

REZERVACIJA KARATA ZA BIOSKOP

Predmet:Klijent server sistemi

Profesor:

**Dr Mirko Kosanović
Miloš Kosanović**

Student:

**Stanković Katarina
REr 20/17**

SADRŽAJ:

1. Uvod.....	-3-
2. Instalacija i podešavanje projekta.....	-3-
2.1 Instaliranje modula.....	-3-
3. Arhitektura aplikacije.....	-4-
3.1 Serverski deo.....	-4-
3.2 Klijentski deo.....	-5-
3.3 Baza podataka.....	-5-
3.4 Komunikacija.....	-5-
4. Kreiranje i povezivanje baze.....	-5-
5. Rad aplikacije.....	-7-
5.1 Opis implementacije.....	-7-
5.2 Opis funkcionalnosti-korisničko uputstvo.....	-11-
6. Literatura.....	-14-

1. Uvod

U ovom projektu obrađena je izrada Web aplikacije po nazivom “Rezervacija karata za bioskop”, koja će služiti za rezervisanje karata za određeni film. Alati koji su korišćeni prilikom izrade i pokretanje aplikaciju su web pretraživači, Command Prompt, Sublime Text i XAMPP zbog povezivanja sa bazom. Tehnologije koje su korišćene na klijentskoj strani su HTML, CSS i JavaScript dok su na serverskoj strani korišćeni Node.js, Express, MySQL i EJS(Embed JavaScript Templates). Aplikacija se sastoji iz 3 dela. Na prvoj strani postoji mogućnost izbora filma koji korisnik želi da pogleda, bira se jedan od četiri ponuđena filma. Nakon odabira filma otvara se druga strana gde su predstavljena sedišta u sali kako bi korisnik odabrao željeno mesto, kao i cena karte i mala forma gde korisnik unosi svoje ime i potvrđuje svoju rezervaciju klikom na dugme “Rezerviši”. Karte nije moguće rezervisati ukoliko korisnik ne unese svoje ime. Takođe, na drugoj strani postoji i dugme “Filmovi”, klikom na ovo dugme korisnik se vraća na prvu stranu i može da izabere drugi film i postoji dugme “Reset seats” koje briše sve rezervacije. I treća strana aplikacije ispisuje osnovne informacije o rezervaciji, kao što su: ime korisnika, naziv filma, broj sale, broj rezervisanih sedišta, ukupna cena karata... I sadrži dugme “Nazad” koje korisnika vraća na prethodnu stranu. Tada se i ime korisnika koji je rezervisao karte čuva u bazi i menja se status za sedišta koja su rezervisana.

2. Instalacija i podešavanje projekta

Da bismo pokrenuli ovu Node.js aplikaciju neophodno je da instaliramo Node.js koji možemo preuzeti sa sledeće stranice: <https://nodejs.org/en/>. Baza koju koristimo u ovom projektu je MySQL i ukoliko želimo da vidimo promene koje se dešavaju usled rezervacije karata, bazi možemo pristupiti preko phpMyAdmin.

2.1 Instaliranje modula

Projekat sadrži **package.json** fajl koji kreiramo komandom **npm init**. A nakon njegove instalacije neophodno je instalirati module koji su potrebni za naš projekat naredbom **npm install**. Ova naredba pretražuje

package.json fajl i u njemu traži i instalira sve module koji su potrebni za ovaj projekat.

Tokom razvoja aplikacije potrebno je prilikom svake promene restartovati aplikaciju kako bi se promene videle. To rešavamo instaliranjem **nodemon** skripte koja će pratiti sve promene unutar projekta u kojem je pokrenuta i automatski restartovati aplikaciju kako bi se promene videle u npr. web pretraživaču. Nodemon se može instalirati na dva načina: lokalno unutar projekta i globalno na računaru. Naredbom **npm install nodemon --save** instaliramo nodemon lokalno unutar projekta, dok naredbom **npm install -g nodemon** instaliramo globalno na računaru.

3. Arhitektura aplikacije

Aplikacija sadrži korenski (eng. Root) direktorijum “/” koji u sebi sadrži foldere: index, movies, reservations i users u kojima se nalaze kodovi rute, folder “views” koji sadrži statičke datoteke koje opisuju izgled aplikacije. Takođe, aplikacija sadrži fajl “db” koji predstavlja konekciju sa bazom podataka i folder “node modules” koji sadrži sve pakete koji su navedeni u “package.json”. Dve datoteke “/package.json” i “/package-lock.json” koji predstavlja JSON dokument koji opisuje samu aplikaciju i sadrži spisak modula od kojih je serverski deo aplikacije zavistan.

