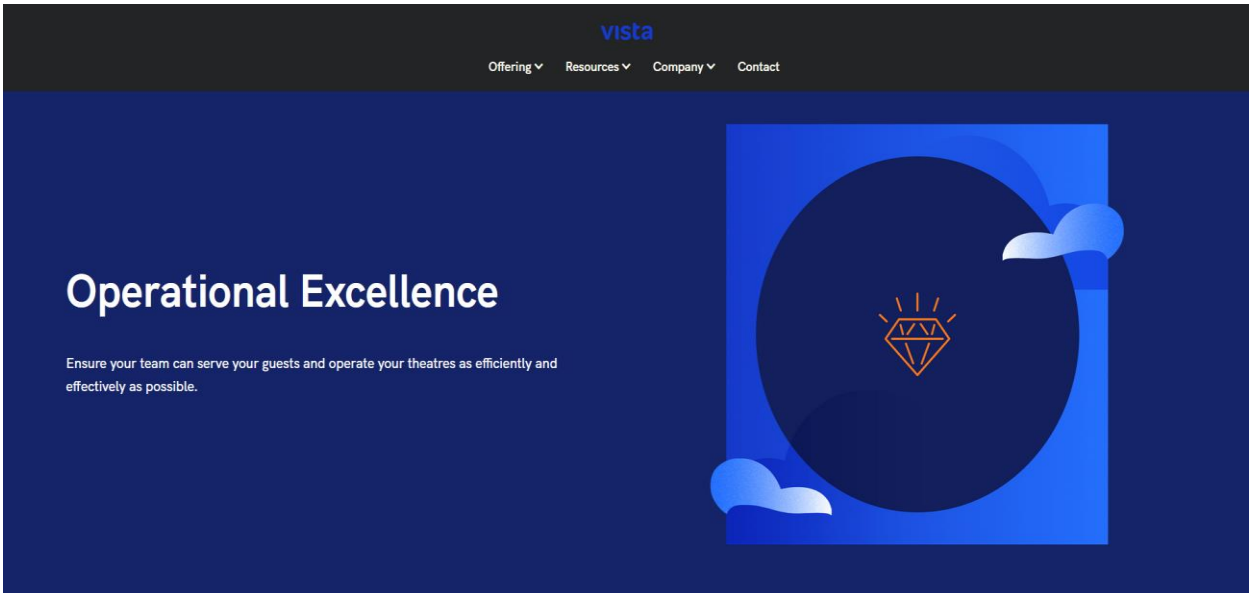
Odeon je vodeći lanac kinodvorana u Ujedinjenom Kraljevstvu i Irskoj. Isto kao i Cinestar, ima svoju web aplikaciju na kojoj se nalaze sva kina kojima oni upravljaju. Proces rezervacije sjedala je također isti kao i za Cinestar, odabirom filma prikažu sve svi dostupni termini. Odabirom termina prikazana su dostupna sjedala te nakon odabira sjedala slijedi proces plaćanja.



Sl. 2.3. Odeon

2.4 Vista Cinema

Osnovana 1996. godine u Aucklandu, Vista Cinema pruža korisne alate potrebne za upravljanje vlastitom kinodvoranom. Vista Cinema omogućuje upravljanje različitim dvoranama te lokacijama koje vlasnik posjeduje te jednostavno postavljanje termina za filmove koji su dostupni za prikaz. Pružaju web i mobilno rješenje za kupce te pouzdan i detaljan prikaz podataka o transakcijama za vlasnika. Slično kao i prethodno navedeni CINEsync, također pružaju mogućnost za kupovinu hrane i pića unutar same kinodvorane.



Sl. 2.3. Vista Cinema

3. KORIŠTENE TEHNOLOGIJE

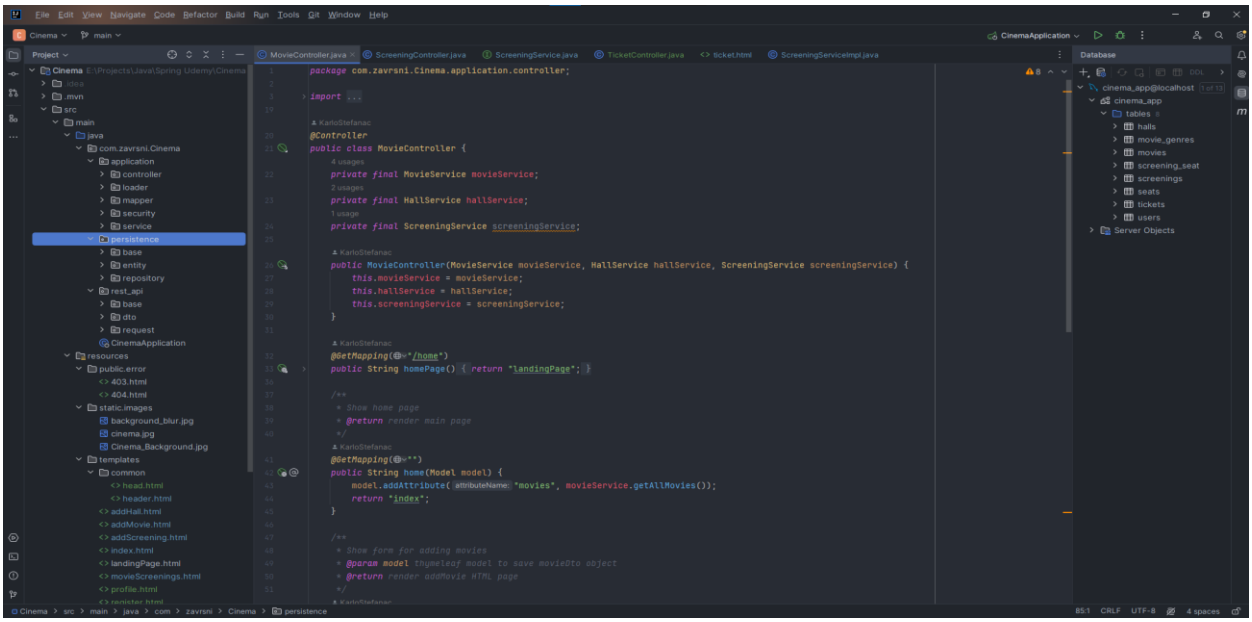
Za izradu ove web aplikacije korišten je Spring Boot okvir za programski jezik Java, a pisan je u IntelliJ IDEA razvojno okruženje. Povezivanje koda u Javi sa HyperText Markup Language (HTML) kodom postignuto je pomoću Thymeleaf upravitelja za podloge. Za izradu i upravljanje bazom podataka korišten je SQLite upravitelj baze podataka.

3.1 Spring Boot

Spring Boot je okvir temeljen na programskom jeziku Java koji pojednostavljuje razvoj samostalnih aplikacija spremnih za proizvodnju. Nadovezuje se na Spring okvir dodavanjem zadanih konfiguracija, ugrađenih poslužitelja i unaprijed izgrađenih funkcionalnosti koje omogućuju brže kreiranje i implementiranje web aplikacija. Spring Boot također podržava funkcionalosti kao što su ubrizgavanje ovisnosti, automatske konfiguracije i bogat ekosustav integracija. Sve to čini Spring Boot popularnim izborom za izradu modernih web aplikacija.

3.2 IntelliJ IDEA

IntelliJ IDEA je integrirano razvojno okruženje za Java programski jezik koje također podržava Spring Boot razvojni okvir. Ovaj uređivač teksta nudi automatsko dovršavanje koda, analizu koda



Sl. 3.1. IntelliJ IDEA razvojno okruženje

u stvarnom vremenu, pronalaženje pogrešaka u sintaksi, alate za refaktoriranje te još razne alate koji znatno olakšavaju i ubrzavaju proces izgradnje web aplikacije.

