Web aplikacija za rezervaciju karata na događajima

Horvat, Andrej

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

Sveučilišni studij

WEB APLIKACIJA ZA REZERVACIJU KARATA NA DOGAĐAJIMA

Završni rad

Andrej Horvat

Osijek, 2023.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 07.09.2023.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime Pristupnika:	Andrej Horvat
Studij, smjer:	Računalno inženjerstvo
Mat. br. Pristupnika, godina upisa:	R 4354, 12.10.2021.
OIB Pristupnika:	74000666192
Mentor:	prof. dr. sc. Krešimir Nenadić
Sumentor:	,
Sumentor iz tvrtke:	
Naslov završnog rada:	Web aplikacija za rezervaciju karata na događajima
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak završnog rad:	Kratko opisati problematiku rezervacije karata za razne vrste događaja. Predložiti funkcionalnosti web aplikacije koja se koristi za rezervaciju karata na raznim događajima i dizajn baze podataka za web aplikaciju. Predvidjeti najmanje dva različita profila korisika: moderator događaja koji zadaje kategorije karate i broj karata po kategoriji, potrebne datume i ostale relevantne podatka; kupac koji se treba registrirati a nakon toga i rezervirati karte za neki objavljeni događaj. Potrebno je opisati postupak izrade web aplikacije. Tema rezervirana: Andrej Horvat
Prijedlog ocjene završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 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:	07.09.2023.
Datum potvrde ocjene od strane Odbora:	

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, 20.09.2023.

Ime i prezime studenta:	Andrej Horvat
Studij:	Računalno inženjerstvo
Mat. br. studenta, godina upisa:	R 4354, 12.10.2021.
Turnitin podudaranje [%]:	4

Ovom izjavom izjavljujem da je rad pod nazivom**: Web aplikacija za rezervaciju karata na događajima**

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

Tema ovog završnog rada je izrada web aplikacije za rezervaciju karata na događajima. Web aplikacija ima dva tipa korisnika, kupac i moderator. Aplikacija kupcu omogućuje pregled svih događaja, odabir željenog događaja te odabir tipa i broja karata. Moderator uz mogućnosti koje ima kupac može dodati novi događaj, napraviti izmjene u postojećem događaju, obrisati događaj i dodijeliti titulu moderatora drugom korisniku. Također, svakom korisniku je omogućena izrada računa unošenjem adrese elektroničke pošte i zaporke.

Motiv izrade ovog projekta je prenstveno dodatno upoznavanje s mogućnostima Firebasea i vježbanje izgradnje web aplikacije pomoću Firebasea, odnosno korištenje nekih od dostupnih alata i funkcija za pohranu podataka i autentikaciju. Također, ovaj projekt je bio konkretna podloga za testiranje do sada stečenog znanja te dodatnu vježbu tehnologija koje su korištene pri izradi ove web aplikacije, posebice JavaScripta, HTML-a (engl. HyperText Markup Language), odnosno hipertekstulanog označnog, jezika i Materialize CSS-a.

U drugom poglavlju opisane su već postojeće stranice za rezerviranje ili kupnju karata putem interneta. U trećem poglavlju objašnjene su tehnologije koje su korištene prilikom izrade web aplikacije, a to su JavaScript, HTML, CSS (engl. Cascading Style Sheets), odnosno kaskadni stilski listovi, Materialize CSS i Firebase. Četvrto poglavlje obuhvaća izradu i izgled web aplikacije.

1.1. Zadatak završnog rada

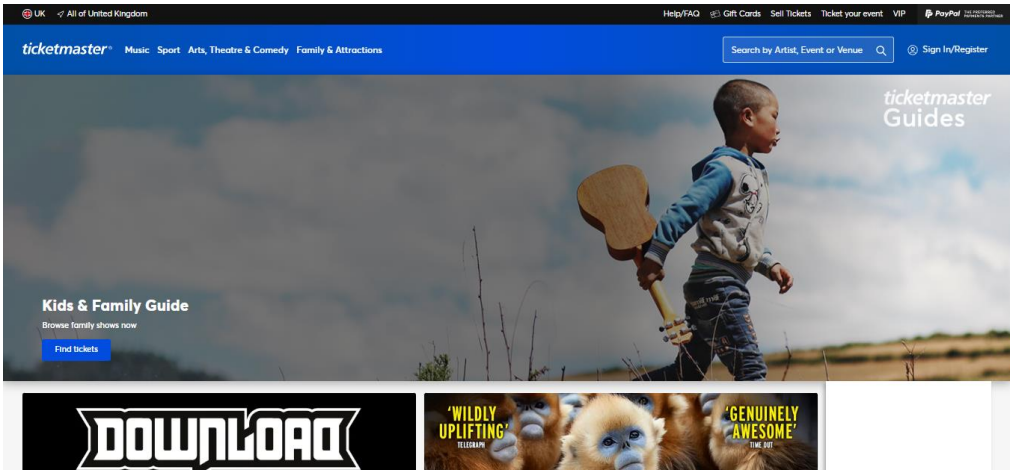
Kratko opisati problematiku rezervacije karata za razne vrste događaja. Predložiti funkcionalnosti web aplikacije koja se koristi za rezervaciju karata na raznim događajima i dizajn baze podataka za web aplikaciju. Predvidjeti najmanje dva različita profila korisnika: moderator događaja koji zadaje kategorije karate i broj karata po kategoriji, potrebne datume i ostale relevantne podatka; kupac koji se treba registrirati a nakon toga i rezervirati karte za neki objavljeni događaj. Potrebno je opisati postupak izrade web aplikacije.

2. SLIČNE POSTOJEĆE WEB STRANICE

Postoje brojne web aplikacije koje korisnicima omogućuju pretraživanje i kupnju karata. U ovom poglavlju opisane su neke od njih te njihove funkcionalnosti.

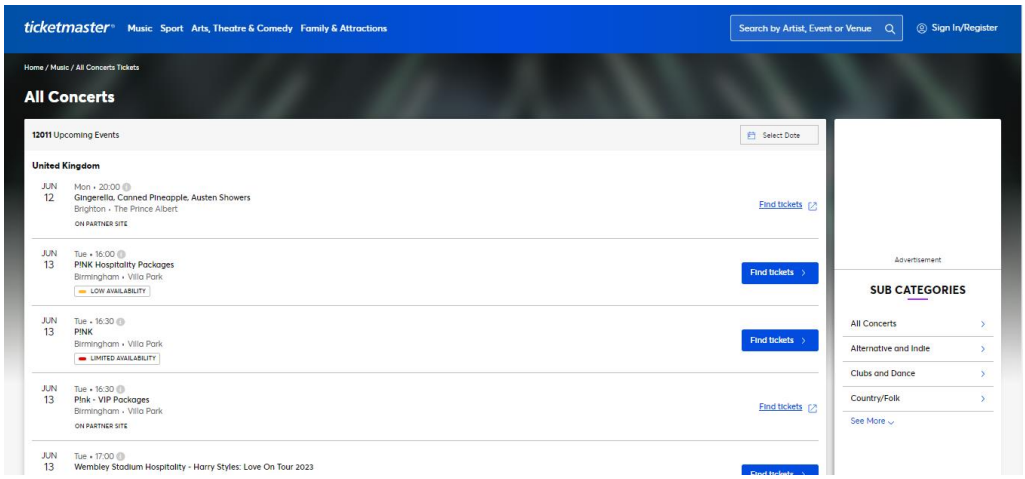
2.1. Ticketmaster

Ticketmaster jedna je od najpoznatijih i najvećih platformi za rezerviranje karata u svijetu. Korisnicima omogućuje prijavu i registraciju računa, pretraživanje događaja poput koncerata, predstava, sportskih događaja i ostalih događaja za koje se prodaju karte te daje mogućnost korisnicima prodaju svojih karata. Odabirom pojedinog događaja otvara se stranica s dodatnim detaljima o tom događaju te mogućnost odabira mjesta sjedenja i željenog broja karata te kupnja karata [1]. Slika 2.1. prikazuje dio izgleda početne stranice Ticketmastera za Ujedinjeno Kraljevstvo.



Slika 2.1. *Prikaz početne stranice Ticketmastera za Ujedinjeno Kraljevstvo*

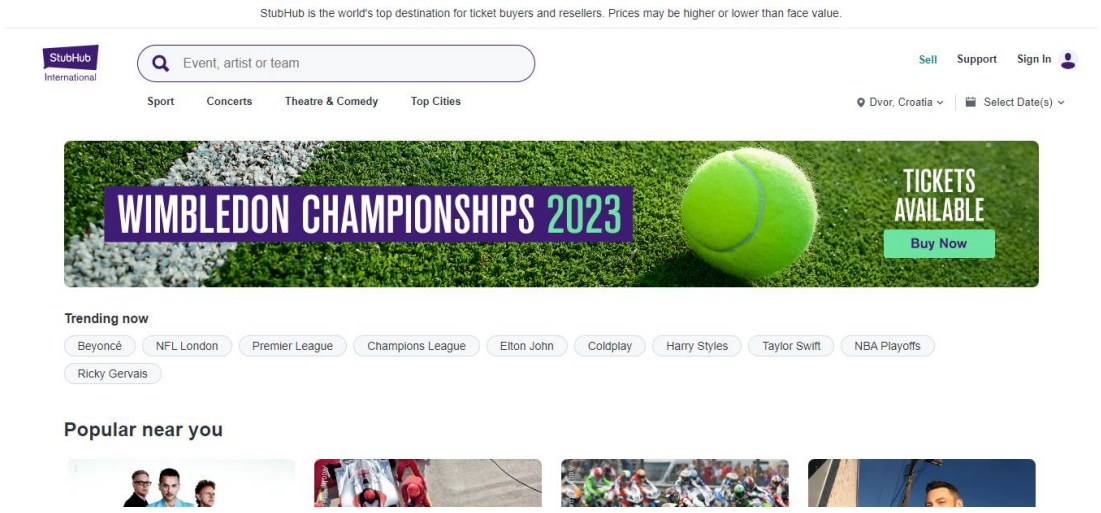
Slika 2.2. prikazuje stranicu za odabir željenog koncerta



Slika 2.2. *Prikaz stranice za odabir željenog koncerta na platformi Ticketmaster*

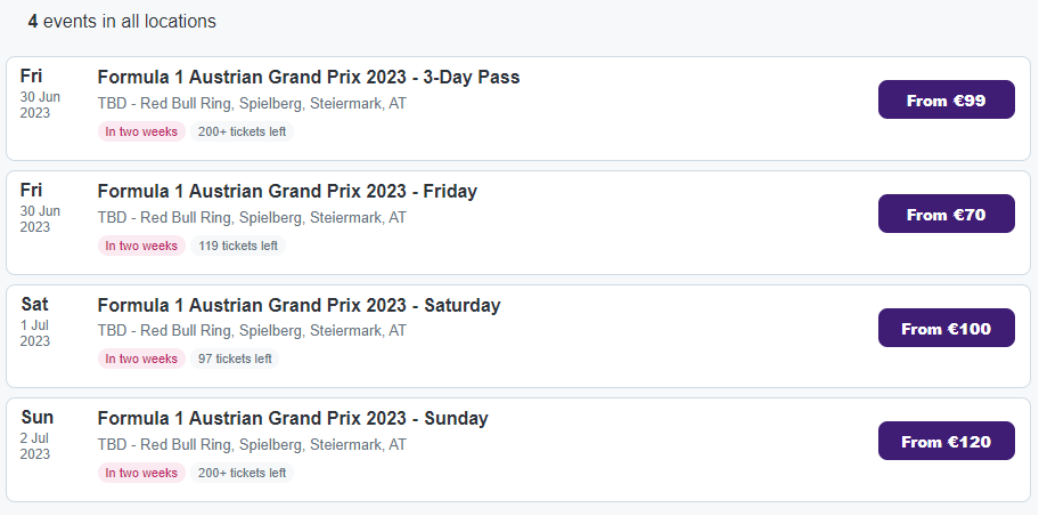
2.2. StubHub

StubHub je popularna platforma za prodaju i kupnju karata za različite vrste događaja putem interneta. Omogućuje korisnicima prijavu i registraciju računa, pretraživanje događaja poput koncerata, predstava, sportskih događaja te ostalih događaja za koje prodaju karte. Uz to omogućava korisnicima prodaju karata za svoje događaje. Korisnici mogu pretražiti događaje po kategorijama te odabirom pojedinog događaja otvara se stranica s dodatnim informacijama o događaju kao i vrstama karata koje korisnik može kupiti [2]. Slika 2.3. prikazuje izgled početne stranice StubHuba.