3.1 Serverski deo

Node.js je programski jezik zasnovan na JavaScript jeziku. On je ne-blokirajući, event driven, lightweight, efikasan jezik čija je glavna namena da se koristi kod distribuiranih aplikacija koje rade na različitim platformama i koje imaju potrebu da rade sa velikim količinama zahteva ili podataka u realnom vremenu. NodeJS je naročito pogodan za aplikacije koje moraju da održavaju perzistentnu konekciju sa serverom, najčešće korišćenjem veb soketa (primer takve aplikacije bi bio chat program). Mrežene aplikacije koje zahtevaju brzinu, skalabilnost i podržavaju veliki broj istovremenih konekcija se razvijaju u ovom programskom jeziku.

Više o NodeJS-u možete pogledati ovde: <https://nodejs.org/en/>

Express modul je najpopularniji i najrasprostranjeniji modul koji se koristi u Node.js aplikacijama. Omogućava ubrzanje razvoja i koristi se

slično kao http i https moduli, samo što ima dodatne mogućnosti. Kako E-xpress nije glavni modul on se mora instalirati lokalno uz pomoć komande npm-a (Node package manager).

EJS je jednostavan šablonski jezik koji nam omogućava da generišemo HTML oznake pomoću običnog JavaScripta.

3.2 Klijentski deo

HTML(Hyper Text Markup Language, jezik za označavanje hipertekstova) je opisni jezik specijalno namenjen opisu Web stranica. Pomoću njega se jednostavno mogu odvojiti elementi kao što su naslovi, paragrafi i slično.

CSS (engl. Cascading Style Sheets) je jezik formatiranja pomoću kog se definiše izgled elemenata veb-stranice. Prvobitno, HTML je služio da definiše kompletan izgled, strukturu i sadržaj veb-stranice, ali je od verzije 4.0 HTML-a uveden CSS koji bi definisao konkretan izgled, dok je HTML ostao u funkciji definisanja strukture i sadržaja.

3.3 Baza podataka

Baza podataka predstavlja kolekciju podataka organizovanih tako da se podacima može lako pristupiti i manipulirati.

MySQL je sistem baza podataka koji se koristi na web-u. Pokreće se na serveru i idealan je za velike i male aplikacije.

Podaci u MySQL bazi podataka se čuvaju u tabelama.

3.4 Komunikacija

Rute putem kojih na osnovu zahteva klijenta server vraća zahtevane stranice koje su generisane:

<http://localhost:3000> -vraća početnu stranicu sajta gde korisnik bira film za koji želi da rezerviše kartu

<http://localhost:3000/reservation/idfilma> -vraća stranicu za rezervaciju karata za film koji je izabrao korisnik.

4. Kreiranje i povezivanje baze

Fajl db.js sadrži kod za povezivanje sa MySQL bazom podataka:

```

db.js
1  "user strict";
2
3  var mysql = require("mysql");
4
5  //local mysql db connection
6  var connection = mysql.createConnection({
7    host: "localhost",
8    user: "root",
9    password: "",
10   database: "rezervacije"
11 });
12
13 connection.connect(function(err) {
14   if (err) throw err;
15 });
16
17 module.exports = connection;
18

```

Dok sama baza izgleda ovako:

Server: 127.0.0.1 » Database: rezervacije

Structure SQL Search Query Export Import Operations Privileges Routines Events

Filters

Containing the word:

Table	Action	Rows	Type	Collation	Size	Overhead
halls		4	InnoDB	latin1_swedish_ci	16 KiB	-
movies		4	InnoDB	latin1_swedish_ci	32 KiB	-
reservations		6	InnoDB	latin1_swedish_ci	64 KiB	-
seats		156	InnoDB	latin1_swedish_ci	32 KiB	-
users		12	InnoDB	latin1_swedish_ci	16 KiB	-
5 tables	Sum	182	InnoDB	latin1_swedish_ci	160 KiB	0 B

☐ Check all With selected:

☐ Show all | Number of rows: 25 | Filter rows: | Sort by key:

+ Options

	id	name	year	image	hall_id
	7	Joker	2019	joker_2019.jpg	1
	8	Ones Upon a time hollywood	2019	ones_upon_a_time_hollywood_2019.jpg	2
	9	The Avengers	2019	the_avengers_2019.jpg	3
	10	The Irishman	2019	the_irishman_2019.jpg	4

☐ Check all With selected:

5. Rad aplikacije

5.1 Opis implementacije

U folderu routes se nalaze fajlovi **movies.js** i **reservations.js**.