3.3 Thymeleaf

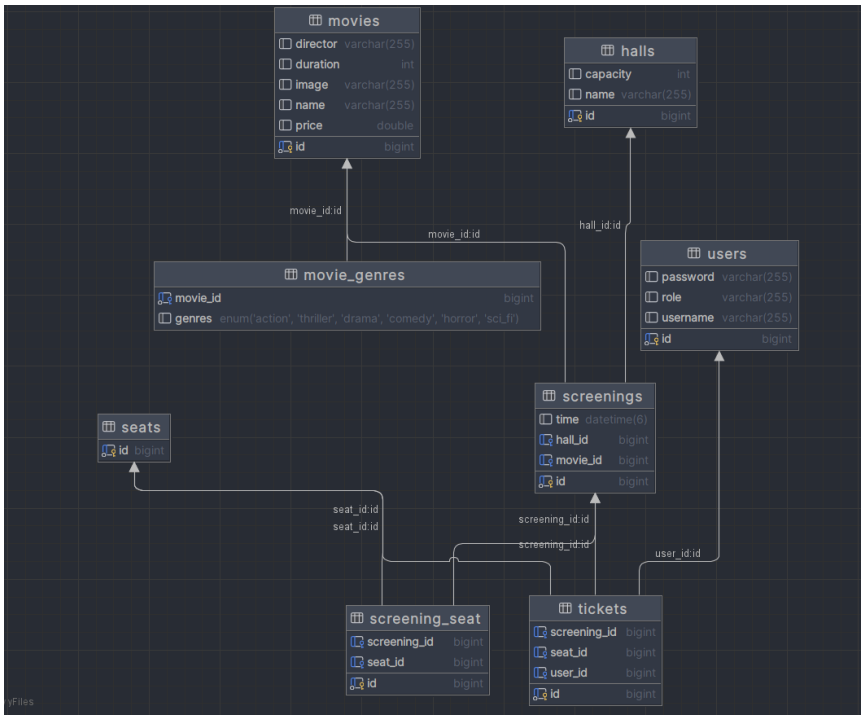
Thymeleaf je upravitelj predložaka temeljen na Java programskom jeziku koji podržava izradu dinamičnih web stranica u Spring Boot aplikacijama. Omogućuje pisanje HTML koda s ugrađenom logikom koja se obrađuje na poslužiteljskoj strani kako bi se generirao određeni sadržaj.

3.4 MySQL

MySQL je sustav za upravljanje relacijskim bazama podataka koji koristi strukturirani jezik upita (engl. Structured Query Language) za upravljanje sa podacima. Radi po klijent-poslužitelj modelu gdje MySQL poslužitelj radi kao zaseban proces i obrađuje više zahtjeva za bazama podataka od klijenata, poput web aplikacija. Prednost MySQL-a je što je njegov kod javno dostupan (engl. open source) te je konstantno u razvoju zadnjih 30 godina.

4. IZRADA WEB APLIKACIJE

Ovo poglavlje će prikazati korake koji su bili potrebni za izradu ove web aplikacije. Prvi korak je modeliranje baze podataka koja će pohranjivati i upravljati podacima koje web aplikacija koristi. Slika 4.1. prikazuje dijagram koji predstavlja model baze podataka korišten za ovu web aplikaciju.



Sl. 4.1. Model baze podataka

Model se sastoji od osam entiteta koji su međusobno povezani na različite načine. Za svaki entitet stvoreni su kontroleri (engl. controller) i servisi (engl. service) koji definiraju ponašanje svakog entiteta.

4.1 Entiteti

Svaki entitet sadrži određene atribute koji opisuju taj entitet, a svaki atribut ima određeni tip podataka koji ga opisuje. U Spring Boot okruženju entitete definiramo sa klasama, a atribute sa klasičnim tipovima podataka iz Java programskog jezika.

Tako programski kod 4.2. definira entitet za film koji se može prikazivati u kinu. Svaki film je opisan imenom filma, skupom žanrova (opisan enum tipom podatka), imenom redatelja, duljinom filma u minutama, cijenom ulaznice za film te slikom koja prikazuje film.

Linija ***Kod***

```
1:      @Table(name = "movies")
2:      public class MovieEntity extends BaseEntity {
3:      @Column(name = "name")
4:      private String name;
5:      @ElementCollection(targetClass = Genre.class, fetch =FetchType.LAZY)
6:      @CollectionTable(name = "movie_genres", joinColumns = @JoinColumn
        (name = "movie_id", referencedColumnName = "id",foreignKey =
        @ForeignKey(name = "fk_movie")))
7:      @Enumerated(EnumType.STRING)
8:      private List<Genre> genres;
9:      @Column(name = "director")
10:     private String director;
11:     @Column(name = "duration")
12:     private int duration;
13:     @Column(name = "price")
14:     private double price;
15:     @Column(name = "image")
16:     private String image;
17:     @OneToMany(mappedBy = "movie", cascade = CascadeType.ALL,
        orphanRemoval = true)
18:     private List<ScreeningEntity> screenings; }
```

Sl. 4.2. Entiet za film

Zadnji atribut u programskom kodu 4.2. prikazuje listu termina za film, no taj podatak se ne pohranjuje u entitetu za film, već u entitetu za termine u obliku stranog ključa (engl. foreign key). Svaki film se može prikazivati u više termina istovremeno, zbog čega je relacija između entiteta za film i termin jedan naprema više (engl. One to Many). Iz tog razloga svaki entitet za termin sadrži polje u kojem se pohranjuje identifikacijski broj filma koji se prikazuje u tom terminu.

Uz atribut za film koji se prikazuje u tom terminu, entitet za termin je još opisan vremenom u kojem će se prikazivati te dvoranom za koju je termin određen, vidljivo u programskom kodu 4.3. Kao i film, dvorana je također pohranjena u obliku identifikacijske oznake za određenu dvoranu zbog relacije jedan naprema više između ta dva entiteta (ista dvorana se može koristiti za više termina). Svaka dvorana je definirana određenim kapacitetom te imenom dvorane.

Svakom terminu pridodana je lista sjedala koja su dostupna za taj termin. Svako sjedalo je definirano samo njegovim identifikacijskim brojem, a relacija između termina i sjedala je definirana relacijom više naprema više (engl. Many to Many). Razlog tome je taj što je lista sjedala

unaprijed definirana zbog očuvanja broja podataka za pohranu te se svako pohranjeno sjedalo u isto vrijeme može koristiti za više termina.

Linija ***Kod***

```
1:      @Table(name = "screenings")
2:      public class ScreeningEntity extends BaseEntity {
3:      @DateTimeFormat(iso = DateTimeFormat.ISO.DATE_TIME)
4:      @Future
5:      private LocalDateTime time;
6:      @ManyToOne
7:      @JoinColumn(name="hall_id", foreignKey = @ForeignKey(name =
        "fk_screening_hall"))
8:      private HallEntity hall;
9:      @OneToMany(mappedBy = "screening", cascade = CascadeType.ALL,
        orphanRemoval = true)
10:     private List<ScreeningSeatEntity> screeningSeats
11:     @ManyToOne
12:     @JoinColumn(name = "movie_id", foreignKey = @ForeignKey(name =
        "fk_screening_movie"))
13:     private MovieEntity movie;
14:     @OneToMany(mappedBy = "screening", cascade = CascadeType.ALL,
        orphanRemoval = true)
15:     private List<TicketEntity> tickets; }
```

Sl. 4.3. Entitet za termin

Linija ***Kod***

```
1:      @Table(name = "screening_seat")
2:      public class ScreeningSeatEntity extends BaseEntity {
3:      @ManyToOne
4:      @JoinColumn(name = "screening_id", foreignKey = @ForeignKey(name =
        "fk_screening_seat"))
5:      private ScreeningEntity screening;
6:      @ManyToOne
7:      @JoinColumn(name = "seat_id", foreignKey = @ForeignKey(name =
        "fk_seat_hall"))
8:      private SeatEntity seat; }
```

Sl. 4.4. Entitet za relaciju više naprema više između termina i sjedala

Relacija više naprema više se definira sa novim entitetom koji pohranjuje identifikacijske oznake međusobno povezanih entiteta, kao što je prikazano u programskom kodu 4.4.

Model prikazan slikom 4.1. prikazuje još jednu relaciju sa entitetom za termine, a to je relacija jedan naprema više sa entitetom za ulaznice. Osim polja za identifikacijsku oznaku, svako polje entiteta za ulaznicu je identifikacijska oznaka drugih entiteta. Za svako sjedalo koje korisnik rezervira, stvara se nova ulaznica koja se pridodaje korisniku za taj termin. Zbog toga entitet za ulaznice ima relacije više naprema više sa entitetima za korisnika, termin te sjedalo koje je korisnik rezervirao.