Slika 2.3. *Prikaz početne stranice StubHuba*

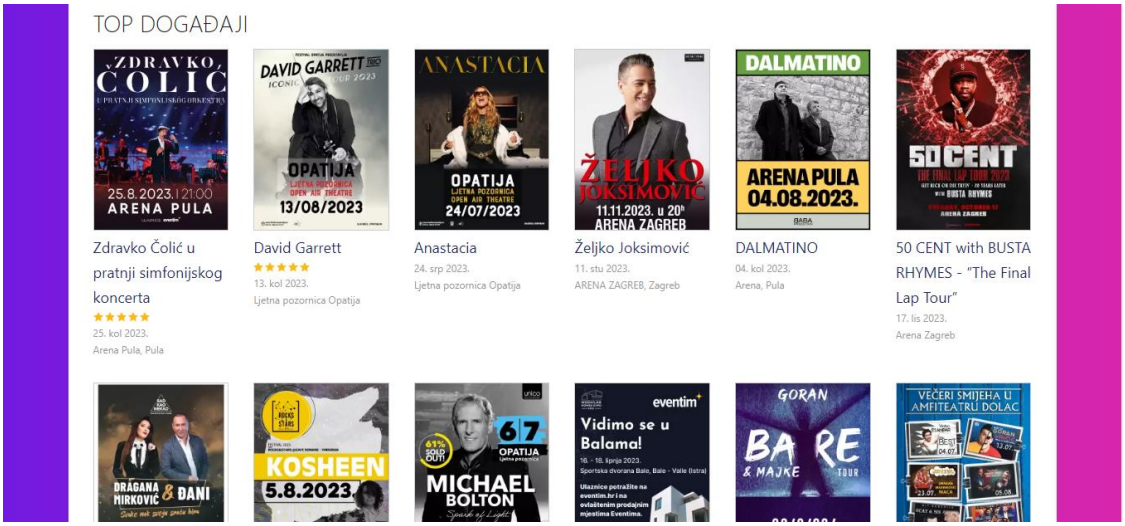
Slika 2.4 prikazuje različite vrste ponuđenih karata za jedan događaj.



Slika 2.4. *Prikaz različitih vrsta karata za jedan događaj*

2.3. Eventim

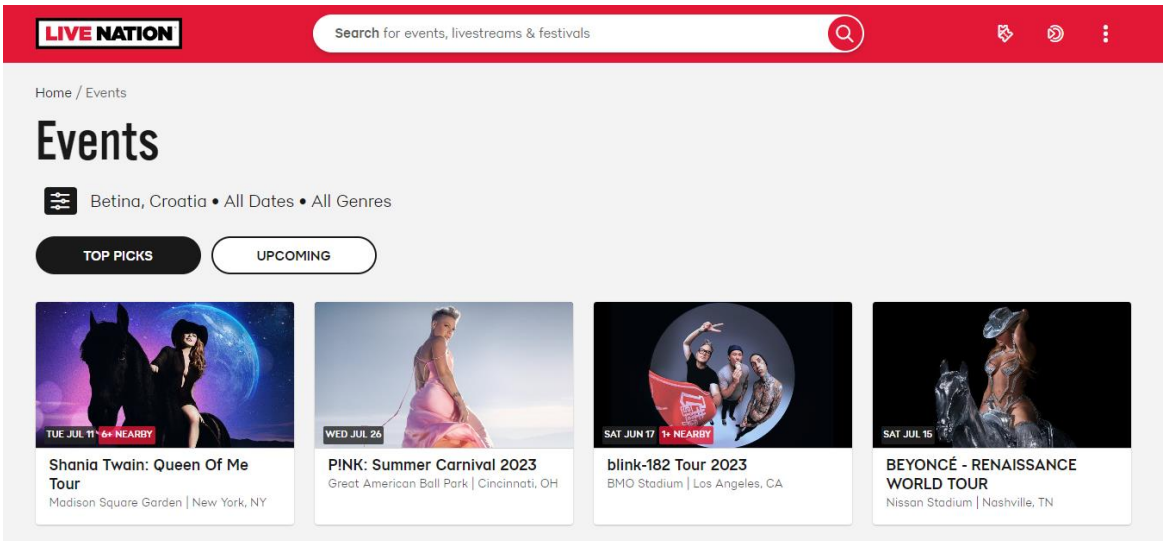
Eventim je vodeća europska platforma za prodaju ulaznica za različite događaje kao što su koncerti, festivali, sportski događaji, kazališne predstave i ostali događaji. Kao i prethodne platforme, omogućuje korisnicima izradu korisničkog računa te pregledavanje dostupnih događaja. Odabirom pojedinog događaja korisnik dobiva uvid u dodatne informacije o događaju kao i mogućnost izbora mjesta sjedenja ili stajanja i željenog broja karata. Slika 2.5. prikazuje listu popularnih događaja na portalu Eventima za Republiku Hrvatsku [3].



Slika 2.5. *Prikaz popularnih događaja na portalu Eventim*

2.4. Live Nation

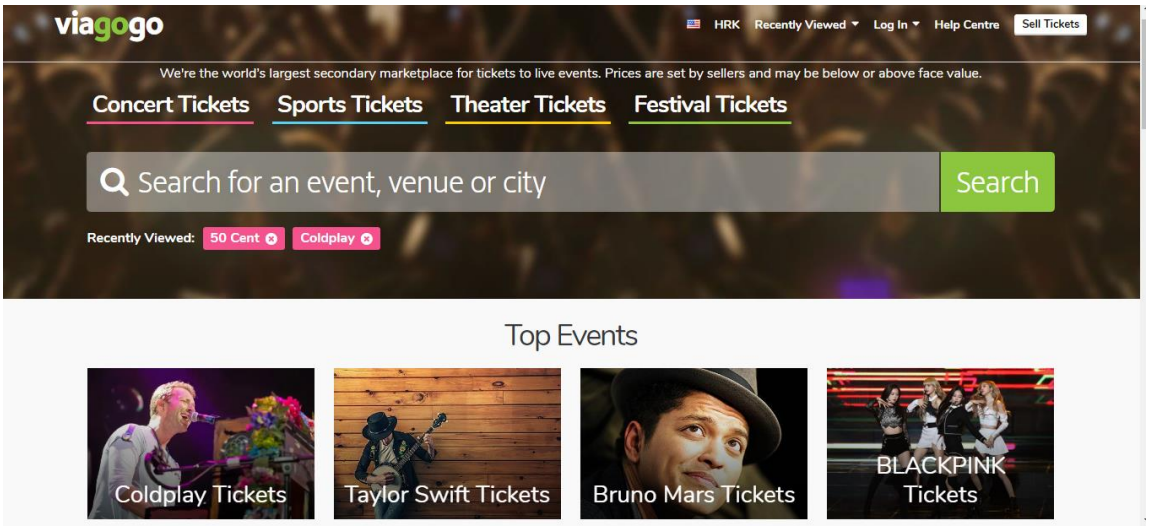
Live Nation je globalna kompanija specijalizirana za organiziranje i prodaju karata za koncerte i druge događaje. Kao i na drugim platformama korisnik može izraditi račun i pregledati sve događaje koji su u ponudi. Jedina razlika nalazi se prilikom same kupnje karata. U nekim slučajevima kupnja karata se ne odvija na samoj stranici već se klikom na gumb za kupnju karata otvara stranica za odabrani događaj na platformi Ticketmaster. To se događa zato jer je Ticketmaster službeni partner za prodaju karata koju promovira Live Nation. Obje platforme, Ticketmaster i Live Nation, nalaze se u vlasništvu kompanije Live Nation Entertainment. Slika 2.6. prikazuje početnu stranicu Live Nation portala [4].



Slika 2.6. *Prikaz početne stranice Live Nation portala*

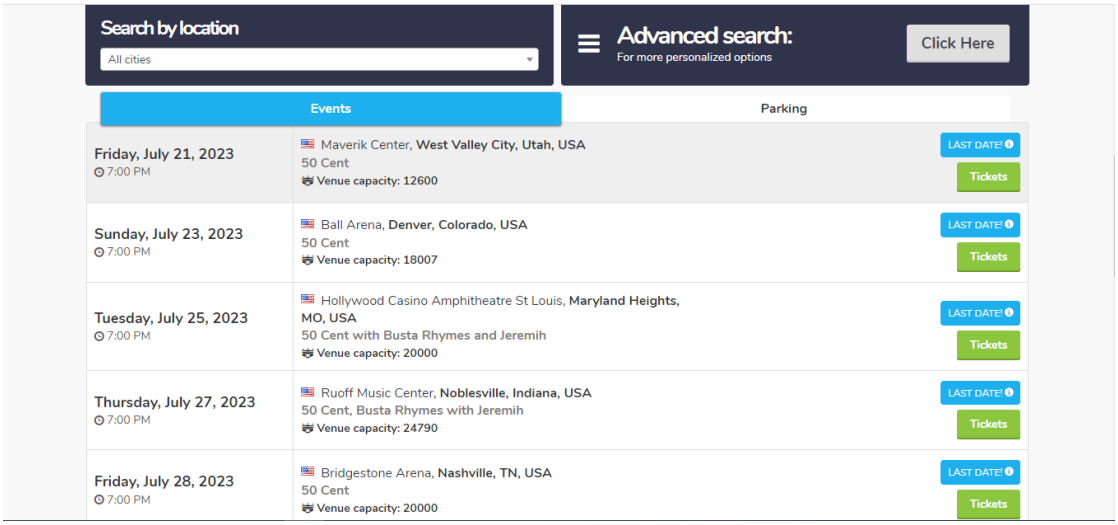
2.5. Viagogo

Viagogo je međunarodna platforma za kupnju i prodaju karata za različite događaje poput koncerata, predstava, stand-up komedija i sličnih događaja. Kao i kod ostalih stranica, korisnik može izraditi korisnički račun, pretražiti događaje po kategorijama te za pojedini događaj vidjeti više informacija te odabrati mjesto sjedenja i broj karata koji želi kupiti [5]. Slika 2.7. prikazuje početnu stranicu portala Viagogo.



Slika 2.7. *Početna stranica portala Viagogo*

Slika 2.8. prikazuje mogućnost odabira koncerta po datumu i mjestu za odabranog izvođača kao i mogućnost pretrage po lokaciji te napredne postavke za pretragu.



Slika 2.8. *Prikaz dostupnih događaja za odabranog izvođača i opcije pretrage*

3. IZRADA APLIKACIJE I KORIŠTENE TEHNOLOGIJE

U ovom poglavlju objašnjen je proces izrade web aplikacije za rezervaciju karata na događajima, od povezivanje projekta s Firebaseom do prikaza samog koda aplikacije. Također navedene su i objašnjene korištene tehnologije.

3.1. Korištene tehnologije

U ovom potpoglavlju navedene su sve tehnologije koje su korištene prilikom izrade ove web aplikacije. HTML, CSS i Materialize CSS korišteni su pri stvaranju izgleda stranice, za prikaz sadržaja korištene su Materialize kartice i modalni prozori, JavaScript i jQuery za izvođenje funkcionalnosti stranice te Firebase jer pruža mogućnosti pohrane podataka, u slučaju ove web aplikacije vezanih za događaje i korisnike i autentikacije korisnika na jednoj platformi.

Dodatno je objašnjen Materialize CSS s obzirom na često korištenje i široku rasprostranjenost ostalih tehnologija

3.1.1. Materialize CSS

Materialize CSS je moderan responzivan frontend okvir (engl. Framework) temeljen na Material Designu [6]. Koristi se za brzo i jednostavno dizajniranje responzivnih web stranica. Izgrađen je s naglaskom na responzivnost što znači da će se raspored elemenata automatski prilagoditi veličini ekrana na kojem se prikazuje sadržaj. Sadrži veliku paletu boja, animacija i efekata kao i velik broj gotovih komponenti i stilizacija koje se mogu koristiti u projektima. Pored CSS-a, Materialize CSS dolazi s nekoliko JavaScript komponenti kako bi se omogućile interaktivne značajke web stranice, poput padajućih izbornika, pokretnih transparenta, automatskog dovršavanja teksta te brojnih drugih komponenti.

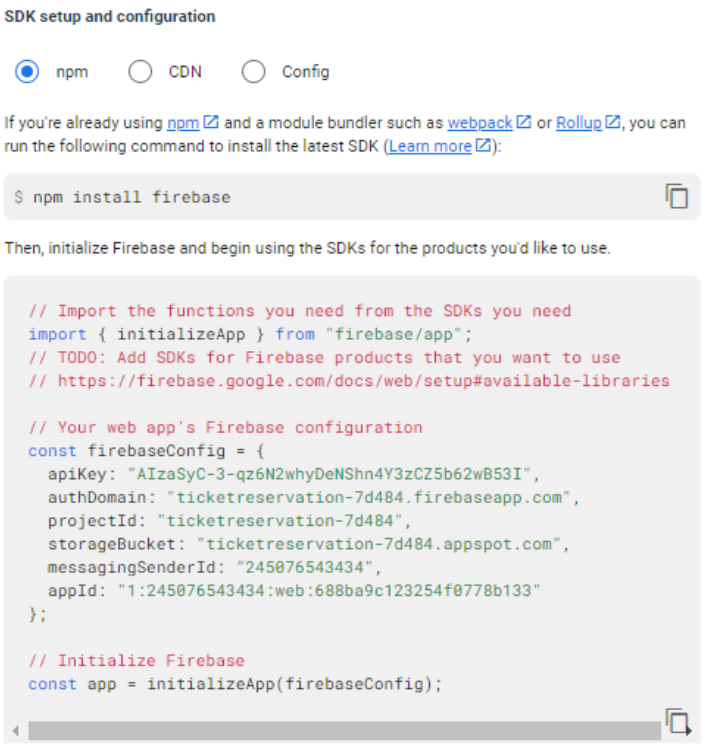
U ovom završnom radu Materialize CSS je korišten za responzivne dijelove stranice. Korišteni su predlošci za modalne prozore, responzivnu navigacijsku traku, gumbe i polja za unos podataka.