U fajlu **movies.js** najpre pokrećemo Node.js express biblioteku, zatim vršimo rutiranje i povezivanje sa bazom podataka. Nakon toga izvlačimo listu svih filmova, tačnije, šaljemo upit u bazu za izvlačenje svih filmova iz baze koji se nalaze u tabeli "movies". Nakon uspešnog odgovora i dobavljanja liste sa filmovima koji se nalaze u bazi, poziva se metoda **res.render()** za prikazivanje template-a stranice. Prosleđujemo koji template želimo da se prikaže, u našem slučaju to je movies.ejs sa podacima smeštenim u objektu **result**.

```
movies.js
1  var express = require("express");
2  var router = express.Router();
3  var sql = require("../db.js");
4
5
6  router.get("/", function(req, res, next) {
7    sql.query("SELECT * FROM movies", function(err, result) {
8      if (err) {
9        console.log("error", err);
10     } else {
11       res.render("movies", { result: result });
12     }
13   });
14 });
15
16 module.exports = router;
17
```

U fajlu **reservations.js** najpre pokazujemo sva sedišta dostupna za ovaj film:

```
reservations.js
router.get("/:id", function(req, res, next) {
  sql.query(
    "SELECT s.id, s.seat_number, s.status, s.hall_id FROM movies AS m JOIN seats AS s ON m.hall_id = s.hall_id AND m.id = ?",
    req.params.id,
    function(err, result) {
      if (err) {
        console.log("error", err);
      } else {
        res.render("reservation", { data: result, resetUrl: req.originalUrl });
      }
    }
  );
});
```

Nakon toga na putanji localhost:3000/reservations/:id, gde id predstavlja id filma, vršimo rezervaciju sedišta, a za to koristimo post metodu za slanje tog zahteva:

```
reservations.js
router.post("/:id", function(req, res, next) {
  let seatArray = [];
  if (req.body["seat[]"]) {
    Array.isArray(req.body["seat[]"])
      ? (seatArray = req.body["seat[]"])
      : seatArray.push(req.body["seat[]"]);
  }
  if (req.body.name !== "" && seatArray.length !== 0) {
    let reservationId = null;
    let userId = null;
    let seatsNumbers = [];
```

Zatim sledi ubacivanje podataka o korisniku koji je napravio rezervaciju u bazu podataka i ubacivanje u tabelu "reservations":

```
reservations.js
sql.query(
  "INSERT INTO users(user_name) VALUES (?)",
  req.body.name,
  function(err, result) {
    if (err) {
    } else {
      userId = result.insertId;
      for (var key in seatArray) {
        sql.query(
          "INSERT INTO reservations(movie_id, user_id, seat_id, price) VALUES (?, ?, ?, ?)",
          [req.params.id, userId, seatArray[key], req.body.price],
          function(err, result) {
            if (err) {
              console.log("error: ", err);
            } else {
            }
          }
        );
      }
    }
  }
);
```

Zatim sledi ažuriranje statusa za sedišta u bazi podataka:

```
reservations.js
sql.query(
  "UPDATE seats SET status = 1 WHERE id = ?",
  seatArray[key],
  function(err, result) {
    if (err) {
      console.log("error", err);
      return;
    } else {
    }
  }
);

sql.query(
  "SELECT * FROM seats WHERE id = ?",
  seatArray[key],
  function(err, result) {
    if (err) {
      console.log("error: ", err);
    } else {
      seatsNumbers.push(result[0].seat_number);
    }
  }
);
```