Zadnji entitet koji opisuje ovaj web projekt je entitet za korisnika. Svaki korisnik sadrži njegovo ime, prezime, korisničko ime, lozinku i korisničke uloge. Korisnik može posjedovati običnu korisničku ulogu, koja omogućuje samo rezervaciju sjedala te administratorsku ulogu koja je dodijeljena samo vlasniku kinodvorane. Administratorska uloga dopušta vlasniku dodavanje filmova, dvorana te termina za prikaz.

4.2 Servisi

Servisi su odgovorni za obradu podataka, obradu složenih operacija i interakciju sa slojem za pristup podacima (baza podataka). Spring Boot aplikacije održavaju čistu podjelu odgovornosti što čini kod lakšim za održavanje, testiranje i skaliranje. Servisi će nam služiti za ispunjavanje funkcionalnih zahtjeva koji dolaze sa izradom ove web aplikacije. Kontroleri pozivaju metode servisa kako bi izvršili određene radnje. Svakom entitetu dodijeljen je vlastiti servis koji će obrađivati podatke tog entiteta te pružiti kontroleru potrebne podatke za prikaz na korisničkom sučelju.

Servis za film entitet mora definirati implementaciju za svaku operaciju vezanu uz filmove, a te operacije su:

- dodavanje filma u bazu podataka
- prikaz svih filmova
- dohvaćanje odabranog filma.

Navedene operacije zahtijevaju samo osnovnu interakciju sa slojem za pristup podacima te je njihova implementacija prikazana u programskom kodu 4.5. Svi argumenti koje servis koristi su

pruženi od strane kontrolera, kao što je argument za identifikacijsku oznaku filma u metodi za dohvaćanje odabranog filma (metoda „getMovie“).

Linija ***Kod***

```
1:      @Override
2:      public void addMovie(MovieRequest movieRequest) {
3:      movieRepository.save(movieMapper.mapToMovie(movieRequest)); }
4:      @Override
5:      public List<MovieDto> getAllMovies() {
6:      List<MovieEntity> movies = movieRepository.findAll();
7:      return movies.stream().map(movieMapper::mapToMovieDto).toList(); }
8:      @Override
9:      public MovieDto getMovie(Long id) {
10:     return movieMapper.mapToMovieDto(movieRepository.findMovieById(id));
        }
```

Sl. 4.5. Servis za entitet film

Prilikom odabira filma, korisniku se prikazuju svi termini koji su dostupni za taj film. Servis za entitet termin sadrži rješenje te funkcionalnosti u obliku metode koja prima identifikacijsku oznaku filma kao argument pomoću kojeg nalazi sve termine koji su povezani sa odabranim filmom. Osim toga, servis za entitet termin implementira mogućnost pohranjivanja same rezervacije koju korisnik obavi za odabrani termin, kao što je prikazano u programskom kodu 4.6.

Linija ***Kod***

```
1:      @Override
2:      public ScreeningDto addReservation(Long screeningId,
                                           List<Long> seatIds) {
3:      ScreeningEntity screening = screeningRepository.findById
                (screeningId).orElseThrow(RuntimeException::new);
4:      List<SeatEntity> seats = seatRepository.findAllById(seatIds);
5:      for (SeatEntity seat : seats){
6:      ScreeningSeatEntity screeningSeat = new ScreeningSeatEntity();
7:      screeningSeat.setScreening(screening);
8:      screeningSeat.setSeat(seat);
9:      screeningSeatRepository.save(screeningSeat); }
10:     return screeningMapper.mapToScreeningDto(screening);
        }
```

Sl. 4.6. Kod za pohranjivanje rezervacije

Prilikom odabira željene rezervacije, kontroler poziva metodu za prikazivanje dostupnih sjedala iz servisa za entitet sjedalo. Nakon odabira željenih sjedala, lista odabranih sjedala se prosljeđuje kao

argument metodi za pohranjivanje rezervacije prikazanoj na slici **Error! Reference source not found.** Za svako odabrano sjedalo pohranjuje se objekt za entitet stvoren relacijom više naprema više između entiteta termin i sjedalo.

Nakon što je korisnik odabrao željeni termin i sjedala koja želi rezervirati potrebno mu je dodijeliti ulaznice. Za to je zaslužan servis za entitet ulaznice koji sadrži samo dvije metode:

- dodjeljivanje ulaznica korisniku
- dohvaćanje svih ulaznica koje je korisnik kupio

Implementacija tih metoda je prikazana u programskom kodu 4.7.

Linija ***Kod***

```
1:      @Override
2:      public void addTickets(List<Long> seatIds, Long screeningId,
                               String username) {
3:      for (Long seat : seatIds){
4:      TicketEntity ticket = new TicketEntity();
5:      ticket.setScreening(screeningRepository.findById(screeningId)
        .orElseThrow(RuntimeException::new));
6:      ticket.setUser(userRepository.findByUsername(username)
        .orElseThrow(RuntimeException::new));
7:      ticket.setSeat(seatRepository.findById(seat)
        .orElseThrow(RuntimeException::new));
8:      ticketRepository.save(ticket); }
9:      }
10:     @Override
11:     public List<TicketDto> getUserTickets(UserDto userDto) {
12:     UserEntity user = userRepository.findById(userDto.getId())
        .orElseThrow(RuntimeException::new);
13:     List<TicketEntity> tickets = user.getTickets();
14:     return tickets.stream().map(ticketMapper::mapToTicketDto).toList();
15:     }
```

Sl. 4.7. Servis za entitet ulaznica

Servis za entitet termin također omogućuje osobi sa administratorskim pravima dodavanje novih termina za prikaz odabranog filma. Kontroler je zaslužan servisu proslijediti sve podatke potrebne za dodavanje novog termina (datum i vrijeme prikazivanja, film koji se prikazuje i dvorana).

Servis za entitet dvorana je zaslužan za dohvaćanje svih pohranjenih dvorana, koje se prikazuju prilikom dodavanja novog termina te za pohranjivanje nove dvorane od strane korisnika sa administratorskim pravima.

Posljednji servis je dodijeljen entitetu korisnik. Ovaj servis je zaslužan za dodavanje novog korisnika u bazu podataka nakon registracije te za dohvaćanje podataka o trenutno prijavljenom korisniku. Implementacije ovih zahtjeva su vidljive u programskom kodu 4.8.

Linija ***Kod***

```
1:      @Override
2:      public UserDto addUser(UserRequest userRequest) {
3:      UserEntity user = userMapper.mapToUser(userRequest);
4:      user.setPassword(passwordEncoder.encode(user.getPassword()));
5:      if (userRequest.isStaff()){
6:      user.setRole("USER,ADMIN"); }
7:      else{
8:      user.setRole("USER"); }
9:      return userMapper.maptoUserDto(userRepository.save(user)); }
10:     @Override
11:     public UserDto getUser(String username) {
12:     UserEntity userEntity = userRepository.findByUsername(username)
        .orElseThrow(RuntimeException::new);
13:     return userMapper.maptoUserDto(userEntity);
15:     }
```

Sl. 4.8. Servis za entitet korisnik

4.3 Kontroleri

U Spring Bootu kontroleri su ključne komponente odgovorne za obradu HTTP (HyperText Transfer Protocol) zahtjeva od klijenta, te vraćanje odgovora. Pomoću kontrolera definira se REST API (Representational State Transfer Application Programming Interface) koji omogućuje izlaganje podataka i funkcionalnosti putem HTTP-a tako što se određenom URL-u (Uniform Resource Locator) dodjeljuje metoda koja prima HTTP zahtjeve kao što su:

- HTTP GET za dohvaćanje resursa
- HTTP POST za stvaranje novih resursa
- HTTP PUT za ažuriranje postojećih resursa
- HTTP DELETE za brisanje resursa.