3.2. Izrada aplikacije

U ovom potpoglavlju objašnjen je proces izrade web aplikacije za rezervaciju karata na događajima, od povezivanje projekta s Firebaseom do prikaza koda i objašnjavanja funkcija aplikacije.

3.2.1. Povezivanje projekta s Firebaseom

Prije početka izrade aplikacije potrebno je izraditi novi projekt na Firebaseu za što je potrebno imati izrađen račun na platformi. Firebase omogućuje stvaranje stvarno-vremenskih baza podataka, opsežnu autentikaciju i autorizaciju te čak i implementaciju [7]. Detaljne upute za povezivanje projekt s Firebaseom navedene su na glavnoj stranici Firebasea pod nazivom Dodavanje Firebase u JavaScript projekt (engl. Add Firebase to your JavaScript project) [8]. U glavnom izborniku ponuđene je mogućnost izrade novog projekta. Za izradu projekta potrebno je dodijeliti ime projektu, odabratit želil li se u projektu koristiti Google analitika te odabrati jedan od brojnih poslužitelja na kojem će se nalaziti projekt. Preporučeno je odabrati poslužitelj koji je najbliže korisničkoj skupini koja će koristiti aplikaciju. Nadalje, potrebno je povezati projekt s Firebaseom. Potrebno je instalirati Firebase unutar projekta uz pomoć naredbe u terminalu *npm install firebase*. Također potrebno je stvoriti novi projekt na platformi Firebase. Nakon što je projekt izrađen potrebno je otići u postavke projekta te pronaći SDK (engl. Software development kit), odnosno set razvojnih alata, za postavljanje i konfiguraciju projekta koji se treba dodati u aplikaciju. SDK je skup programskih alata i programa koji pomažu pri izradi aplikacija za specificirane platforme ili programske jezike[9]. Slika 3.1. prikazuje izgled SDK unutar postavki na Firebaseu.



Slika 3.1. *Prikaz SDK-a unutar postavki na Firebaseu*

Nakon toga potrebno je SDK dodati u projekt. U projektu pod nazivom *firebase.js* nalazi se konfiguracija i inicijalizacija projekta kao i dohvaćanje usluga za autentikacija i *Cloud Firestore.* U varijable *db* i *auth* spremaju se instance dobivenih objekata kako bi se mogle koristiti u daljnjem dijelu programa. Slika 3.2. prikazuje kod unutar *firebase.js* datoteke.

```
scripts > JS firebase.js > ...
1  import { initializeApp } from 'https://www.gstatic.com/firebasejs/9.22.0/firebase-app.js';
2  import { getAuth } from 'https://www.gstatic.com/firebasejs/9.22.0/firebase-auth.js';
3  import { getFirestore } from 'https://www.gstatic.com/firebasejs/9.22.0/firebase-firestore.js';
4
5  // TODO: Add SDKs for Firebase products that you want to use
6  // https://firebase.google.com/docs/web/setup#available-libraries
7
8
9  // Your web app's Firebase configuration
10 const firebaseConfig = {
11   apiKey: "AIzaSyC-3-qz6N2whyDeNShn4Y3zCZ5b62wB53I",
12   authDomain: "ticketreservation-7d484.firebaseapp.com",
13   projectId: "ticketreservation-7d484",
14   storageBucket: "ticketreservation-7d484.appspot.com",
15   messagingSenderId: "245076543434",
16   appId: "1:245076543434:web:688ba9c123254f0778b133"
17 };
18
19 // Initialize Firebase
20 const app = initializeApp(firebaseConfig);
21 //make auth and firebase reference
22 const auth = getAuth(app);
23 const db = getFirestore(app);
24 export {auth, db}
```

Slika 3.2. *Kod unutar firebase.js datoteke*

3.2.2. Prikaz i objašnjenje koda aplikacije

Za pravilnu funkcionalnost početne stranice potrebno je više datoteka. Datoteka *index.html* u kojoj se nalaze elementi i raspored stranice, *index.js* u kojoj se nalaze funkcije za rukovanje događajima i ostale funkcionalnosti web stranice, poput prikaza elemenata iz baze na početnu stranicu i *auth.js* datoteka koja služi za potvrdu autentikacije i dohvaćanje podatak iz baze.

Na početku *index.html* dokumenta nalazi se navigacijska traka koja ovisno o tome radi li se o moderatoru ili kupcu mijenja svoj izgled. Mogućnosti prikazane na traci prilikom pojave događaja *click* na jednu od njih otvaraju modalne prozore koji služe za prijavu i registraciju korisnika, kreiranje novog događaja te pregled informacija o korisničkom računu. Elementi navigacijske trake s nazivom klase *logged-out* vidljivi su kada korisnik nije prijavljen u sustav, a elementi s nazivom klase *logged-in* vidljivi su nakon prijave u sustav. Element s oznakom *Id-a admin* vidljiv je samo moderatorima. Na slici 3.3. prikazan je HTML kod za navigacijsku traku.

```
<!-- NAVBAR -->
<nav class="z-depth-0 grey lighten-4">
  <div class="nav-wrapper container">
    <a href="#" class="brand-logo">
      <a href="index.html" class="black-text modal-trigger" >Ticekt Reservation</a>
    </a>
    <a href="#" class="sidenav-trigger" data-target="mobile-nav">
    <i class="material-icons black-text">menu</i></a>
    <ul class="right hide-on-med-and-down">
        <li class="logged-in" style="display: none;">
          <a href="#" class="grey-text modal-trigger" data-target="modal-account">Account</a>
        </li>
        <li class="logged-in" style="display: none;">
          <a href="#" class="grey-text" id="logout">Logout</a>
        </li>
        <li class="logged-in" id="admin" style="display: none;">
          <a href="#" class="grey-text modal-trigger" data-target="modal-create">Create Event</a>
        </li>
        <li class="logged-out" style="display: none;">
          <a href="#" class="grey-text modal-trigger" data-target="modal-login">Login</a>
        </li>
        <li class="logged-out" style="display: none;">
          <a href="#" class="grey-text modal-trigger" data-target="modal-signup">Sign up</a>
        </li>
    </ul>
  </div>
</nav>
```

Slika 3.3. *HTML kod za navigacijsku traku*

Na slici 3.4. prikazan je HTML kod u slučaju da se sadržaj prikazuje na manjem ekranu poput mobilnog uređaja. Kao i kod obične navigacijske trake elementi imaju različita imena klasa kako bi se mogli prikazivati ovisno o tome je li korisnik prijavljen ili ne.

```
<ul class="black sidenav" id="mobile-nav">
    <li class="logged-in" style="display: none;">
      <a href="#" class="grey-text modal-trigger" data-target="modal-account">Account</a>
    </li>
    <li class="logged-in" style="display: none;">
      <a href="#" class="grey-text" id="logout">Logout</a>
    </li>
    <li class="logged-in" id="admin" style="display: none;">
      <a href="#" class="grey-text modal-trigger" data-target="modal-create">Create Event</a>
    </li>
    <li class="logged-out" style="display: none;">
      <a href="#" class="grey-text modal-trigger" data-target="modal-login">Login</a>
    </li>
    <li class="logged-out" style="display: none;">
      <a href="#" class="grey-text modal-trigger" data-target="modal-signup">Sign up</a>
    </li>
</ul>
```

Slika 3.4. *HTML Kod za navigacijsku traku na uređajima s manjim ekranom*

U oba slučaja za uređivanje navigacijske trake kao i za ostale elemente korišten je Materialize CSS. Kako bi navigacijska traka bila prikazana na uređajima s manjim ekranima u HTML kod potrebno je dodati jQuery metodu *sidenav()*, koja je dio Materialize CSS biblioteke, koja se poziva

na svim elementima koji imaju klasu *sidenav*. Glavne značajke jQuerya pomažu pri pristupu elementima u dokumentu, mijenjanju izgleda web stranice, promjeni sadržaja dokumenta, odgovoru na korisničku interakciju, animaciji promjena u dokumentu, povratu informacija s poslužitelja bez osvježavanja stranice i pojednostavljenju JavaScript zadataka [10]. Slika 3.5. prikazuje jQuery kod za prikaz navigacijske trake kod uređaja s manjim ekranima.

```
<script>
 $(document).ready(function(){
    $('.sidenav').sidenav();});
</script>
```

Slika 3.5. *jQuery kod za prikaz navigacijske trake kod uređaja s manjim ekranima*

Kako bi se prikazali različiti elementi na početnoj stranici ovisno o tome je li korisnik prijavljen ili ne koristi se funkcija *setupUIIndex*. Funkcija *setupUIIndex* poziva se unutar *onAuthStateChanged* funkcije unutar *auth.js* datoteke zajedno sa *setupUIEvnetPage* funkcijom koja služi za prikaz određenih elemenata na stranici za pojedini događaj. Funkcija provjerava autentikaciju korisnika te predaje objekt tipa *user*, zajedno s funkcijama *makeAdmin* i *isUserAdmin* funkcijama *setupUIIndex* i *setupUIEvnetPage* ovisno o tome na kojoj se stranici korisnik nalazi. Za pisanje ove i drugih funkcija korištena je streličnih funkcija (engl. arrow functions). Postoje dvije prednosti streličnih funkcija. Prvo. manje su opsežne od tradicionalnih funkcija. Drugo, njihov *this* se uzima iz okoline (leksički). Stoga, više nije potrebna funkcija *bind()* ili *this=that* [11]. Slika 3.6. prikazuje kod *onAuthStateChanged* funkcije.

```
//listen for auth status changes
onAuthStateChanged(auth, user=>{
    async function makeAdmin(email){
        const userRef = doc(db, 'users', email);
        return setDoc(userRef, {
          admin: true
        });
      }
      async function isUserAdmin(user){
        const docRef = doc(db, "users", user.email);
        const docSnap = await getDoc(docRef);
        console.log(docSnap.data().admin)
        if(docSnap.data().admin==true){
          return true
        }
        else {
         return false
        }
      }
    if(urlId==null)
    {
    if(user){
        setupUIindex(user, makeAdmin, isUserAdmin)
    }
    else{
        setupUIindex()
    }
    }
    else{
        if(user){
            setupUIEventPage(user,makeAdmin, isUserAdmin)
        }
        else{
            setupUIEventPage()
        }
    }
});
```

Slika 3.6. *Kod onAuthStateChanged funkcije*

Ako *setupUIIndex* primi vrijednost različitu od *null* vrijednosti, odnosno praznog objekta, poziva funkciju *isUserAdmin* kako bi se vidjelo radi li se o korisniku koji je moderator ili o korisniku koji je kupac, te ovisno o tome postavlja vidljivost određenih elemenata i mogućnosti na stranici. *isUserAdmin* funkcija pomoću elektroničke pošte prijavljenog korisnika dohvaća dokument u kolekciji *users* s podudarajućom adresom elektroničke pošte te provjerava atribut *admin* i vraća vrijednost t*rue* ili f*alse* ovisno o vrijednosti atributa *admin.* U slučaju da je korisnik moderator, odnosno da je funkcija *isUserAdmin* vratila vrijednost *true*, moderator u *Account* modalnom prozoru može postaviti ulogu drugog korisnika u moderator unošenjem elektroničke pošte tog korisnika. Slika 3.7. prikazuju kod *setupUIIndex.*

```
const setupUIIndex=(user, makeAdmin, isUserAdmin)=>{
  if(user){
    const html=`<div>Logged in as ${user.email}</div>`
      accountDetails.innerHTML=html
    isUserAdmin(user).then((isAdmin) => {
      if (isAdmin) {
        console.log(isAdmin);
        loggedInLinks.forEach((item) => (item.style.display = 'block'));
        loggedOutLinks.forEach((item) => (item.style.display = 'none'));
        makeAdminForm.addEventListener("submit", (event)=>{
          event.preventDefault()
          const userEmail=makeAdminForm['account-make-admin'].value
          makeAdmin(userEmail)
          const modal = document.querySelector("#modal-account");
          M.Modal.getInstance(modal).close();
          makeAdminForm.reset();
        })
        makeAdminForm.style.display='block'
      }
      else{loggedInLinks.forEach((item) => {
        if (item.id === 'admin') {
          item.style.display = 'none';
        } else {
          item.style.display = 'block';
        }
      });
      loggedOutLinks.forEach((item) => (item.style.display = 'none'));}})
      makeAdminForm.style.display="none"
  }
  else{
    accountDetails.innerHTML=''
    loggedInLinks.forEach(item => item.style.display='none')
    loggedOutLinks.forEach(item=> item.style.display='block')
    makeAdminForm.style.display="none"
  }
}
```