Nakon uspešne rezervacije izvlačimo podatke iz baze za tu rezervaciju i prikazujemo u template-u **success**. Neophodne podatke prosleđujemo kao objekat **data**, dok nam objekat **backUrl** služi za povratak na prethodnu stranicu sa koje smo došli. I na kraju imamo prikaz template-a **error.ejs** ukoliko podaci nisu ispravno uneti:

```
reservations.js

sql.query(
  "SELECT * FROM reservations JOIN movies ON reservations.movie_id = movies.id JOIN users ON reservations.user_id = users.id",
  function(err, result1) {
    if (err) {
      console.log(err);
    } else {
      sql.query(
        "SELECT * FROM halls WHERE id = ?",
        result1[0].hall_id,
        function(err, result2) {
          if (err) {
            console.log(err);
          } else {
            let numbers = { seats: seatsNumbers };
            let hallName = { hall_name: result2[0].name };
            let obj = Object.assign(result1[0], numbers, hallName);
            res.render("success", {
              backUrl: req.originalUrl,
              data: obj
            });
          }
        }
      );
    }
  }
);
} else {
  res.render("error", {
    message: "Unesite ime i selektujte najmanje jedno sediste",
    backUrl: req.originalUrl
  });
}
```

I na kraju fajla **reservations.js** imamo resetovanje podataka za sedišta, brisanje rezervacije iz tabele na osnovu filma i ažuriranje statusa sedišta tj vraćanje na slobodno stanje:

```
reservations.js

router.get("/:id/reset/:hall_id", function(req, res, next) {

  sql.query(
    "DELETE FROM reservations WHERE movie_id = ?",
    req.params.id,
    function(err, result) {
      if (err) {
        console.log("error: ", err);
      } else {
        sql.query(
          "UPDATE seats SET status = 0 WHERE hall_id = ?",
          req.params.hall_id,
          function(err, result) {
            if (err) {
              console.log("error", err);
            } else {
              res.redirect("/reservation/" + req.params.id);
            }
          }
        );
      }
    }
  );
});

module.exports = router;
```

U folderu **views** nalaze se fajlovi: **error.ejs**, **footer.ejs**, **header.ejs**, **movies.ejs**, **reservation.ejs** i **success.ejs**. Ovo su statički fajlovi koji služe za dizajn aplikacije.

Error.ejs se prikazuje ukoliko korisnik nije uneo ime ili odabrao sedište, a **movies.ejs** je stranica koja prikazuje filmove za koje mogu da se rezervišu karte. **Reservation.ejs** predstavlja stranicu gde rezervišemo karte za film i **success.ejs** je stranica koja nam pokazuje da je rezervacija uspešno izvršena.

Movies.ejs:

```
movies.ejs
<!-- Header -->
<%- include('header') %>
<!-- End Header -->

<div class="content">
  <div class="container">
    <div class="header-title">
      <h2>Filmovi</h2>
    </div>
    <div class="movies-wrapper">
      <% for(var i=0; i< result.length;i++) { %>

        <div class="movies-item">
          <a href="/reservation/<%= result[i].id %>">
            <div class="cover">
              <div class="cover-overlay"></div>
              <div class="colored-overlay"></div>
              
            </div>
            <div class="movie-details">
              <div class="movie-name"><%= result[i].name %></div>
              <div class="movie-year"><%= result[i].year %></div>
            </div>
          </a>
        </div>

      <% } %>
    </div>
  </div>
</div>
```