Korištenjem kontrolera određuje se funkcija koja će se obaviti prilikom učitavanja određenog URL-a. Cilj je korisniku pružiti intuitivne poveznice pomoću kojih će koristiti web aplikaciju prema svojim željama. Svakom entitetu pridružen je poseban kontroler koji obrađuje zahtjeve vezane za podatke o tom entitetu, koji se prosljeđuju servisu za daljnju obradu podataka.

Kontroler za entitet film pruža mogućnosti za:

- prikazivanje svih filmova
- prikazivanje forme za dodavanje novog filma
- pohranjivanje novog filma

Programski kod 4.9. prikazuje implementaciju navedenih operacija. Posjećivanjem početnog URL-a šalje se HTTP GET metoda koja vrši operaciju za prikazivanje svih filmova (metoda „home“) nakon čega se poziva servis za entitet film. Servis vraća listu svih pohranjenih filmova te se ta lista dodaje u model i prosljeđuje na zaslon koji prikazuje sve filmove.

Linija ***Kod***

```
1:      @GetMapping("")
2:      public String home(Model model) {
3:      model.addAttribute("movies", movieService.getAllMovies());
4:      return "index"; }
5:      @GetMapping("/movie")
6:      public String showMovieForm(Model model) {
7:      MovieRequest movieRequest = new MovieRequest();
8:      model.addAttribute("movie", movieRequest);
9:      return "addMovie"; }
10:     @PostMapping("/movie/save")
11:     public String saveMovie(@ModelAttribute("movie") MovieRequest
                                movieRequest) {
12:     movieService.addMovie(movieRequest);
13:     return "redirect:/movie?success";
15:     }
```

Sl. 4.9. Kontroler za entitet film

Pomoću Thymeleaf-a moguće je pristupiti proslijeđenim podacima unutar HTML koda i prikazati te podatke. Programski kod 4.10. prikazuje način pristupa proslijeđenim podacima kako bi se prikazali detalji o svim postojećim filmovima.

Linija ***Kod***

```
1:      <!DOCTYPE html>
2:      <html lang="en" xmlns:th="http://www.thymeleaf.org">
3:      <head th:replace="~{common/head.html :: headFragment}">
4:      <title>YourCinema</title>
5:      </head>
6:      <header th:insert="~{common/header.html :: header}"> </header>
7:      <body> <br>
8:      <div class="container">
9:      <div class="row">
10:     <div class="col-lg-3" th:each="movie : ${movies}">
11:     <div class="card bg-light"  style="max-height: 600px">
12:     <div class="card-header">
13:     <a class="link-dark" th:href="@{'movie/' + ${movie.id}}">
15:     <h4 class="text-center" th:text="${movie.name}"></h4></a>
16:     </div>
17:     <img th:src="${movie.image}" th:alt="${movie.name}"
                                    style="min-height: 400px">
18:     <div class="card-body m-1 p-0">
19:     <p class="m-1 p-0 text-center" th:text="${movie.director}"></p>
20:     <p class="m-1 p-0 text-center">
21:     <p class="m-1 p-0 text-center"><span th:each="genre :
        ${movie.genres}" th:text="${genre} + ' '"></span></p>
22:     <p class="m-1 p-0 text-center" th:text="'Duration: ' +
        ${movie.duration} + 'min'"></p>
23:     <p class="m-1 p-0 text-center" th:text="'Price: ' +
        ${movie.price} + '€'"></p>
24:     </div></div></div></div></div>
25:     </body>
26:     </html>
27:
```

Sl. 4.10. HTML zaslon za prikazivanje svih filmova

Metoda „showMovieForm“ prikazana u programskom kodu 4.9. prikazuje zaslon koji sadrži formu za upisivanje podataka o novom filmu. Nakon što vlasnik doda novi film odveden je na „/movie/save“ URL koji prosljeđuje podatke o novom filmu metodi za spremanje filma unutar servisa za entitet film.

Kontroler za entitet termin omogućuje:

- prikazivanje svih termina za odabrani film

- dodavanje novog termina za odabrani film
- rezervaciju sjedala za odabrani termin

Programski kod 4.11. implementira mogućnost prikazivanja svih termina za određeni film. Unutar URL-a nalazi se identifikacijska oznaka filma za koji prikazujemo sve termine. Tako bi za film sa identifikacijskom oznakom 12, URL „/movie/12“ prikazivao sve termine za taj film.

Linija ***Kod***

```
1:      @GetMapping("/movie/{id}")
2:      public String showMovieScreenings(@PathVariable Long id,
                                          Model model){
3:      List<ScreeningDto> screenings = movieService
        .getMovieScreenings(id);
4:      model.addAttribute("screenings", screenings);
5:      model.addAttribute("id", id);
6:      return "movieScreenings";
7:      }
```

Sl. 4.11. Kontroler za entitet termin – prikaz termina

Jednako kao i za film, prilikom dodavanja novog termina prikazuje se zaslon sa formom unutar koje se upisuju podaci o novom terminu, nakon čega se učitava URL („/movie/12/screening“) koji pokreće HTTP POST metodu za spremanje tog termina. Prilikom prikaza forme prosljeđuju se sve dostupne dvorane (pomoću servisa za entitet dvorana) kako bi vlasnik mogao jednostavno odabrati dvoranu unutar koje će se održati termin.

Kontroler za entitet termin također omogućuje prikaz i odabir svih mogućih sjedala koja su slobodna ili zauzeta. Programski kod 4.12. prikazuje metodu „showScreeningReservation“ koja pomoću servisa dohvaća listu svih zauzetih sjedala unutar tog termina. Pomoću te liste moguće je onemogućiti odabir tih sjedala unutar zaslona za prikaz svih sjedala.

Metoda „saveScreeningReservation“ unutar istog programskog koda prikuplja sve podatke o trenutnoj rezervaciji (film, termin, odabrana sjedala) te stvara prikaz stvorenih ulaznica za svako odabrano sjedalo. Ulaznice i sjedala koja su trenutno odabrana neće biti pohranjena sve dok korisnik ne potvrdi svoju transakciju. Nakon potvrde transakcije šalje se HTTP POST metoda na

kontroler za entitet ulaznica na URL „/ticket/save“, nakon čega se ulaznice dodjeljuju korisniku i pohranjuju u bazu podataka pomoću servisa, kao što je prikazano u programskom kodu 4.13.

Linija ***Kod***

```
1:      @GetMapping("/screening/{screeningId}")
2:      public String showScreeningReservation(@PathVariable
                            Long screeningId, Model model) {
3:      ScreeningDto screeningDto = screeningService
        .getScreening(screeningId);
4:      List<Integer> seatList = screeningService.getSeats(screeningId);
5:      model.addAttribute("id", screeningId);
6:      model.addAttribute("hallDto", screeningDto.getHallDto());
7:      model.addAttribute("seatList", seatList);
8:      return "screeningReservation"; }

9:      @PostMapping("/screening/ticket")
10:     public String saveScreeningReservation(@RequestParam("seatId")
        List<Long> seatIds, @RequestParam("screeningId") Long screeningId,
        Model model) {
11:     ScreeningDto screeningDto = screeningService
        .getScreening(screeningId);
12:     Double totalPrice = screeningService.getTotalPrice((long)
        seatIds.size(),screeningDto.getMovieDto().getPrice());
13:     model.addAttribute("totalPrice", totalPrice);
14:     model.addAttribute("screeningDto", screeningDto);
15:     model.addAttribute("seats", seatIds);
16:     return "ticket"; }
```

Sl. 4.12. Kontroler za entitet termin – prikaz sjedala

Linija ***Kod***

```
1:      @PostMapping("/ticket/save")
2:      public String saveTicket(@RequestParam("seatIds")
        List<Long> seatIds, @RequestParam("screeningId") Long screeningId){
3:      ScreeningDto screeningDto = screeningService
        .addReservation(screeningId, seatIds);
4:      Authentication authentication = SecurityContextHolder
        .getContext().getAuthentication();
5:      String username = authentication.getName();
6:      ticketService.addTickets(seatIds, screeningId, username);
7:      return "redirect:/"; }
```

Sl. 4.13. Kontroler za entitet ulaznica

Kontroler za entitet dvorana omogućuje dodavanje i spremanje nove dvorane.