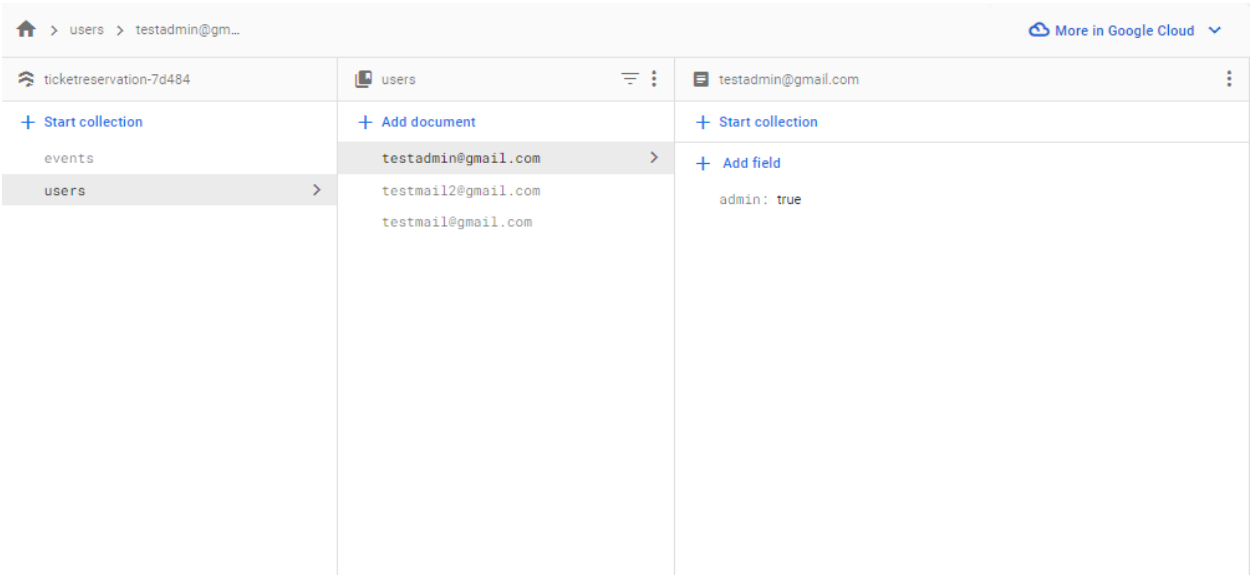
Slika 3.7. *JavaScript kod setupUIIndex funkcije*

Za stvaranje računa potreban je unos adrese elektroničke pošte i zaporke. Za kreiranje korisničkog računa potrebno je zadovoljiti uvjete za stvaranje novog računa poput uvjeta da se adresa elektroničke pošte ne koristi za neki drugi korisnički račun, odnosno da se ne nalazi u bazi već korištenih adresa za autentikaciju i da je zaporka dovoljno dugačka. Slika 3.8. prikazuje JavaScript kod za registraciju korisnika.

```
const signupForm=document.querySelector('#signup-form')
signupForm.addEventListener('submit',(e)=>{
    e.preventDefault();
    //get user info
    const email=signupForm['signup-email'].value
    const password=signupForm['signup-password'].value
    //sign up the user
createUserWithEmailAndPassword(auth, email, password).then(cred=>{
    const userUid = cred.user.uid;
    const userRef = doc(db, 'users', email);
    return setDoc(userRef, {
      admin: false
    });
  }).then(() => {
    const modal = document.querySelector("#modal-signup");
    M.Modal.getInstance(modal).close();
    signupForm.reset();
  }).catch((error) => {
    console.log(error);
  });
})
```

Slika 3.8. *JavaScript koda za registraciju korisnika*

Funkcija za registraciju korisnika u sustav osluškuje događaj *submit* obrasca *singup-form* unutar modalnog prozora s oznakom *modal-singup* u HTML dokumentu. Kada se taj događaj dogodi, očitavaju se vrijednosti iz obrasca za registraciju te se predaju funkciji *createUserWithEmailAndPassword* kao argumenti zajedno s *auth* instancom. Prilikom stvaranja novog korisnika, uz stvaranje objekta unutar *Firebase Authenticationa* koji će se kasnije koristit za prijavu korisnika u sustav, adresu elektroničke pošte korisnika dodaje se u kolekciju u bazi podataka pod nazivom *users*. Svaki dokument kolekcije nosi ime elektroniče pošte korisnika te sadrži jedan atribut pod nazivom *admin* koji može imati vrijednost *true* ili *false* za određivanje radi li se o moderatoru (vrijednost *true*) ili kupcu (vrijednost *false*). Slika 3.9. prikazuje izgled kolekcije *users*.



Slika 3.9. *Izgled kolekcije users*

Funkcija za prijavu korisnika u sustav osluškuje događaj *submit* obrasca *login-form* unutar modalnog prozora s oznakom *modal-login* u HTML dokumentu. Kada se taj događaj dogodi, unesena adresa elektroničke pošte i zaporka predaju se funkciji *signInWithEmailAndPassword* zajedno s instancom objekta *auth* kako bi se korisnika prijavilo u sustav te se nakon toga zatvara modalni prozor i čiste se polja za unos kako ne bi bila popunjena kod sljedećeg unosa. Na slici 3.10. prikazan je JavaScript kod za modalni prozor za prijavu korisnika u sustav.

```
//login
const loginForm=document.querySelector('#login-form')
loginForm.addEventListener('submit',(e)=>{
    e.preventDefault();
    //get user info
    const email=loginForm['login-email'].value;
    const password=loginForm['login-password'].value;
    signInWithEmailAndPassword(auth, email, password).then(cred => {

        //close the login modal and reset
        const modal=document.querySelector('#modal-login');
        M.Modal.getInstance(modal).close();
        loginForm.reset();

    });
});
```

Slika 3.10. *JavaScript kod za prijavu korisnika u sustav*

Kada je korisnik prijavljen u sustav na navigacijskoj traci pojavljuju se dvije opcije, *Account* i *Logout*, a ako je korisnik moderator pojavljuje se i *Create Event* .

Funkcija za odjavu korisnika iz sustava osluškuje događaj *click* gumba s oznakom *logout* u HTML dokumentu. Pri pojavi tog događaja, poziva se funkcija *signOut* kojo se predaje instanca objekta *auth* te se korisnika odjavljuje iz sustava. Nakon odjave na stranici se prikazuje sadržaj koji u HTML dokumentu ima klasu pod nazivom *logged-out.* Na slici 3.11. prikazan je JavaScript kod za odjavljivanje korisnika iz sustava.

```
//logout
const logout=document.querySelector('#logout');
    logout.addEventListener('click', (e)=>{
        e.preventDefault();
        signOut(auth);
    });
```

Slika 3.11. *JavaScript kod za odjavljivanje korisnika iz sustava*

Gumb *Account* na navigacijskoj traci osluškuje događaj *click* na sebe te potom otvara modalni prozor s oznakom *modal-account* gdje korisnik može vidjeti s kojom je adresom elektroničke pošte prijavljen. U slučaju da je korisnik moderator pojavljuje se i polje za unos adrese elektroničke pošte koje služi za postavljanje drugog korisnika kao moderatora.

Funkcija *makeAdmin* u kolekciji *users* traži adresu elektroničke pošte koja se podudara s unesenom adresom i mijenja vrijednost atributa *admin* iz *false* u *true.* Slika 3.12. prikazuje JavaScript kod za funkciju *makeAdmin* koja se poziva unutar *setupUIIndex* funkcije.

```
async function makeAdmin(email){
  const userRef = doc(db, 'users', email);
  return setDoc(userRef, {
    admin: true
  });
}
```

Slika 3.12. *JavaScript kod za funkciju makeAdmin*

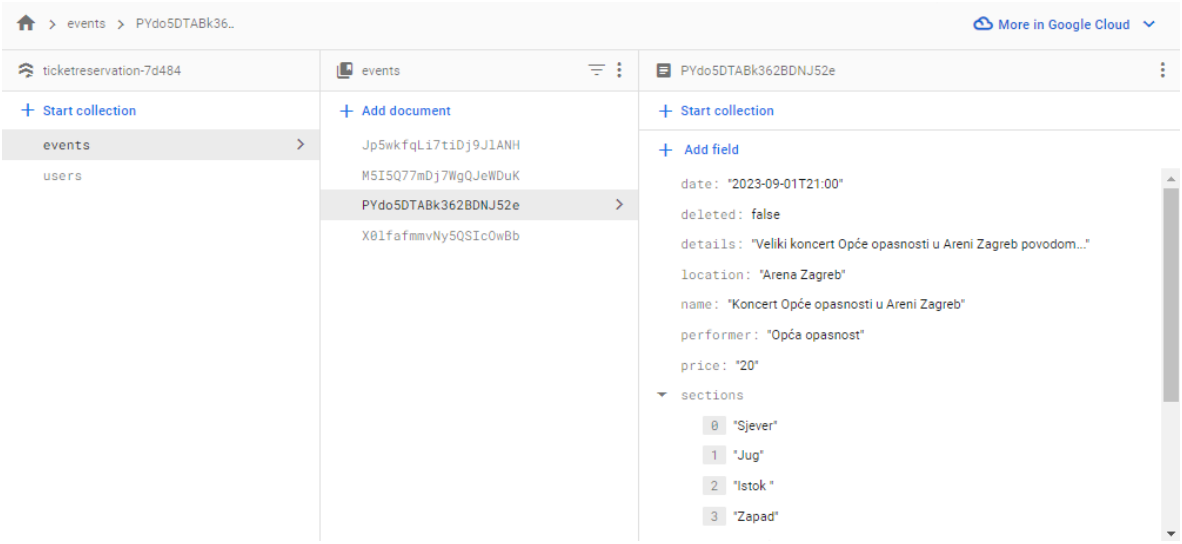
U slučaju da je korisnik moderator na navigacijskoj traci pojavit će se *Create Event* koji osluškuje pojavu događaja *click* na sebe te otvara modalni prozor za stvaranje novog događaja s oznakom *modal-create* u HTML dokumentu.

Funkcija za stvaranje novog događaja osluškuje događaj *submit create-form* obrasca unutar modalnog prozora *modal-create*. Pri aktivaciji funkcije, vrijednosti unesene u polja za unos spremaju se u varijable te se poziva funkcija *addDoc* koja stvara novi dokument u kolekciju *users* na Cloud Firestoreu. Cloud Firestore je NoSQL baza podataka koja služi za pohranu i sinkronizaciju podataka u stvarnom vremenu[12]. Slika 3.13. prikazuje JavaScript kod za stvaranje novog događaja.

```
const createForm=document.querySelector('#create-form')
var itemSections,itemTickets,name,performer,details,date,location,price,url,type
createForm.addEventListener('submit',(e)=>{
    e.preventDefault();
    var inputSections=document.getElementById("event-sections").value
    var inputTickets=document.getElementById("event-sections-tickets").value
    itemSections=inputSections.split(",");
    itemTickets=inputTickets.split(",");
    name= document.getElementById('event-name').value,
    performer= document.getElementById('event-performer').value,
    details= document.getElementById('event-details').value,
    date= document.getElementById('event-date').value,
    location= document.getElementById('event-location').value,
    price= document.getElementById('event-price').value,
    url= document.getElementById('event-image').value
    type=document.getElementById('event-type').value
    const modal=document.querySelector("#modal-create");
    M.Modal.getInstance(modal).close();
    createForm.reset();
    addDocument()
})
async function addDocument(){
const docRef=await addDoc(collection(db, 'events'),{
    name: name,
    performer: performer,
    details: details,
    date: date,
    location: location,
    price: price,
    sections: itemSections,
    sectionstickets: itemTickets,
    url: url,
    type:type,
    deleted:false
})
}
```

3.13. *JavaScript kod za stvaranje novog događaja*

Slika 3.14. prikazuje izgled kolekcije *events* koja sadrži *Id* događaja i attribute potrebne za izradu događaja



Slika 3.14. *Izgled kolekcije events*

Kako bi se modalni prozori nakon pojave događaja *click* ili *submit* otvorili te kako bi se pravilno zatvorili nakon završetka radnje ili prilikom radnje *click* izvan modalnog prozora potrebno je dodati *addEventListener* na *DOMContentLoaded* kako nalaže Materialize CSS dokumentacija. Slika 3.15. prikazuje JavaScript kod koji omogućuje prikaz i zatvaranje modalnih prozora.

```
document.addEventListener('DOMContentLoaded', function() {
    // Init all modals
    var modals = document.querySelectorAll('.modal');
    M.Modal.init(modals);
    // Init all collapsibles
    var collapsibles = document.querySelectorAll('.collapsible');
    M.Collapsible.init(collapsibles);
})
```