Error.ejs:

```
error.ejs
<!-- Header -->
<%- include('header') %>
<!-- End Header -->

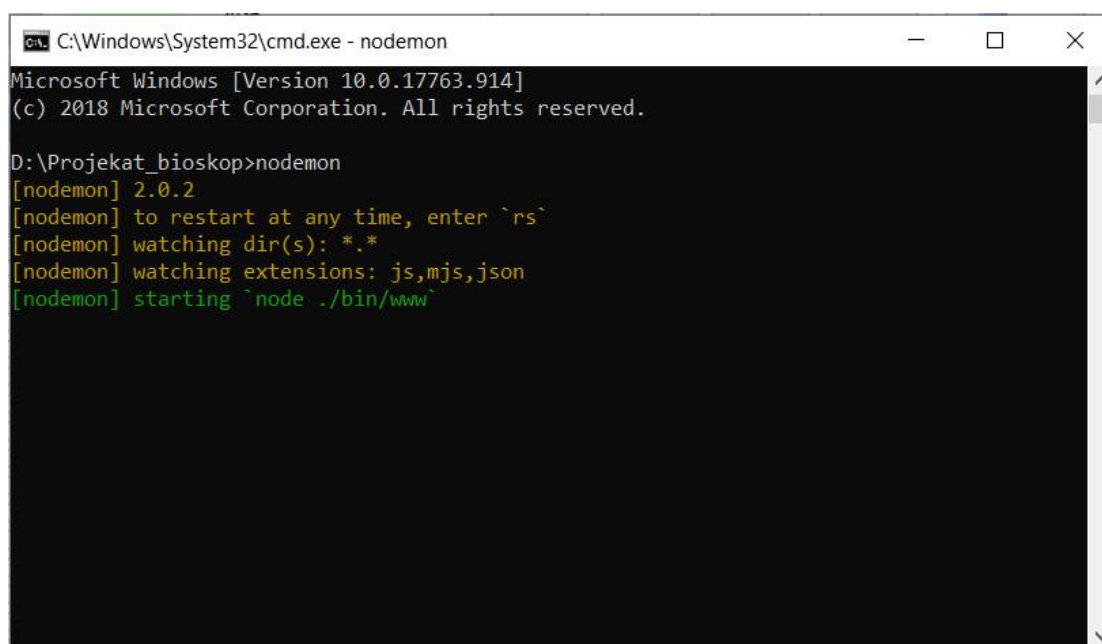
<div class="message">
  <h3><%= message %></h3>
  <a href="<%= backUrl %>"><button>Nazad</button></a>
</div>

<!-- footer -->
<%- include('footer') %>
<!-- End Footer -->
```

5.2 Opis funkcionalnosti-korisničko uputstvo

Za pokretanje aplikacije neophodan nam je Node.js sa odgovarajućim modulima. Nakon što instaliramo Node.js neophodno je da instaliramo module. Moduli se instaliraju naredbom `npm install` ili `npm install` ime modula. U ovom slučaju korišćena je naredba: **npm install**.

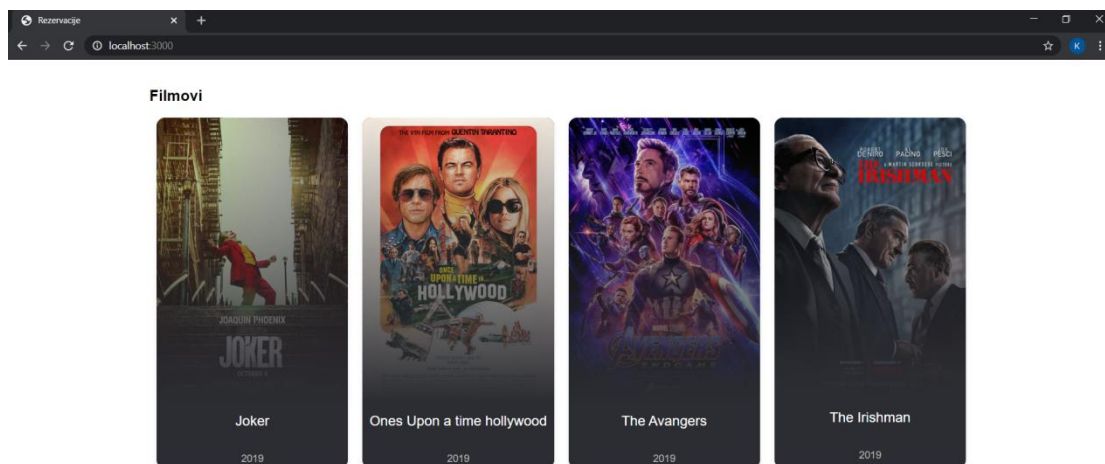
Nakon instalacije modula, instaliramo nodemon uz pomoć naredbe: **npm install -g nodemon**. Pre nego što pokrenemo samu aplikaciju neophodno je da uključimo XAMPP(moguće je skinuti sa sledećeg linka:<https://www.apachefriends.org/download.html>) koji nam kreira MySQL bazu podataka. Zatim aplikaciju pokrećemo preko komandne linije, komandom **nodemon**.



```
C:\Windows\System32\cmd.exe - nodemon
Microsoft Windows [Version 10.0.17763.914]
(c) 2018 Microsoft Corporation. All rights reserved.

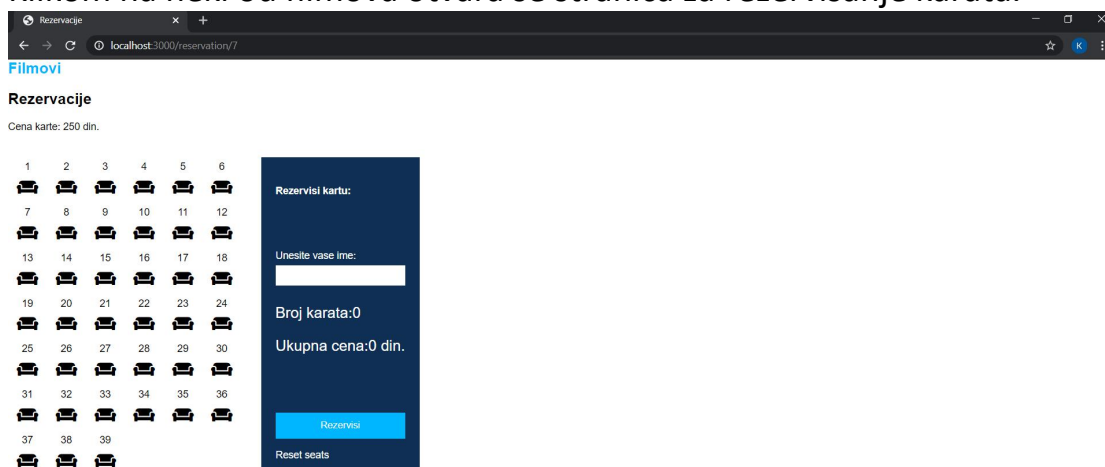
D:\Projekat_bioskop>nodemon
[nodemon] 2.0.2
[nodemon] to restart at any time, enter `rs`
[nodemon] watching dir(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node ./bin/www`
```