Kontroler za entitet korisnik omogućuje prikaz zaslona za detalje o korisniku te povijesti prijašnje rezerviranih ulaznica. Kontroler za registraciju korisnika odvojen je u posebnu klasu te pruža prikaz forme za registraciju korisnika i pohranjivanje istog, kao što je prikazano u programskom kodu 4.14.

Linija ***Kod***

```
1:      @GetMapping("/register")
2:      public String showRegistrationForm(Model model){
3:      UserRequest user = new UserRequest();
4:      user.setStaff(false);
5:      model.addAttribute("user", user);
6:      return "register"; }

7:      @PostMapping("/register/user")
8:      public String createUser(@ModelAttribute("user") UserRequest user){
9:      userService.addUser(user);
10:     return "redirect:/?success"; }
```

Sl. 4.14. Kontroler za registraciju

4.4 Autorizacija

Registrirani korisnici mogu posjedovati obična korisnička prava i administratorska prava. Korisnik sa administratorskim pravima ima pristup svim dijelovima web aplikacije, dok obični korisnik ima ograničeni pristup web aplikaciji.

Obični korisnik ne smije imati sljedeće mogućnosti:

- dodavanje novog filma
- dodavanje nove dvorane
- dodavanje novog termina

Programski kod 4.16. prikazuje pravila za pristup određenom URL-u. Svaki URL- koji prima HTTP POST metodu za spremanje novih objekata je dostupan samo korisniku sa administratorskim pravima.

Poveznice koje vode na URL-ove namijenjene samo za vlasnika su također skrivene od običnih korisnika uz pomoć autorizacijskih pravila.

Linija ***Kod***

```
1:      @Bean
2:      public SecurityFilterChain securityFilterChain(HttpSecurity
        httpSecurity) throws Exception {
3:      return httpSecurity
4:      .csrf(AbstractHttpConfigurer::disable)
5:      .authorizeHttpRequests(registry -> {
6:      registry.requestMatchers("/", "/register/**").permitAll();
7:      registry.requestMatchers("/home", "/register/**").permitAll();
8:      registry.requestMatchers("/movie").hasRole("ADMIN");
9:      registry.requestMatchers("/movie/*").permitAll();
10:     registry.requestMatchers("/movie/save").hasRole("ADMIN");
11:     registry.requestMatchers("/screening/**").hasRole("USER");
12:     registry.requestMatchers("/screening/save").hasRole("ADMIN");
13:     registry.requestMatchers("movie/*/screening").hasRole("ADMIN");
        registry.requestMatchers("halls/**").hasRole("ADMIN");
        registry.requestMatchers("/register/**").permitAll();
        registry.requestMatchers("/ticket/**").permitAll();
        registry.requestMatchers("/profile").hasRole("USER");
        registry.anyRequest().permitAll(); })
        .formLogin(AbstractAuthenticationFilterConfigurer::permitAll)
        .build(); }
```

Sl. 4.16. Autorizacijska pravila za korisnike

Programski kod 4.15. prikazuje primjer HTML poveznice koja se prikazuje samo ako trenutno prijavljeni korisnik posjeduje administratorska prava.

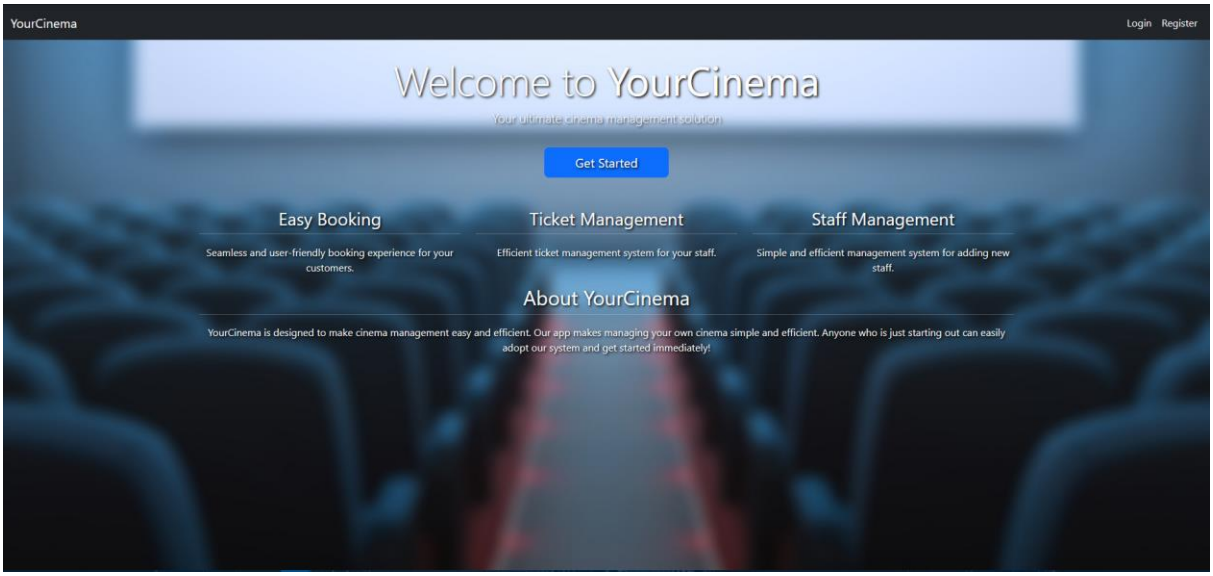
Linija ***Kod***

```
1:      <li class="nav-item">
2:      <a class="nav-link active" aria-current="page" th:href="@{/movie}"
        th:if="${#authorization.expression('hasRole(''ROLE_ADMIN'')')}">
        Add Movie</a>
3:      </li>
```

Sl. 4.15. Prikaz poveznice samo za korisnike sa administratoskim pravima

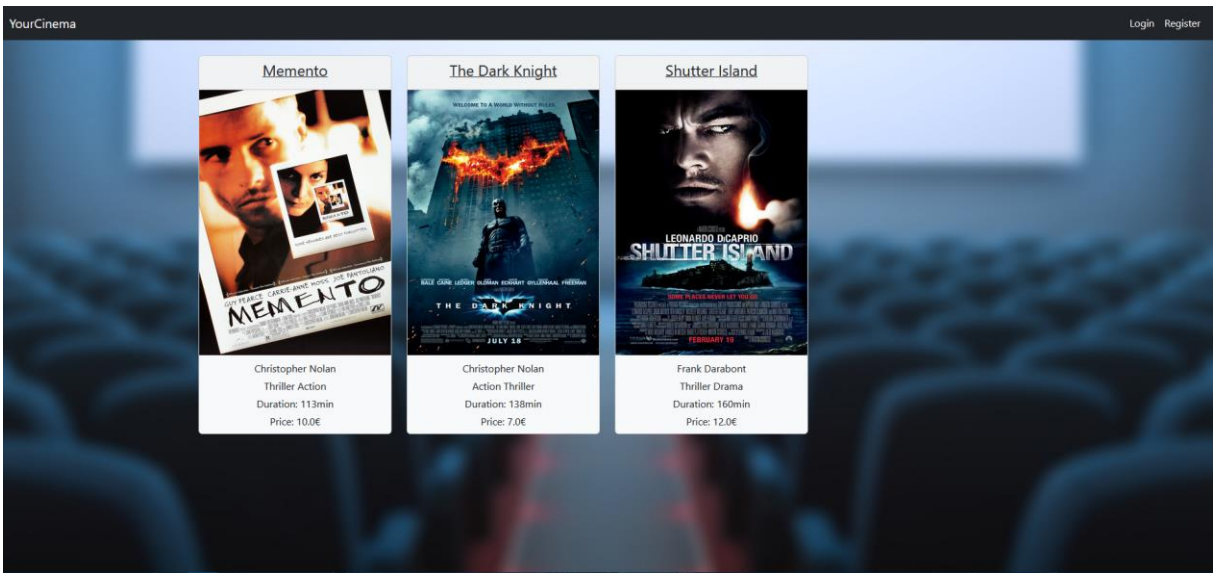
5. PRIKAZ FUNKCIONALNOSTI WEB APLIKACIJE

Ovo poglavlje prikazati će korisničko sučelje te način korištenja web aplikacije. Pokretanje web aplikacije dovodi korisnika na naslovnu stranicu koja ukratko opisuje funkcionalnosti aplikacije.