Slika 3.15. *JavaScript kod koji prikazuje pravilan prikaz i zatvaranje modalnih prozora*

Ispod navigacijsku traku na početnoj stranici prikazani su i svi dostupni događaji. Prilikom učitavanja stranice poziva se funkcija *getDocs* koja dohvaća sve elemente kolekcije *events* te za svaki od elemenata poziva funkciju *setupEvents.* Slika 3.16. prikazuje poziv funkcije *getDocs* i funkcije *setupEvents* za svaki od elemenata u bazi.

```
const querySnapshot = await getDocs(collection(db, "events"));
querySnapshot.forEach((doc) => {
  // doc.data() is never undefined for query doc snapshots
    let data=(doc.id, " => ", doc.data())
    let id=doc.id
    setupEvents(data, id)
```

Slika 3.16. *Poziv funkcije getDocs i funkcije setupEvents za svaki od elemenata u bazi*

Funkcija *setupEvents* pomoću funkcije *querySelector* traži mjesto u datoteci *indeks.html* na kojem se nalazi element s oznakom *event.* Nakon toga, funkcija provjerava je li primila podatke iz baze (provjerava imaju li varijable *data* i *id* vrijednost jednaku *null*) te ako nije ispisuje poruku na ekranu. U slučaju da je funkcija primila potrebno podatke, unutar kosih navodnika složena je struktura izgleda svakog od događaja u Materialize kartice. Kosi navodnici osim što mogu sadržavati znakove, mogu sadržavati i vrijednosti [13]. U ovom slučaju potrebni su kako bi se prikazala HTML struktura zajedno s vrijednostima JavaScript varijabli unutar znakova *${}*[14]. Prikazani su slika, ime, izvođač, datum, vrsta i lokacija svakog događaja kao i gumb *More info* koji vodi na stranicu za više informacija o pojedinom događaju. Pojavom događaja *click* na gumb *More info, window.location.href* naredba prebacuje stranicu s index.html na EventPage.html datoteku zajedno s vrijednosti *id* unutar putanje odabranog elementa, odnosno događaja. Slika 3.17. prikazuje funkciju *setupEvents.*

```
const eventList=document.querySelector(".event")
//setup events
let html='';
const setupEvents=(data, id)=>{
  if(data==null ||id==null){
    const li=`<div class="card"><p>Ther is no data</p><div>`;
    html+=li;
    eventList.innerHTML=html
  }
  else{
    if(data.deleted==false){
  const li=`
      <div class="col s12 m6 l3" >
      <div class="card large" >
        <div class="card-image">
          <img src="${data.url}" width="500" height="300">
          <span class="card-title black white-text" >${data.name}</span>
        </div>
        <div class="card-content">
          <p>${data.performer}</p>
          <p>${data.date}</p>
          <p>${data.type}</p>
          <p>${data.location}</p>
        </div>
        <div class="card-action">
        <a id="link-button-${id}" class="link-button" data-id="${id}">More info</a>
        </div>
      </div>
  </div>
      `;
      html+=li;
      eventList.innerHTML=html
      const linkButtons = document.getElementsByClassName("link-button");
      Array.from(linkButtons).forEach((button) => {
        button.addEventListener("click", (event) => {
          event.preventDefault();
          const id = button.dataset.id;
          localStorage.setItem("setId", id)
          const url = `./EventPage.html?id=${id}`;
          window.location.href = url;
        });
      });}

}}
```

Slika 3.17. *Prikaz funkcije setupEvents*

Prilikom otvaranja stranice iz URL-a (engl. *Uniform Resource Locator*) uzima se *id* te se pomoću *getDoc* i *doc* funkcija pronalazi element u kolekciji *events* s odgovarajućim *idom*. U slučaju da takav element postoji ostale informacije kao i mogućnost odabira željenih karata nalaze se unutar Materialize kartice. Na slici 3.18. prikazan je JavaScript kod za dohvaćanje događaja s vrijednosti *id* iz URL-a i njegov prikaz.

```
const queryString = window.location.search;
const urlParams = new URLSearchParams(queryString);
const urlId = urlParams.get('id');
var docRef
var docSnap
if(urlId){
 docRef = doc(db, "events", urlId);
 docSnap = await getDoc(docRef);
}
if (docSnap) {
  const data=docSnap.data()
const eventInfo=document.querySelector(".event-page")
let html=''
const getEvents=()=>{
    const li=`
    <div class="row">
  <div class="col s12 m6">
    <div class="card-image">
      <img src="${data.url}" class="responsive-img" style="padding-top:7px; object-fit: fill; width: 100%; height: 508px;">
    </div>
  </div>
  <div class="col s12 m6">
    <div class="card">
      <div class="card-content">
        <span class="card-title" style="font-size: 24px;">${data.name}</span>
        <p style="font-size: 18px;">Performer: ${data.performer}</p>
        <p style="font-size: 18px;">Date and time: ${data.date}</p>
        <p style="font-size: 18px;">Details: ${data.details}</p>
        <p style="font-size: 18px;">Price: ${data.price}€</p>
        <p style="font-size: 18px;">Select a seating section nad number of cards:</p>
        <div class="input-field">
          <select id="dropdown">
            <option value="" disabled selected>Select an option</option>
          </select>
        </div>
        <div class="input-field">
          <input placeholder="Number of tickets" id="ticket-number" type="number" class="validate">
          <label for="ticket-number">Number of tickets</label>
        </div>
      </div>
      <div class="card-action" style="display: flex; justify-content: space-between;">
      <p class="logged-out" style="display: none;"> Log in to reserve tickets</p>
        <button id="submit-btn" class="btn waves-effect waves-light" type="none" name="action">Submit
          <i class="material-icons right">send</i>
        </button>
        <div>
          <button id="update-btn" class="btn waves-effect waves-light modal-trigger" data-target="modal-update" name="action">Update</button>
          <button id="delete-btn" class="btn waves-effect waves-light modal-trigger red" name="action">Delete</button>
        </div>
      </div>
    </div>
  </div>
</div>
<style>
  img {
    height: 500px; /* Adjust the desired height */
  }
</style>
      <script> $(document).ready(function() {
        M.updateTextFields();
      });</script>
      `;
  html+=li;
  eventInfo.innerHTML=html
}
```

Slika 3.18. *JavaScript kod za prikaz pojedinog događaja*

Na slici 3.19. JavaScript kod za popunjavanje padajućeg izbornika s mogućim odabirom mjesta sjedenja.

```
getEvents()
const sections=data.sections
console.log(sections)
const sectionsDropdown=document.getElementById("dropdown")
for(let key in sections){
    let option=document.createElement("option")
    option.value = sections[key];
    option.textContent = sections[key];
    sectionsDropdown.appendChild(option);
    var elems = document.querySelectorAll('select');
    var instances = M.FormSelect.init(elems, option);
}
```

Slika 3.19. *JavaScript kod za popunjavanje padajućeg izbornika s mogućim odabirom mjesta sjedenja*

Funkcija za rezerviranje karata osluškuje pojavu događaja *click* na gumb s oznakom *submit-btn*. Pri pojavi tog događaja, poziva se funkcija *getTickets* koja uzima vrijednost iz padajućeg izbornika za odabranu vrstu i broj karata iz kućice za unos te mijenja vrijednost preostalog broja karata u bazi i ispisuje poruku o potvrdi rezervacije karata. U slučaju da korisnik zatraži više karata nego li je ostalo u sustavu ispisat će se poruka o tome koliko je karata ostalo za odabranu vrstu karte, a u slučaju da više nema karata ispisat će se poruka koja obavještava korisnika kako su sve karte rezervirane za tu vrstu karte. Slika 3.20. prikazuje JavaScript kod funkcije za rezerviranje karata.

```
const submitBtn=document.getElementById("submit-btn")
async function getTickets(){
    var section
    var sectionPicked=document.getElementById('dropdown').value
    const docRef=doc(db, 'events', urlId)
    for(var i=0; i<data.sections.length;i++){
        if(data.sections[i]==sectionPicked){
            console.log(data.sections[i])
        section=i
        }
    }
    var ticketNumber=parseInt(document.getElementById('ticket-number').value)
    if(data.sectionstickets[section]-ticketNumber>=0){
        console.log(data.sectionstickets[section])
        var sectionsTicketsLocal=data.sectionstickets
        console.log(sectionsTicketsLocal)
        sectionsTicketsLocal[section]-=ticketNumber
       await updateDoc(docRef,{
            sectionstickets:sectionsTicketsLocal
        })
        console.log(ticketNumber)
        alert("You have reserved "+ ticketNumber+ " tikcets for section "+sectionPicked)
        location.reload()
    }
    else if(data.sectionstickets[section]-ticketNumber<0){
        alert("There is only"+data.sectionstickets[section]+"tickets left for that section")
    }
    else if(data.data.sectionstickets[section]==0){
        alert("There is no more tickets for that section")
    }
}
submitBtn.addEventListener("click", (event)=>{
    getTickets()
})
```

Slika 3.20. *JavaScript kod funkcije za rezerviranje karata*

U slučaju da je korisnik moderator u desnom kutu Materialize kartice pojavljuju se dodatna dva gumba, *DELETE* i *UPDATE*. Funkcija *updateDoc* osluškuje pojavu događaja *click* na gumbu DELETE i postavlja vrijednost atributa *isDeleted* iz *false* u *true* trenutnom događaju te taj događaj više neće bit vidljiv na početnoj stranici. Slika 3.21. prikazuje JavaScript kod za brisanje događaja.

```
const eventDeleteBtn=document.getElementById('delete-btn')
eventDeleteBtn.addEventListener("click",(event)=>{
  eventDelete()
})
async function eventDelete(){
  let confirmAction=confirm("Are you sure you whant to delete this event?")
  if(confirmAction){
  await updateDoc(docRef,{
    deleted:true
  })
  window.location.href="index.html"
  }
  else{
  alert("Deletion canceleds")
  }
}
```

Slika 3.21. *JavaScript kod za brisanje događaja*

Iznad *DELETE* gumba nalazi se gumb *UPDATE* koji otvara modalni prozor nalik *CREATE EVENT* modalnom prozoru koji služi za promjenu vrijednosti atributa za trenutni događaj. Trenutne vrijednosti atributa nekog događaja prikazane su unutar polja za unos te ja za njihovu promjenu u bazi potrebno promijeniti njihove vrijednosti. Slika 3.22. prikazuje JavaScript kod za otvaranje i popunjavanje modalnog prozora s postojećim vrijednostima atributa događaja koji se želi izmijeniti.

```
const eventUpdateBtn=document.getElementById('update-btn')
eventUpdateBtn.addEventListener("click",(event)=>{
    eventUpdateForm()
})
async function eventUpdateForm(){
    const updateForm=document.querySelector('#update-form')
    document.getElementById('event-name-update').value=data.name
    document.getElementById('event-performer-update').value=data.performer
    document.getElementById('event-date-update').value=data.date
    document.getElementById('event-type-update').value=data.type
    document.getElementById('event-location-update').value=data.location
    document.getElementById('event-details-update').value=data.details
    document.getElementById('event-price-update').value=data.price
    document.getElementById('event-image-update').value=data.url
    document.getElementById('event-sections-update').value=data.sections
    document.getElementById('event-sections-tickets-update').value=data.sectionstickets

    updateForm.addEventListener("submit",(e)=>{
        e.preventDefault()
        eventUpdate()
        const modal=document.querySelector("#modal-update");
        M.Modal.getInstance(modal).close();
    })
}
```

Slika 3.22. *JavaScript kod za otvaranje i popunjavanje modalnog prozora s postojećim vrijednostima*

Funkcija *eventUpdate* poziva se pojavom događaja *click* na element *update-btn* unutar obrasca *update-form* koja sprema unesene vrijednosti u bazu. Slika 3.23. prikazuje JavaScript kod za funkciju *eventUpdate.*

```
async function eventUpdate(){
    var nameUpdate, performerUpdate, dateUpdate, typeUpdate, locationUpdate, detailsUpdate, priceUpdate, urlUpdate, sectionsUpdate, sectionticketsUpdate
    nameUpdate=document.getElementById('event-name-update').value
    performerUpdate=document.getElementById('event-performer-update').value
    dateUpdate=document.getElementById('event-date-update').value
    typeUpdate=document.getElementById('event-type-update').value
    locationUpdate=document.getElementById('event-location-update').value
    detailsUpdate=document.getElementById('event-details-update').value
    priceUpdate=document.getElementById('event-price-update').value
    urlUpdate=document.getElementById('event-image-update').value
    sectionsUpdate=document.getElementById('event-sections-update').value.split(",")
    sectionticketsUpdate=document.getElementById('event-sections-tickets-update').value.split(",")

    await updateDoc(docRef,{
        name:nameUpdate,
        performer:performerUpdate,
        date:dateUpdate,
        type:typeUpdate,
        location:locationUpdate,
        details: detailsUpdate,
        price:priceUpdate,
        url:urlUpdate,
        sections:sectionsUpdate,
        sectionstickets:sectionticketsUpdate
    })
    alert("Event updated")
    location.reload()
}
```