Nakon pokretanja, aplikacija se nalazi na lokaciji: <http://localhost:3000/>. U web pretraživaču se otvara početna stranica gde se nalaze filmovi za koje je moguće rezervisati karte.



Copyright 2020

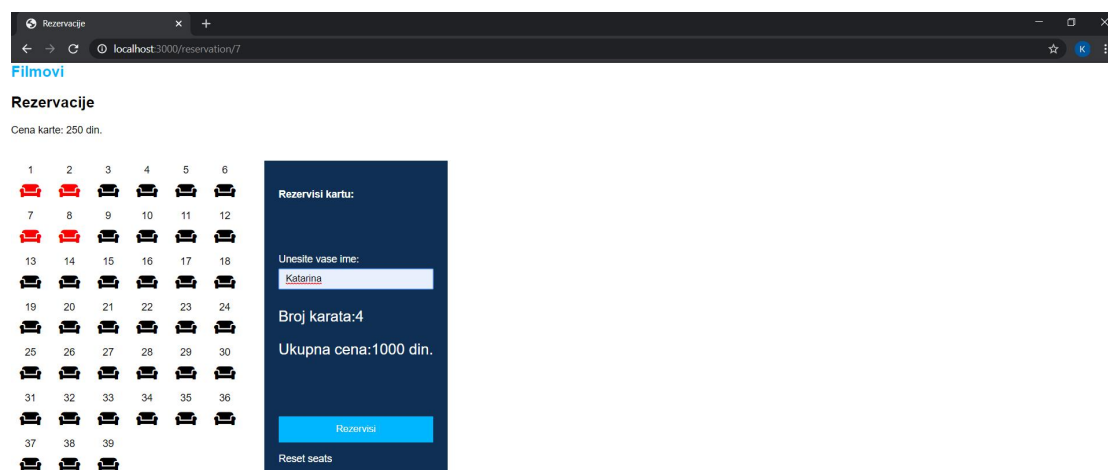
Klikom na neki od filmova otvara se stranica za rezervisanje karata.