Sl. 5.1. Naslovna stranica web aplikacije

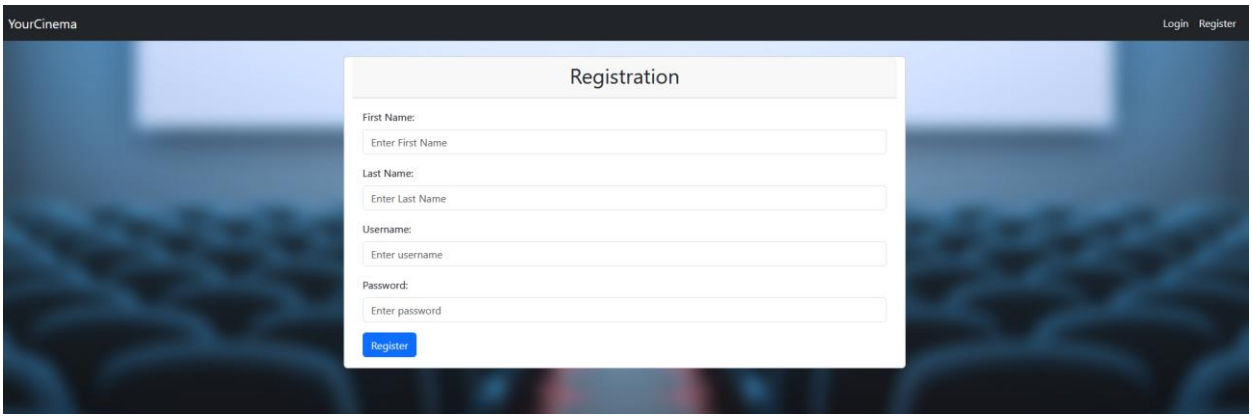
Na vrhu stranice nalazi se navigacijska traka (engl. navigation bar) koja omogućuje korisniku pristupanje ostalim zaslonima projektima ovisno o njihovim korisničkim pravima. Slika 5.1. prikazuje navigacijsku traku za korisnika koji nije prijavljen. Odabirom poveznice „Get Started“ korisnik je preusmjeren na zaslon koji prikazuje filmove koji su trenutno dostupni.



Sl. 5.2. Zaslon trenutno dostupnih filmova

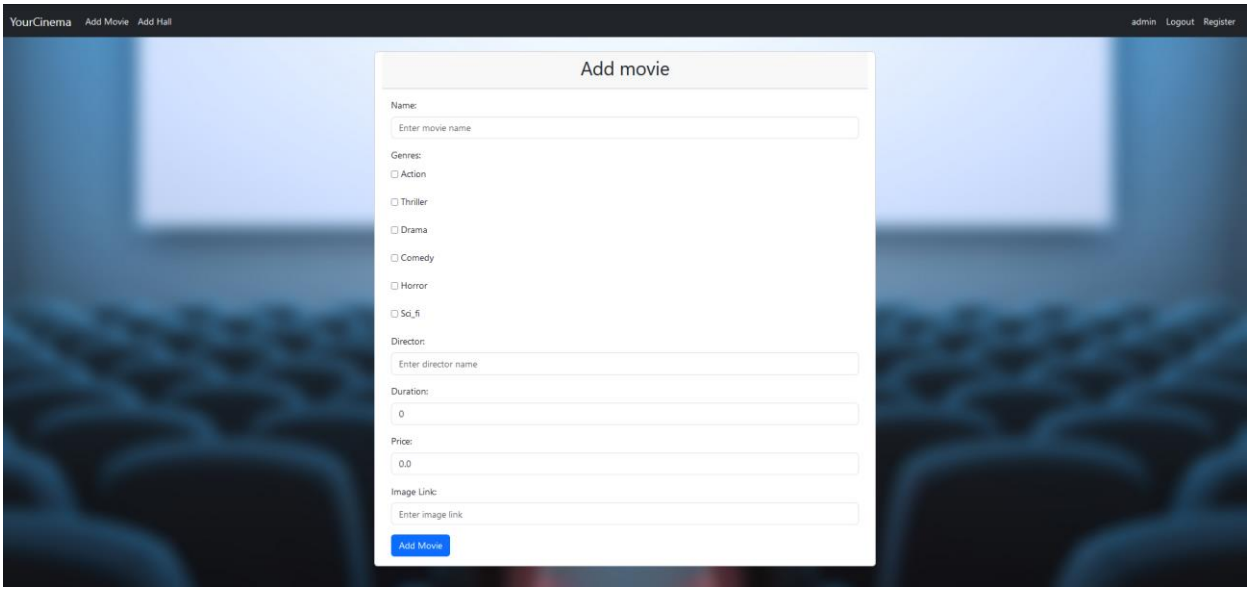
Korisnik koji trenutno nema vlastiti korisnički račun svoj može kreirati na „Register“ poveznici unutar navigacijske trake koja preusmjerava korisnika na zaslon za registraciju.

Slika 5.3. prikazuje formu koju korisnik može popuniti osobnim podacima kako bi stvorila korisnički račun. Nakon stvaranja korisničkog računa, korisnik se može prijaviti pomoću poveznice „Login“. Ukoliko korisnik ima administratorska prava, na navigacijskoj traci moći će pristupiti poveznicama za dodavanje novog filma za prikaz te dodavanja nove dvorane.



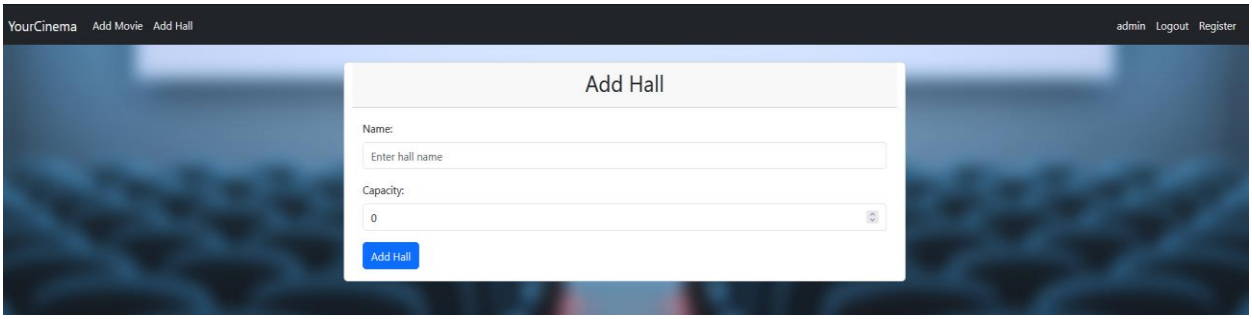
Sl. 5.3. Zaslon za registraciju korisnika

Slika 5.4. prikazuje formu, koja se nalazi na poveznici koju korisnik sa administratorskim pravima može popuniti kako bi dodao novi film za prikazivanje. Pritiskom na gumb „Add Movie“ novi film se dodaje u bazu podataka te će biti vidljiv na zaslonu prikazan na slici 5.2.



Sl. 5.4. Zaslon za dodavanje novog filma

Zaslon za dodavanje nove dvorane nalazi se na poveznici „Add Hall“ unutar navigacijske trake. Slika 5.5. prikazuje formu koja zahtijeva unos imena i kapaciteta dvorane koju korisnik sa administratorskim pravima može dodati.