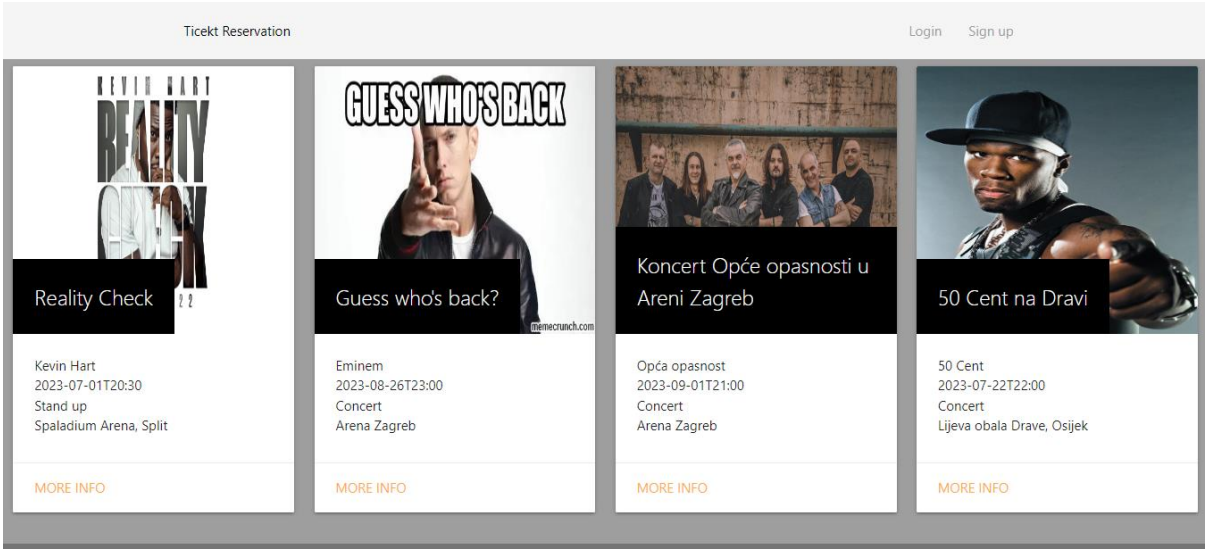
Slika 3.23. *JavaScript kod za funkciju eventUpdate.*

4. PRIKAZ RADA APLIKACIJE I USPOREDBA SLIČNOSTI I RAZLIKA S VEĆ POSTOJEĆIM WEB APLIKACIJAMA

U ovom poglavlju prikazan je izgled i funkcionalnost aplikacije te je uspoređena funkcionalnost aplikacije u usporedbi s web aplikacijama navedenim u drugom poglavlju.

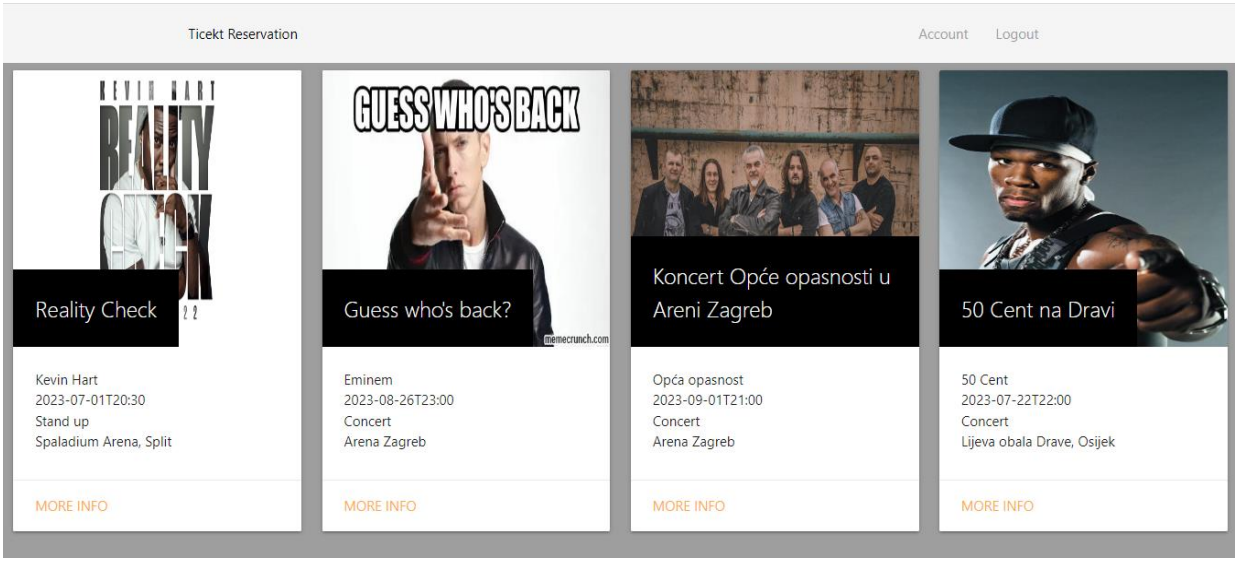
4.1. Prikaz rada aplikacije

Na početnoj stranici korisniku su vidljivi svi dostupni događaji o kojima pritiskom miša na gumb *MORE INFO* može vidjeti više informacija. Također pritiskom na gumb *Sing up* na navigacijskoj traci korisnik može napraviti račun na stranici te ako ima postojeći račun, pritiskom na gumb *Login,* može se prijaviti u sustav. Na slici 4.1. prikazana je početna stranica aplikacije kada korisnik nije prijavljen u sustav.



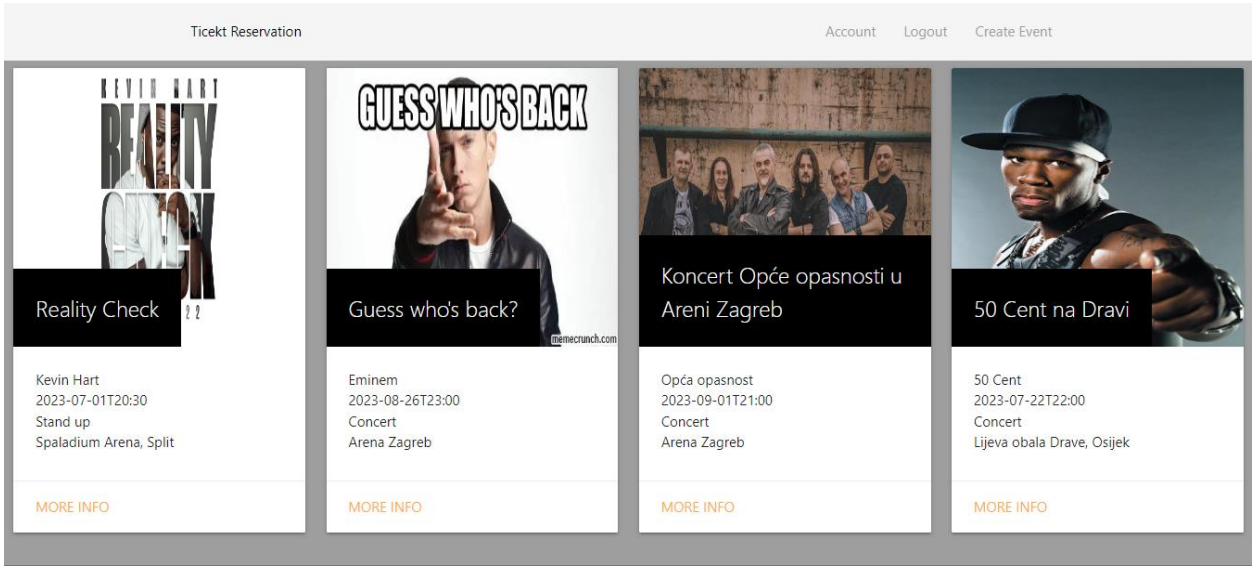
Slika 4.1. *Izgleda početne stranice kada korisnik nije prijavljen u sustav*

Kada je korisnik prijavljen, navigacijska traka mijenja izgled te korisnik dobiva nove opcije koje može pritisnuti na traci, *Account i Logout.* Slika 4.2. prikazuje izgled početne stranice kada je korisnik prijavljen u sustav kao kupac.



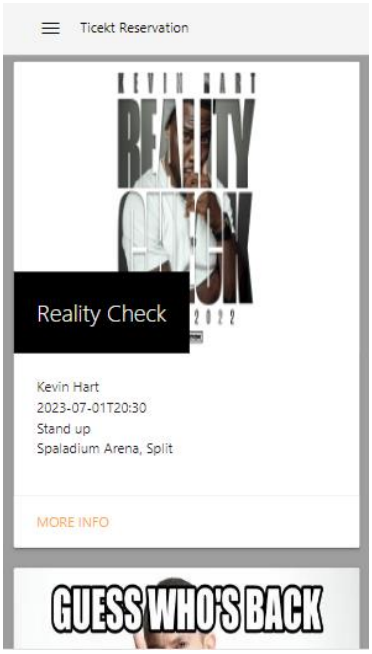
Slika 4.2. *Izgled stranice kada je korisnik prijavljen u sustav kao kupac*

Kada je korisnik prijavljen kao moderator na navigacijskoj traci pojavljuje mu se mogućnost izrade novog događaja pritiskom na gumb *Create Event.* Slika 4.3. prikazuje izgled početne stranice kada je korisnik prijavljen u sustav kao moderator.



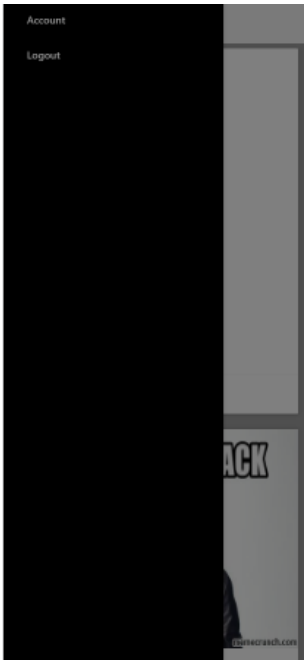
Slika 4.3. *Izgled početne stranice kada je korisnik prijavljen u sustav kao moderato*r

Funkcionalnost stranice jednaka je i na uređajima s manjim ekranima. Na slici 4.4. prikazan je izgled početne stranice na uređajima s manjim ekranima.



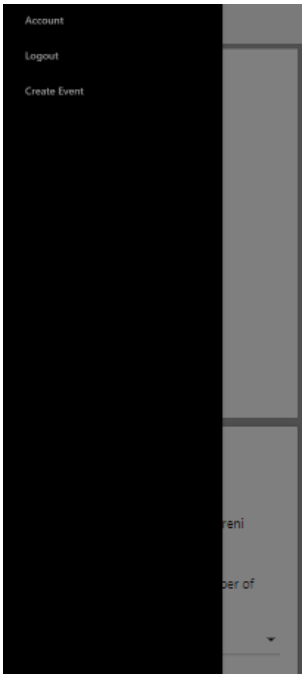
Slika 4.4. *Izgled početne stranice na uređajima s manjim ekranima*

Na slici 4.5. prikazan je izgled navigacijske trake na uređajima s manjim ekranima kada je korisnik kupac.



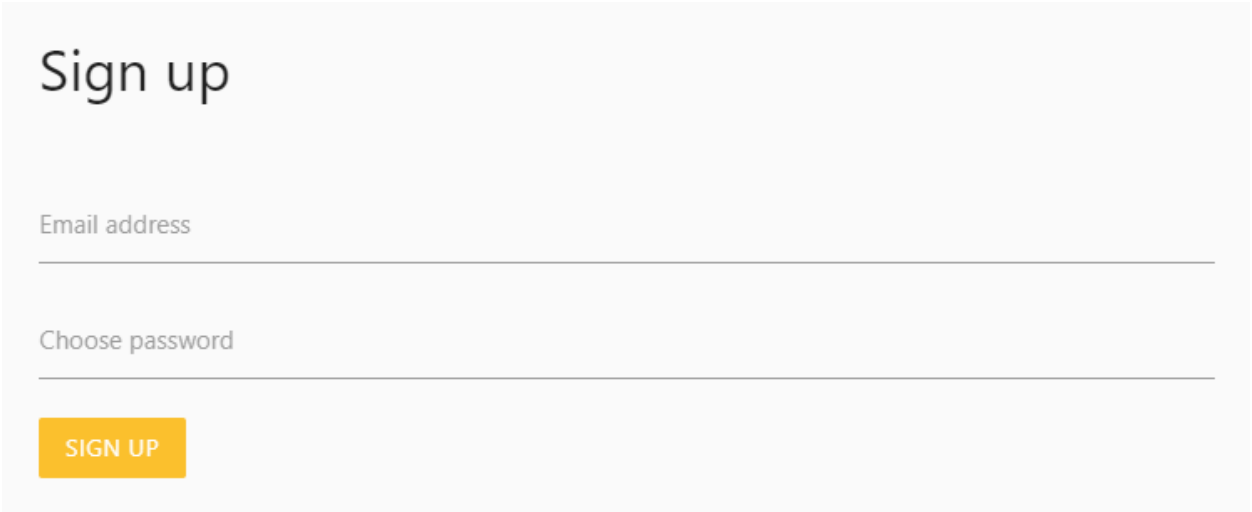
Slika 4.5. *Izgled navigacijske trake na uređajima s manjim ekranima kada je korisnik kupac*

Na slici 4.6. prikazan je izgled navigacijske trake na uređajima s manjim ekranima kada je korisnik moderator.