Copyright 2020

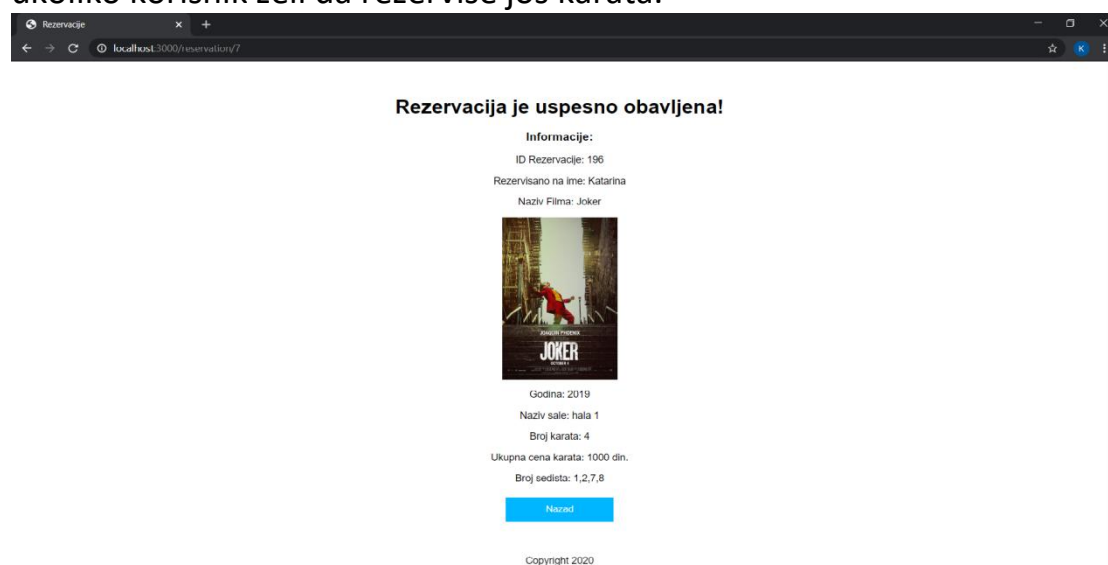
Ovde korisnik ima mogućnost da obeleži broj sedišta koji mu je potreban i unese svoje ime kako bi potvrdio rezervaciju. Karte se rezervišu klikom na dugme **Rezerviši**, a takođe korisnik ima mogućnost da poništi sedišta tj da ih resetuje klikom na dugme **Reset seats**.

Takođe na ovoj strani postoji dugme **Filmovi** čiji klik vraća na prethodnu stranicu u slučaju da korisnik želi da izabere drugi film.

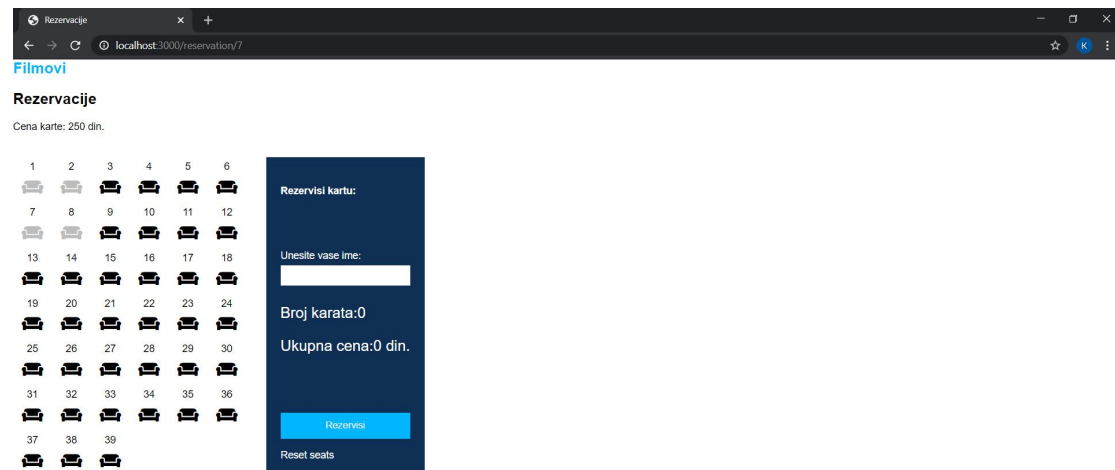


Copyright 2020

Nakon što je korisnik izabrao sedišta i uneo svoje ime, klikom na dugme **Rezerviši** prikazuje se stranica na kojoj se nalaze informacije o filmu i korisniku, gde postoji i dugme **nazad** koji vraća na prethodnu stranicu ukoliko korisnik želi da rezerviše još karata.



Nakon rezervacije sva sedišta koja su popunjena su obeležena drugom bojom i to izgleda ovako:



Copyright 2020

6. Literatura

<https://sr.wikipedia.org/wiki/>
<https://github.com/>
<https://www.youtube.com/>
<https://www.phpmyadmin.net/>
<https://nodejs.org/docs/latest-v11.x/api/>