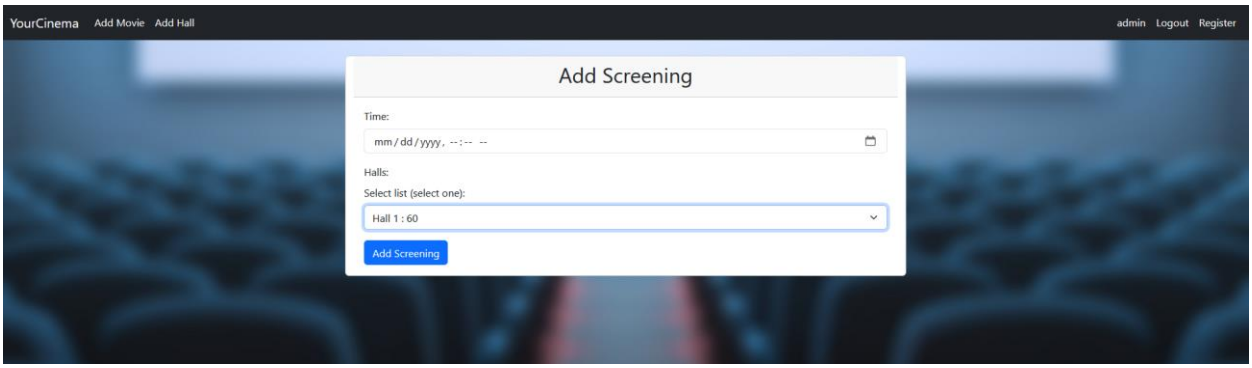


Sl. 5.5. Zaslon za dodavanje nove dvorane

Svaki film prikazan na slici 5.2. sadrži poveznicu koja vodi na zaslon koji prikazuje dostupne termine za rezervaciju sjedala. Zaslon na slici 5.6. sadrži tablicu sa svim terminima, poveznicu „Reserve“, koja se prikazuje svim prijavljenim korisnicima i poveznicu „Add Screening“, koja se prikazuje samo prijavljenim korisnicima sa administratorskim pravima.

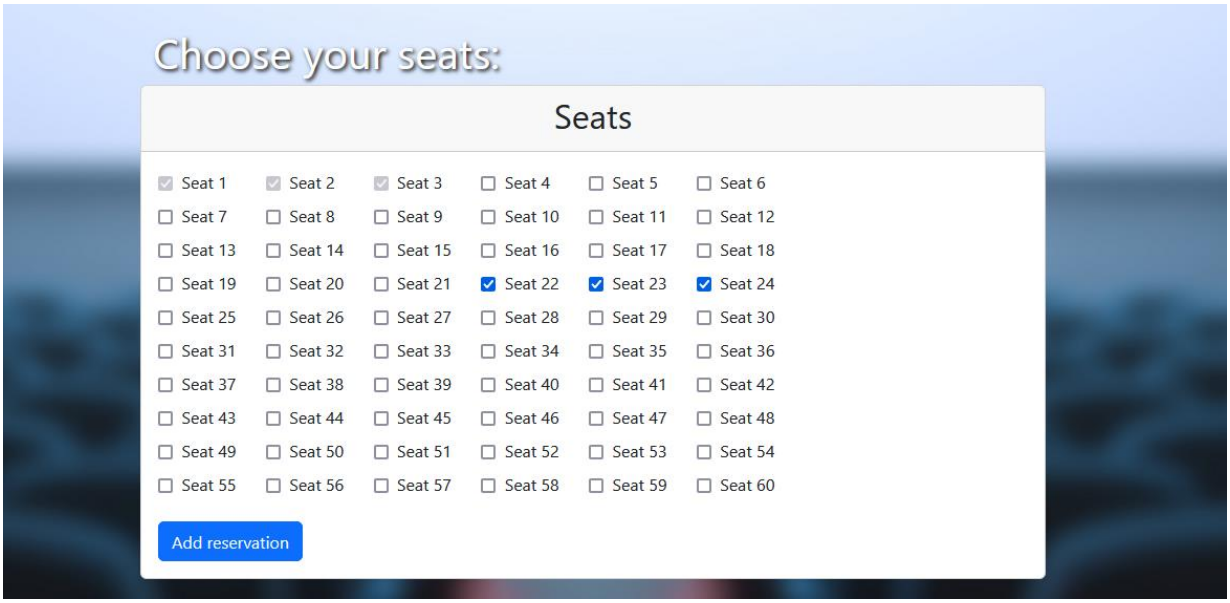
Sl. 5.6. Zaslon za prikaz, dodavanje te odabir termina

Poveznica „Add Screening“ sadrži zaslon za dodavanje novog termina za odabrani film, koji može dodati samo korisnik sa administratorskim pravima. Zaslon na slici 5.8. prikazuje formu u kojoj korisnik odabire datum, vrijeme i dvoranu u kojoj će se film prikazivati. Odabir dvorane moguć je pomoću padajućeg izbornika (engl. dropdown menu) koji sadrži sve dostupne dvorane. Poveznica „Add Screening“ sprema termin u bazu podataka.



Sl. 5.8. Zaslon za dodavanje termina

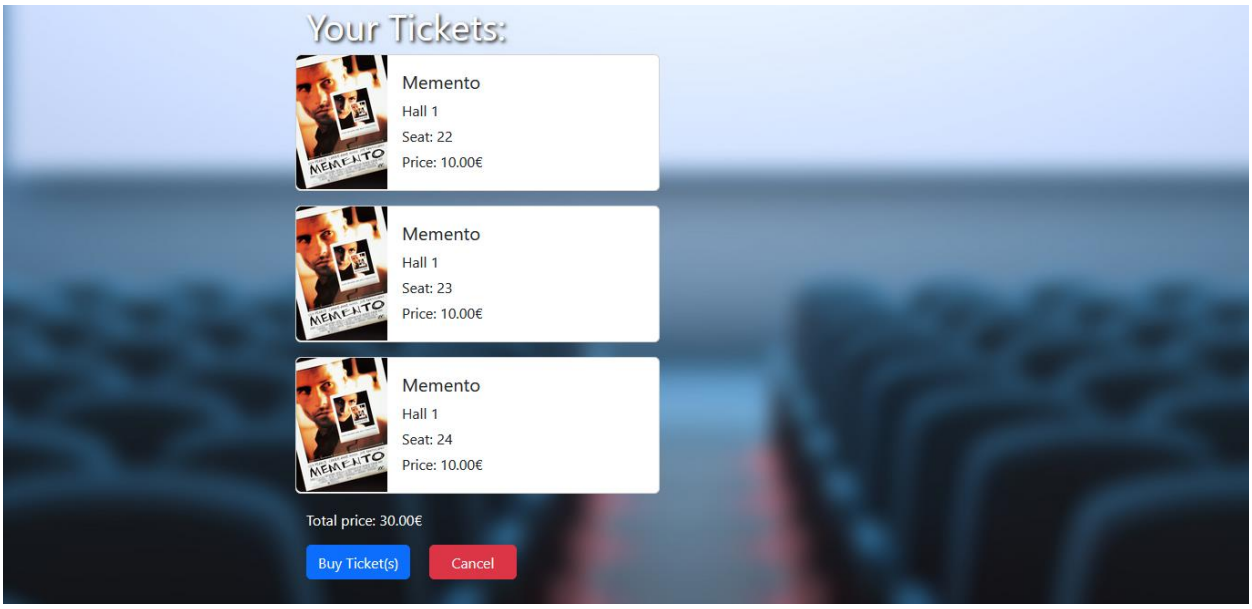
Poveznica „Reserve“ na zaslonu sa slike 5.6. sadrži zaslon za odabir sjedala prilikom rezervacije. Na zaslonu je moguće odabrati proizvoljan broj sjedala pritiskom na potvrdni okvir (engl. checkbox) za željeno mjesto. Sjedala koja su već rezervirana od strane drugih korisnika nisu moguća za odabir te su označena sa sivom bojom, a sjedala koja je korisnik trenutno odabrao su prikazana plavom bojom. Slika 5.7. prikazuje sjedala koja su dostupna za rezervaciju.