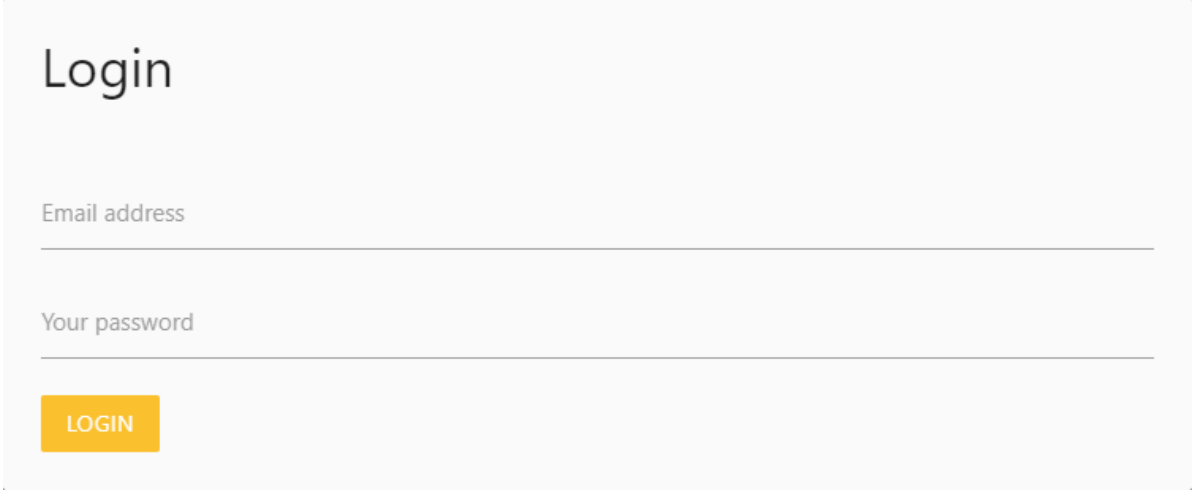
Slika 4.6. *Izgled navigacijske trake na uređajima s manjim ekranima kada je korisnik moderator*

Pritiskom na gumb *Sign up* na navigacijskoj traci otvara se modalni prozor za registraciju korisnika. Korisnik treba unijeti adresu elektroničke pošte i zaporku te pritisnuti gumb *SIGN UP* ispod obrasca kako bi napravio korisnički račun. Slika 4.7. prikazuje izgled modalnog prozora za registraciju korisnika



Slika 4.7. *Modalni prozor za registraciju korisnika*

Pritiskom na *Login* na navigacijskoj traci otvara se modalni prozor za prijavu korisnika u sustav. Korisnik treba unijeti adresu elektroničke pošte i zaporku te pritisnuti gumb *LOGIN* ispod obrasca kako bi bio se prijavio u sustav. Na slici 4.8. prikazan je izgled modalnog prozora za prijavu korisnika.

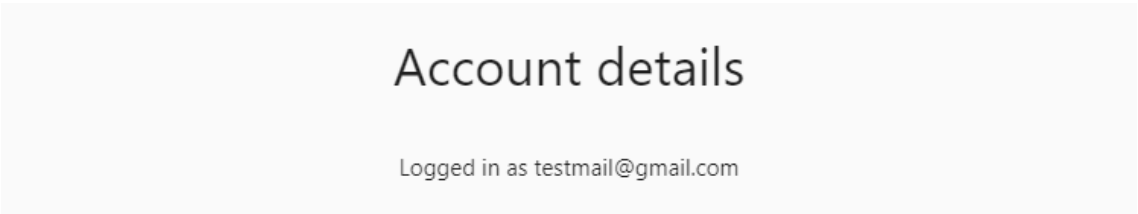


Slika 4.8. *Modalni prozor za prijavu korisnika u sustav*

Kada je korisnik prijavljen u sustav na navigacijskoj traci pojavljuju se dvije opcije, *Account* i *Logout* te ako je korisnik moderator pojavljuje se i *Create Event* .

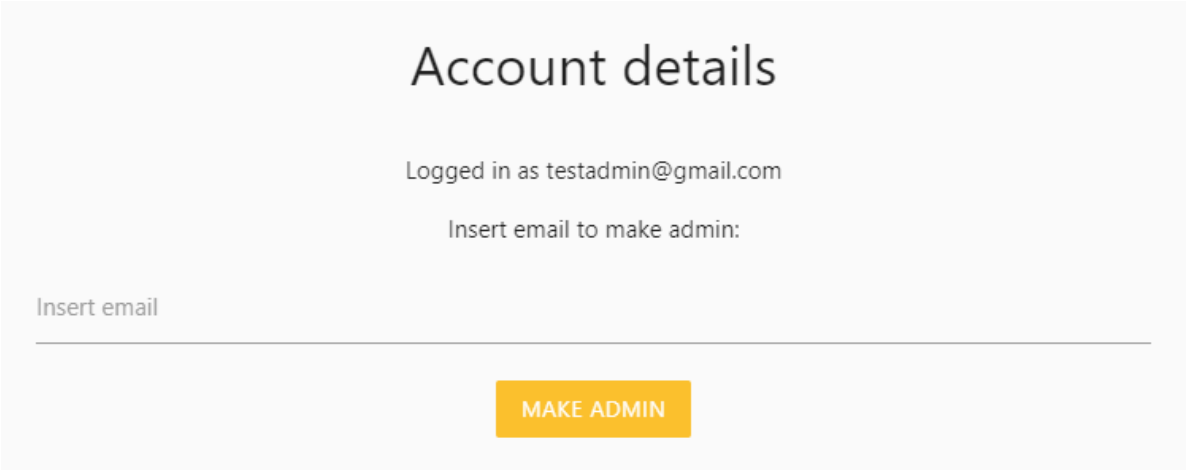
Pritiskom na gumb *Logout* na navigacijskoj traci, korisnik se odjavljuje iz sustava te se prikazuje stranica na kojoj se nalazi u obliku kada korisnik nije prijavljen u sustav.

Slika 4.9. prikazuje izgled modalnog prozora koji ispisuje informacije o korisničkom računu, odnosno s kojom je adresom elektroničke pošte korisnik prijavljen, koji se otvara pritiskom na gumb *Account* na navigacijskoj traci u slučaju kada je korisnik kupac.



Slika 4.9. *Izgled modalnog prozora kad je korisnik kupac*

Slika 4.10. prikazuje izgled modalnog prozora koji se otvara pritiskom na *Account* u slučaju kada je korisnik moderator.



Slika 4.10. *Izgled modalnog prozora kada je korisnik moderator*

Kada je korisnik moderator može unijeti adresu elektroničke pošte drugog korisnika te pritiskom na gumb *MAKE ADMIN* tog korisnika postaviti kao moderatora.

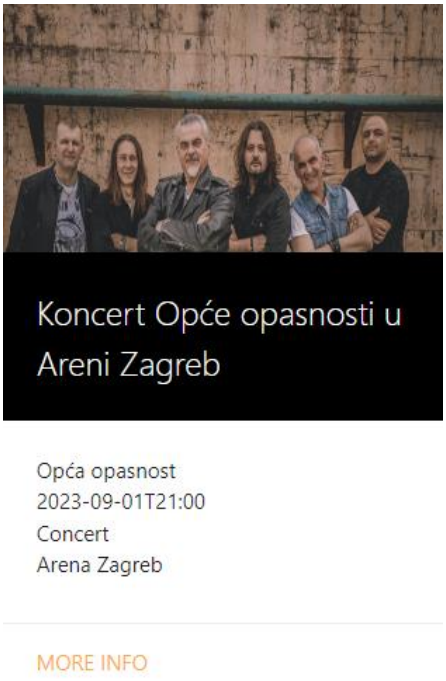
U slučaju da je korisnik moderator na navigacijskoj traci pojavit će se gumb *Create Event* koji kada je pritisnut otvara modalni prozor za stvaranje novog događaja. Slike 4.11. i 4.12. prikazuju izgled modalnog prozora za stvaranje novog događaja.

Slika 4.11. *Izgled modalnog prozora za stvaranje novog događaja*

Slika 4.12. *Izgled modalnog prozora za stvaranje novog događaja*

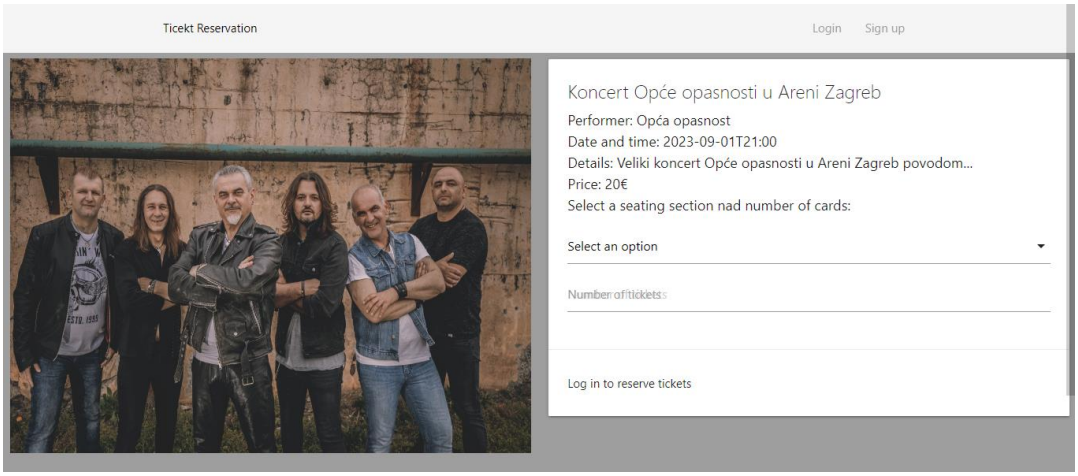
Moderator unosi potrebne informacije o događaju te pritiskom na gumb *CREATE* stvara novi događaj koji postaje vidljiv na početnoj stranici.

Pritiskom na gumb *MORE INFO* unutar kartice za pojedini događaj otvorit će se stranica sa dodatnim informacijama o tom događaju. Slika 4.13. prikazuje izgled kartice jednog događaja.



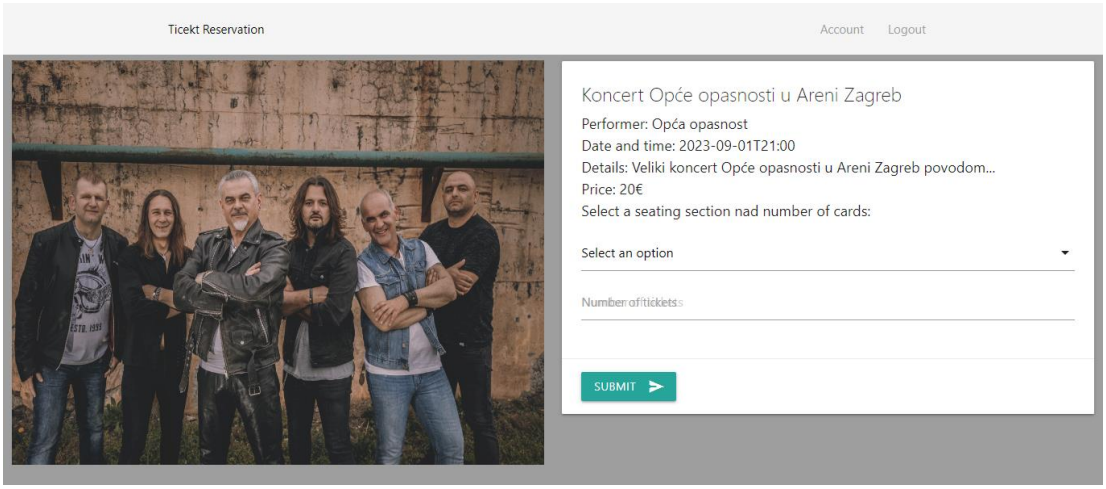
Slika 4.13. *Izgled kartice jednog događaja*

Stranica za prikaz više detalja o pojedinom događaju ima više izgleda ovisno o tome je li korisnik prijavljen ili nije te je li korisnik moderator ili je kupac. Slika 4.14. prikazuje izgled stranice *EventPage.html* kada korisnik nije prijavljen.



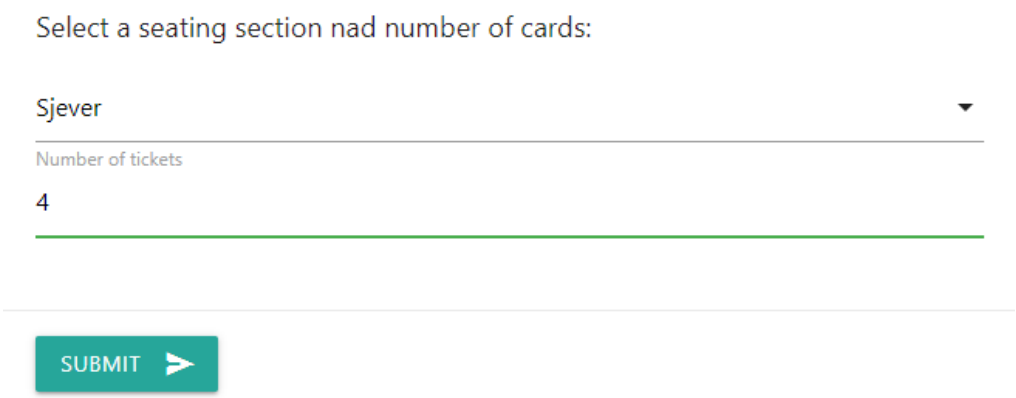
Slika 4.14. *Izgled stranice EventPage.html kada korisnik nije prijavljen*

Slika 4.15. izgled stranice *EventPage.html* kada je korisnik prijavljen kao kupac.



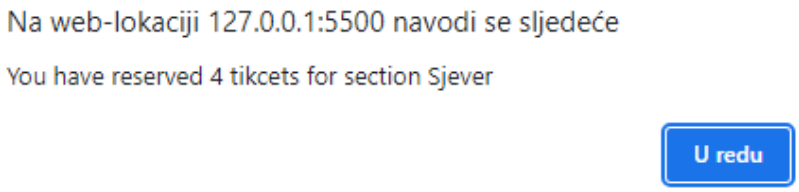
Slika 4.15. *Izgled stranice EventPage.html kada je korisnik prijavljen kao kupac*

Kada je korisnik prijavljen kao kupac može odabrati željenu opciju sjedenja i broj karata za tu opciju te pritiskom na gumb *SUBMIT* može rezervirati karte. Slika 4.16. prikazuje izgled popunjenog obrasca za rezervaciju karata.



Slika 4.16. *Popunjen obrazac za rezervaciju karata*

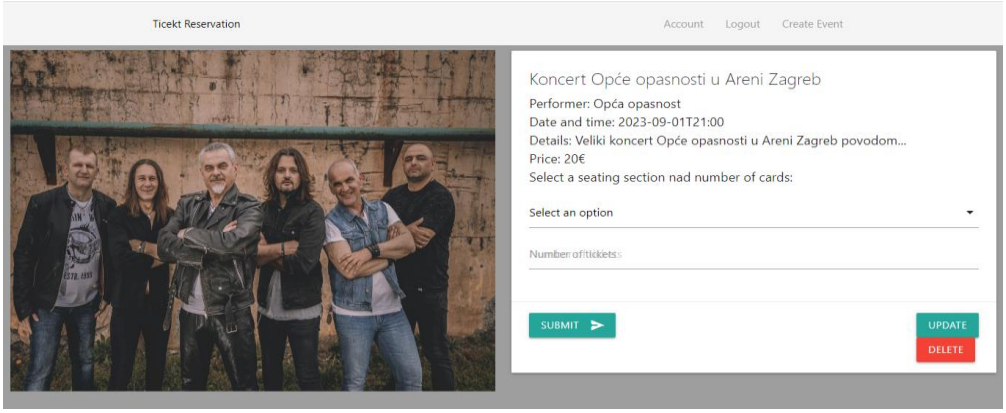
Nakon pritiska gumba *SUBMIT* u skočnom prozoru pojavit će se poruka o potvrdi rezervacije. Slika 4.17. prikazuje izgled skočnog prozora o potvrdi rezervacije karata.



Slika 4.17. *Izgled skočnog prozora o potvrdi rezervacije karata*

U slučaju da je korisnik pokušao rezervirati više karata nego li je trenutno dostupno u skočnom prozoru će se pokazati broj trenutno dostupnih karata za odabranu opciju sjedenja, a ako više nema dostupnih karata za odabrano mjesto sjedenja u skočnom prozoru će pisati kako su sve karte za taj sektor rezervirane.

Slika 4.18. prikazuje izgled stranice *EventPage.html* kada je korisnik prijavljen kao moderator.



Slika 4.18. *Izgled stranice EventPage.html kada je korisnik prijavljen kao moderator*

Kada je korisnik prijavljen kao moderator pritiskom na gumb *UPDATE* može ažurirati informacije o događaju na kojem se nalazi. Nakon pritiska tog gumba otvorit će se modalni prozor nalik modalnom prozoru za stvaranje događaja s već popunjenim vrijednostima za taj događaj u kojem moderator može napraviti potrebne izmjene. Nakon unošenja izmjena potrebno je pritisnuti gumb *UPDATE* kako bi se željene promjene sačuvale. Moderator također može obrisati događaj pritiskom na gumb *DELETE*. Nakon pritiskanja gumba na stranici se pojavljuje skočni prozor s potvrdom želi li moderator stvarno obrisati događaj. U slučaju potvrde, moderator se vraća na početnu stranicu, a obrisani događaj više neće biti vidljiv.

4.2. Usporedba s već postojećim aplikacijama

Kao što je ranije navedeno u drugom poglavlju ovog završnog rada, za rezervaciju i kupnju karata na internetu već postoje brojna rješenja. Neke od tih web aplikacija su one navedene u drugom poglavlju poput Ticketmastera, StubHuba, Eventima, Live Nationa i Viagogoa. Stranice su veoma intuitivne i lake za korištenje što i trebaju biti jer ih svakodnevno koristi velik broj ljudi. Optimizirane su za velik broj korisnika te njihovi poslužitelji trebaju moći podnijeti velik broj korisnika istovremeno. Pružaju lako pretraživanje željenih događaja po imenu izvođača, tipu događaja i ostalim parametrima kao i sortiranje tih događaja po datumu izvođenja i sličnim stavkama.

Najveća razlika između tih web aplikacija i web aplikacije izrađene u ovom završnom radu je što se gore navedene web aplikacije prvenstveno koriste za kupnju karata, a ne samo za rezervaciju. Te stranice nalaze se u vlasništvu organizacije koje zajedno sa organizatorima i izvođačima događaja direktno zarađuju na prodaji karata, dok je cilj ovo završnog rada bio omogućiti samo rezervaciju određenog broja karata. Također, kupci za neke događaje mogu izabrati točno mjesto sjedenja za svaku kartu, dok u aplikaciji u ovom završnom radu mogu odabrati samo sekciju sjedenja i broj karata. Nadalje, te stranice nude pretraživanje i filtriranje događaja po imenu, mjestu izvođenja, datumu, izvođaču, tipu događaja te ostalim parametrima. Pored toga, gore navedene web aplikacije, moraju moći podnijeti velik broj korisnika koji istovremeno pokušavaju rezervirati karte za jedan ili više događaja, dok web stranica u ovom završnom radu neće imati toliki broj korisnika.

Sličnosti između navedenih aplikacija i aplikacije ovog završnog rada je što nude karte za više vrsta događaja. Uz to, kako bi korisnik kupio ili rezervirao kartu potreban je biti prijavljen s važećim korisničkim računom kojeg je prethodno izradion na istoj stranici.

Web aplikacija u ovom završnom radu puno je jednostavnije nego web aplikacije navedene u drugom poglavlju te ne nudi opcije pretraga i sortiranja, ali korisniku nudi mogućnost odabira, pregleda i rezervacije karata za više vrsta događaja

5. ZAKLJUČAK

U ovom završnom radu prikazana je web aplikacija za rezervaciju karata na događajima kao i njena izrada. U samoj izradi korišteni su JavaScript, HTML, CSS, Materialize CSS, jQuery te je za potrebe baze podatak i autentikacije korišten Firebase. Korisniku je omogućena izrada računa i prijava u sustav kako bi mogao rezervirati karte za željeni događaj. Na početnoj stranici prikazani su svi dostupni događaji te odabirom jednog od događaja korisnik može vidjeti više informacije o svakom događaju. Ako je korisnik prijavljen kao kupac može rezervirati željen broj karata za odabranu kategoriju. Ako je odabrani broj karata dostupan, može rezervirati karte. U slučaju da željeni broj karata nije dostupan, kupac će dobiti poruku o preostalom broju karata za tu vrstu karata. Kada je korisnik prijavljen kao moderator uz opciju za rezerviranje karata dobiva mogućnost izrade novog događaja što podrazumijeva unošenje podatka poput naziva događaja, izvođača, kategorije, broja karata te ostalih informacija važnih za događaj. Uz to ima mogućnost izmjene tih informacija kao i brisanja događaja te mogućnost da korisnika pretvori iz kupca u moderatora.

LITERATURA

[1] Ticketmaseter, dostupno na: https://www.ticketmaster.co.uk/ [13.6.2023]

[2] StubHub, dostupno na: https://www.stubhub.ie/?valueMessaging=true&utm_medium=cpc&utm_source=google&utm_campaign=IE-EN-%5BS%5BBRA_0%5D_G0%3AStubhub&gclid=CjwKCAjwp6CkBhB_EiwAlQVyxXxuViF5WF13NjWaRia-q61i-zBbJR0dXLUqffSaFigQRpL2S3CMvxoC-t4QAvD_BwE, [13.6.2023.]

[3] Eventim, dostupno na: https://www.eventim.hr/hr/, [13.6.2023.]

[4] Live Nation, dostupno na: https://www.livenation.com/, [13.6.2023.]

[5] Viagogo, dostupno na: https://www.viagogo.com/, [13.6.2023.]

[6] Materealize CSS, dostupno na: https://materializecss.com/, [14.6.2023.]

[7] R. Wieruch, The Road to React with Firebase, Leanpub, travanj 2019.

[8] Add Firebase to your JavaScript project, dostupno na: https://firebase.google.com/docs/web/setup , [14.6.2023.]

[9] R. Mohanan, What Is an SDK (Software Development Kit)? Working, Types and Importance, 8.12.2022., dostupno na: https://www.spiceworks.com/tech/devops/articles/what-is-sdk/ [15.8.2023.]

[10] J. Chaffer, K. Swedberg, Learning jQuery, Fourth Edition, Packt Publishing, 2013.

[11] Dr. A. Rauschmayer, Exploring ES6, Upgrade to the next version of JavaScript, Leanpub, 2015.

[12] Cloud Firestore , dostupno na: https://firebase.google.com/docs/firestores , [15.8.2023.]

[13] M. Haverbeke, Eloquent JavaScript, 3rd edition, No Starch Press, 2018.

[14] Template literals, dostupno na: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Template_literals , [15.8.2023.]

SAŽETAK

Za potrebe ovog završnog rada izrađena je web aplikacije za rezervaciju karata na događajima. Prilikom izrade web aplikacije korištene su tehnologije poput JavaScripta, HTML-a, CSS-a, Materialize CSS-a, jQuerya i Firebasea. Aplikacija omogućuje korisniku pregled dostupnih događaja te rezerviranje karata ako je korisnik kupac. U slučaju da je korisnik moderator, omogućeno mu je kreiranje novog događaja, izmjena i brisanje postojećih događaja te postavljanje drugog korisnika kao moderatora.

Ključne riječi: karte, korisnik, kupac, moderator, rezervacija, web aplikacija

ABSTRACT

Web application for booking tickets at events

The development of a web application for ticket reservation at events for this final thesis involved the use of technologies including JavaScript, HTML, CSS, Materialize CSS, jQuery and Firebase. The application offers users the ability to browse through available events and reserve tickets if the user is a customer. If the user is a moderator, he can create a new event, edit and delete existing events, and set another user as moderator.

Keywords: customer, moderator, reservation, tickets, user, web application

ŽIVOTOPIS

Andrej Horvat rođen je 6.12.2000. u Požegi, Hrvatska. Pohađao je Osnovnu školu Julije Kempfa u Požegi. Nakon završene osnovne škole upisao se u Gimnaziju Požega, smjer matematička gimnazija. Nakon završene srednje škole 2019. godine upisuje se na Fakultet elektrotehnike, računarstva, i informacijskih tehnologija u Osijeku, te trenutno pohađa preddiplomski sveučilišni studij računarstva, smjer računalno inženjerstvo.

_____

Potpis autora

PRILOZI

Prilog 1. Završni rad u formatu .docx

Prilog 2, Završni rad u formatu .pdf

Prilog 3. Programsko rješenje izrađene web aplikacije