Sl. 5.7. Zaslon za rezervaciju sjedala

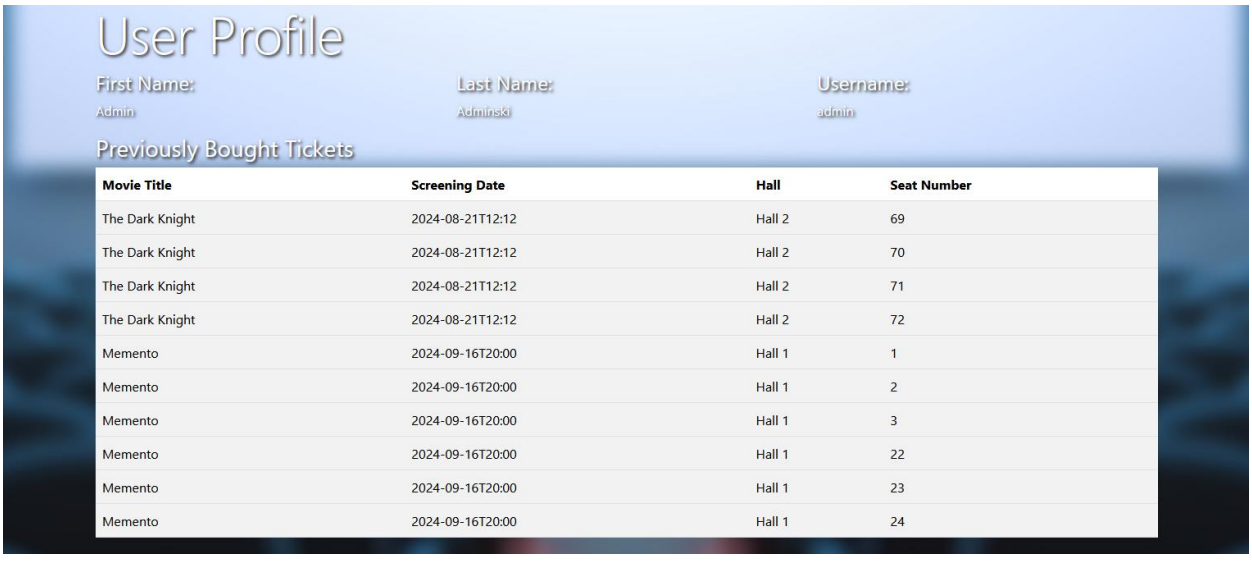
Poveznica „Add reservation“ sadrži zaslon prikazan na slici 5.9. koji prikazuje ulaznice koje će biti dodijeljene korisniku nakon potvrde kupovine. Svaka ulaznica sadrži ime filma, dvoranu, broj sjedala te cijenu. Zaslon također prikazuje ukupnu cijenu svih ulaznica, poveznicu za potvrdu kupovine i poveznicu za otkazivanje kupovine. Ukoliko korisnik potvrdi kupovinu ulaznica biti će

odveden na zaslon prikazan na slici 5.2., u suprotnom će biti odveden na zaslon prikazan na slici 5.7. te će morati ponovno odabrati željena sjedala.



Sl. 5.9. Zaslon za kupovinu ulaznica

Unutar navigacijske trake se također nalazi poveznica koja je imenovana prema korisniku koji je trenutno prijavljen. Odabirom te poveznice korisnik je odveden na svoj korisnički profil, prikazan na slici 5.10., koji prikazuje detalje o njihovom korisničkom računu te popis svih ulaznica koje su kupili. Ukoliko korisnik nije prijavljen poveznica neće biti prikazana.



Sl. 5.10. Zaslon za korisnički profil

6. ZAKLJUČAK

Cilj ovoga rada bio je stvoriti jednostavno i intuitivno rješenje za osobe i gradove koje žele osnovati vlastitu kinodvoranu. Inspiriran je od strane gradova koje imaju kinodvoranu, ali nemaju web stranicu pomoću koje je moguće unaprijed rezervirati ulaznicu. Opisano rješenje bi pružilo jednostavno rješenje za manje gradove koji nemaju mogućnost stvaranja vlastitog softvera za kinodvoranu.

U drugom poglavlju, gdje su opisana slična rješenja za isti problem, mogu se uočiti buduće implementacije koje u ovoj web aplikaciji nedostaju, kao što su prikaz statističkih podataka o prometu po kojima bi vlasnik mogao doći do daljnjih zaključaka o prodaji ulaznica. Također postoji mogućnost implementiranja kupovine hrane i pića unutar kinodvorane.

Izrađena web aplikacija nudi najosnovnije mogućnosti za vođenje vlastite kinodvorane te kao takva može dalje napredovati te bi uz dodatne naknade mogla pružati specifične mogućnosti prema izboru klijenta.

LITERATURA

[1] N. Tartaglione, »Global Box Office Reaches $33.9B In 2023, Up 31% On 2022 – Analysts,« Deadline, 4 1 2024. [Mrežno]. Available: https://deadline.com/2024/01/global-box-office-2023-total-barbie-super-mario-bros-oppenheimer-international-china-1235694955/. [Pokušaj pristupa 21 8 2024].

[2] CineSYNC, »FAQ on CINEsync SaaS Cinema and Movie Theater Software Solution,« CineSYNC, [Mrežno]. Available: https://www.cinesync.io/faq. [Pokušaj pristupa 21 8 2024].

[3] CineStar, »O nama,« CineStar, [Mrežno]. Available: https://osijek.cinestarcinemas.hr/o-nama/. [Pokušaj pristupa 21 8 2024].

[4] Odeon, »About Us,« Odeon, [Mrežno]. Available: https://www.odeon.co.uk/about-us/. [Pokušaj pristupa 21 8 2024].

[5] Vista, »About us,« Vista, [Mrežno]. Available: https://www.vista.co/about-us. [Pokušaj pristupa 21 8 2024].

[6] IBM, »What is Java Spring Boot?,« IBM, [Mrežno]. Available: https://www.ibm.com/topics/java-spring-boot. [Pokušaj pristupa 22 8 2024].

[7] Jetbrains, »Features Overview,« Jetbrains, [Mrežno]. Available: https://www.jetbrains.com/idea/features/. [Pokušaj pristupa 22 8 2024].

[8] Thymeleaf, »Thymeleaf,« Thymeleaf, [Mrežno]. Available: https://www.thymeleaf.org/. [Pokušaj pristupa 22 8 2024].

[9] J. Erickson, »MySQL: Understanding What It Is and How It's Used,« Oracle, 29 8 2024. [Mrežno]. Available: https://www.oracle.com/mysql/what-is-mysql/. [Pokušaj pristupa 3 9 2024].

SAŽETAK

U radu je razvijena web aplikacija za organizaciju rada malih kinodvorana. U teorijskom dijelu opisane su tehnologije koje su korištene prilikom izrade web aplikacije. Navede tehnologije su Spring Boot ovkir, IntelliJ razvojno okruženje, SQLite za operacije nad bazom podataka i Thymeleaf za stvaranje dinamičnih HTML stranica. Nadalje je opisan proces izrade web aplikacije, gdje su predstavljene funkcije koje bi kinodvorana trebala imati te način na koje su te funkcije ispunjene. Glavni dijelovi web aplikacije su objašnjeni prema entitetima koji ju čine, te njihovi servisi i kontroleri koji se brinu za logiku iza navedenih entiteta. Također je prikazano korisničko sučelje web aplikacije, način na koji vlasnik ili običan korisnik mogu koristiti web aplikaciju te prava koja stječu administrator i običan korisnik.

Ključne riječi: entitet, kinodvorana, kontroler, servis, web aplikacija

ABSTRACT

Web application for reservation of seats in a cinema

This paper presents a web application developed for organizing the operations of small cinemas. In the theoretical part, the technologies used in the development of the web application are described. The mentioned technologies include the Spring Boot framework, the IntelliJ development environment, SQLite for database operations, and Thymeleaf for creating dynamic HTML pages. Furthermore, the process of creating the web application is described, presenting the features that the cinema should have and how those features are implemented. The main parts of the web application are explained based on the entities that make up the web application, along with their services and controllers that handle the logic behind the mentioned entities. The user interface of the web application is also shown, including how the owner or a regular user can use the web application, as well as the rights granted to the administrator and the regular user.

Keywords: cinema, controller, entity, service, web application

ŽIVOTOPIS

Karlo Štefanac rođen je u Bjelovaru 25. siječnja 2003. godine. Osnovno obrazovanje započinje 2009. godine u Osnovnoj Školi Garešnica. Nakon stjecanja osnovnog obrazovanja 2017. godine započinje školovanje u Tehničkoj školi Kutini, smjer tehničar za računalstvo. Srednju školu završava 2021. godine te iste upisuje preddiplomski studij Računarstva, smjer programsko Inženjerstvo na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek.