Univerzitet u Beogradu  
Fakultet organizacionih nauka  
Katedra za elektronsko poslovanje

**Web aplikacija za prodaju ulaznica**

Seminarski rad iz predmeta Internet tehnologije

Mentor: Studenti:

Milica Simić Aleksa Božanić 0286/2020

Marko Mišović 0041/2020

Beograd, 2024.

Sadržaj

[1. Korisnički zahtev 1](#_Toc158247753)

[2. Opis sistema 2](#_Toc158247754)

[1. Opis slučajeva korišćenja 2](#_Toc158247755)

[1.1. SK1 - Dodavanje dogadjaja 2](#_Toc158247756)

[1.2. SK2 - Brisanje dogadjaja 4](#_Toc158247757)

[1.1.3 SK3 – Azuriranje dogadjaja 5](#_Toc158247758)

[2. Opis arhitekture aplikacije 6](#_Toc158247759)

[3. Opis procesa slučajeva korišćenja (dijagrami sekvenci) 7](#_Toc158247760)

[3.1. DS1: Dijagram sekvenci – Kreiranje dogadjaja 7](#_Toc158247761)

[3.2 DS2: Dijagram sekvenci – Brisanje dogadjaja 8](#_Toc158247762)

[3.3 DS3 – Ažuriranje događaja 9](#_Toc158247763)

[4. Model podataka 10](#_Toc158247764)

[5. Detaljna specifikacija REST API-ja 11](#_Toc158247765)

[3. Kratak opis tehnologija korišćenih u aplikaciji 19](#_Toc158247766)

[4. Korisničko uputstvo 20](#_Toc158247767)

[5. Prikaz reprezentativnih delova koda 28](#_Toc158247768)

[6. GitHub link ka repozitorijumu 40](#_Toc158247769)

[7. Reference 41](#_Toc158247770)

# Korisnički zahtev

Potrebno je razviti web aplikaciju namenjenu kupovini ulaznica za različite događaje. Aplikacija treba da uključuje funkcionalnost registracije i prijave za korisnike. Jednom kada se korisnik uspešno prijavi, trebalo bi da mu bude dostupan pregled aktuelnih događaja iz baze podataka. Kroz interaktivni interfejs, korisnici će imati priliku da odaberu događaj po želji i kupe karte. Važno je napomenuti da za svaki događaj postoji više kategorija ulaznica, kao što su Standard, Premium, VIP, Student i Grupna, pri čemu svaka kategorija ima različitu cenu.

Da bi se korisnicima olakšalo pronalaženje događaja koji odgovaraju njihovim interesovanjima, aplikacija će sadržati opcije za pretragu, filtriranje i paginaciju događaja. Onima koji su ulogovani biće omogućeno ne samo pregledanje istorije kupljenih karata, već i opcija za njihovo štampanje.

Za administratore, planira se kreiranje posebne admin stranice koja će im omogućiti da upravljaju događajima - od kreiranja novih, preko brisanja, do izmene postojećih događaja. Na admin stranici biće dostupne i dodatne funkcionalnosti za prikaz detalja o događajima kao i različite statistike, omogućavajući time administratorima bolji uvid i lakše upravljanje sadržajem na platformi.

# Opis sistema

## Opis slučajeva korišćenja

U ovoj aplikaciji, identifikovano je više slučajeva korišćenja, od kojih ćemo izdvojiti nekoliko:

1. Dodavanje dogadjaja
2. Brisanje dogadjaja
3. Azuriranje dogadjaja

### 1.1. SK1 - Dodavanje dogadjaja

Opis: Ovaj slučaj korišćenja opisuje postupak dodavanja novog dogadjaja u bazu podataka putem sistema od strane administratora.

Akteri:

* Administrator
* Sistem (program)

Učesnici:

* Administrator
* Sistem (program)

Preduslov:

* Sistem je uključen i administrator ima pristup funkcionalnosti za dodavanje novog dogadjaja u bazu.

Osnovni scenario:

1. Administrator poziva sistem da otvori formu za dodavanje novog dogadjaja u bazu.
2. Sistem prikazuje formu za unos podataka o dogadjaju.
3. Administrator unosi podatke o dogadjaju u formu.
4. Administrator proverava da li je uneo ispravne podatke o dogadjaju.
5. Administrator poziva sistem da sačuva podatke o dogadjaju u bazu.
6. Sistem proverava da li su uneti podaci ispravni i validni.
7. Sistem dodaje novi dogadjaj u bazu i prikazuje administratoru poruku: "Sistem je uspešno dodao novi dogadjaj u bazu".

Alternativni scenario:

4.1 Ukoliko administrator nije uneo ispravne podatke o dogadjaju, sistem prikazuje mu poruku o grešci i vraća ga na korak 2

6.1 Ukoliko sistem utvrdi da su podaci neispravni, prikazuje se administratoru poruka o grešci i vraća ga na korak 2.

7.1 Ukoliko sistem ne može da sačuva podatke o dogadjaju u bazu, prikazuje se administratoru poruka o grešci i vraća ga na korak 2

### 1.2. SK2 - Brisanje dogadjaja

Slučaj korišćenja – Brisanje događaja

Opis: Ovaj slučaj korišćenja opisuje postupak brisanja postojećeg događaja iz sistema od strane administratora ili korisnika.

Akteri:

* Administrator
* Sistem (program)

Učesnici:

* Administrator
* Sistem (program)

Preduslov:

* Sistem je uključen, a administrator ima pristup funkcionalnosti za brisanje događaja iz sistema.

Osnovni scenario:

1. Administrator poziva sistem da otvori listu događaja za pregled ili brisanje.
2. Sistem prikazuje listu postojećih događaja.
3. Administrator bira događaj koji želi da obriše iz liste.
4. Administrator potvrđuje brisanje odabranog događaja.
5. Sistem proverava da li je odabrani događaj moguće obrisati
6. Sistem briše događaj iz sistema i prikazuje poruku korisniku: "Događaj je uspešno obrisan".

Alternativni scenario:

5.1 Ukoliko sistem utvrdi da odabrani događaj nije moguće obrisati (na primer, zbog već postojećih veza ili drugih ograničenja), prikazuje se korisniku poruka o grešci.

### 1.1.3 SK3 – Azuriranje dogadjaja

Slučaj korišćenja – Ažuriranje događaja

Opis: Ovaj slučaj korišćenja opisuje postupak ažuriranja postojećeg događaja u sistemu od strane administratora.

Akteri:

* Administrator
* Sistem (program)

Učesnici:

* Administrator
* Sistem (program)

Preduslov:

* Sistem je uključen, a admin ima pristup funkcionalnosti za ažuriranje događaja u sistemu.

Osnovni scenario:

1. Administrator poziva sistem da otvori listu događaja za pregled i ažuriranje.
2. Sistem prikazuje listu postojećih događaja.
3. Administrator bira događaj koji želi da ažurira iz liste.
4. Sistem prikazuje formu sa trenutnim podacima o događaju i omogućava administratoru ili korisniku da unese izmene.
5. Administrator unosi željene izmene u podacima o događaju.
6. Administrator proverava unesene izmene i poziva sistem da sačuva ažurirane podatke o događaju.
7. Sistem proverava da li su uneseni podaci ispravni i validni.
8. Sistem ažurira podatke o događaju u sistemu i prikazuje poruku korisniku: "Podaci o događaju su uspešno ažurirani

Alternativni scenario:

7.1 Ukoliko sistem utvrdi da su uneseni podaci neispravni ili nevalidni, prikazuje se korisniku poruka o grešci i vraća ga na korak 4.

## Opis arhitekture aplikacije

U ovom poglavlju opisaćemo arhitekturu aplikacije koja je razvijena korišćenjem React i Laravel tehnologija. Arhitektura aplikacije igra ključnu ulogu u organizaciji koda, komunikaciji između različitih delova sistema i obezbeđivanju performansi i skalabilnosti aplikacije.

Aplikacija se sastoji od dva glavna dela: frontend (React) i backend (Laravel). Ovaj razvojni stack omogućava razdvajanje odgovornosti između korisničkog interfejsa i poslovne logike, što olakšava održavanje i skaliranje.

Frontend deo aplikacije je implementiran koristeći React, popularni JavaScript framework za izradu korisničkog interfejsa. Evo ključnih komponenti arhitekture React dela:

* Aplikacija je organizovana u više komponenti, od kojih svaka ima svoj specifičan zadatak i odgovornost. To olakšava ponovno korišćenje, testiranje i održavanje koda.
* React Router se koristi za upravljanje rutama i navigacijom unutar aplikacije. Omogućava prikazivanje različitih stranica i komponenata u zavisnosti od URL adrese.
* Za komunikaciju sa backendom koristi se Axios biblioteka za slanje HTTP zahteva prema API-ju servera.
* React omogućava upravljanje stanjem aplikacije korišćenjem stanja i efekata. Ovo omogućava praćenje promena u aplikaciji i ažuriranje korisničkog interfejsa u skladu s tim.

Backend deo aplikacije se bazira na Laravel, PHP frameworku za izradu web aplikacija. Evo ključnih komponenti arhitekture Laravel dela:

* Laravel se koristi za izgradnju API-ja koji omogućava komunikaciju između frontend i backend dela aplikacije. API se često koristi za slanje i primanje JSON podataka.
* Laravel kontroleri se koriste za upravljanje zahtevima, obradu poslovne logike i vraćanje odgovora frontendu.
* Koristi se za definisanje ruta i mapiranje HTTP zahteva na odgovarajuće kontrolere i akcije.
* Laravelov ORM (Object-Relational Mapping) omogućava interakciju sa bazom podataka putem objekata i modela, što olakšava upravljanje podacima.

## Opis procesa slučajeva korišćenja (dijagrami sekvenci)

### DS1: Dijagram sekvenci – Kreiranje dogadjaja

Slika na kojoj se nalazi tekst, snimak ekrana, broj, Font

Opis je automatski generisan

Slika 1 dijagram sekvenci 1

### 3.2 DS2: Dijagram sekvenci – Brisanje dogadjaja

Slika na kojoj se nalazi tekst, snimak ekrana, Font, broj

Opis je automatski generisan

Slika 2 dijagram sekvenci 2

### DS3 – Ažuriranje događaja

Slika na kojoj se nalazi tekst, snimak ekrana, Font, broj

Opis je automatski generisan

Slika 3 dijagram sekvenci 3

## 4. Model podataka

Slika na kojoj se nalazi tekst, snimak ekrana, dijagram, Font

Opis je automatski generisan

Slika 4 dijagram klasa

## Detaljna specifikacija REST API-ja

|  |  |
| --- | --- |
| Opis funkcije | Prikaz svih dogadjaja |
| HTTP metoda | GET |
| URL | /api/dogadjaji |
| URL parametri | ?page=1 |
| HTTP body parametri | nema |
| Format HTTP body parametara | JSON |
| Izlazni parametri | "data": [          {              "id": 1,              "naziv": "New York Marathon",              "datumVreme": "2023-11-05 09:00:00",              "mesto": "New York, USA",              "tip": {                  "id": 3,                  "naziv": "Sportski događaj"              },              "organizator": "NYRR",              "slika": "images/ny-marathon.jpg",              "kapacitet": 50000          },          {              "id": 2,              "naziv": "Comic-Con International",              "datumVreme": "2023-07-20 10:00:00",              "mesto": "San Diego, USA",              "tip": {                  "id": 4,                  "naziv": "Konferencija"              },              "organizator": "San Diego Comic Convention",              "slika": "images/comic-con.jpg",              "kapacitet": 130000          },          {              "id": 3,              "naziv": "Glastonbury Festival",              "datumVreme": "2023-06-21 12:00:00",              "mesto": "Pilton, England",              "tip": {                  "id": 5,                  "naziv": "Festival"              },              "organizator": "Glastonbury Festivals Ltd.",              "slika": "images/glastonbury-festival.jpg",              "kapacitet": 135000          }, |
| Format izlaznih parametara | Application/json |
| Opis funkcije | Kreiranje dogadjaja |
| HTTP metoda | POST |
| URL | /api/dogadjaji |
| URL parametri | naziv=DOG A&datumVreme=2023-11-05 09:00:00&mesto=bbbb&tip=1&organizator=cccc&kapacitet=10 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "message": "Uspesno kreiran dogadjaj",      "dogadjaj": {          "id": 7,          "naziv": "DOG A",          "datumVreme": "2023-11-05 09:00:00",          "mesto": "bbbb",          "tip": {              "id": 1,              "naziv": "AAA"          },          "organizator": "cccc",          "slika": **null**,          "kapacitet": "10"      }  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Ažuriranje dogadjaja |
| HTTP metoda | PUT |
| URL | /api/dogadjaji |
| URL parametri | 7?naziv=DOGADJAJ 2&datumVreme=2023-11-05 09:00:00&mesto=bbbb&tip=1&organizator=cccc&kapacitet=10 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "message": "Uspesno azuriran dogadjaj",      "dogadjaj": {          "id": 7,          "naziv": "DOGADJAJ 2",          "datumVreme": "2023-11-05 09:00:00",          "mesto": "bbbb",          "tip": {              "id": 1,              "naziv": "AAA"          },          "organizator": "cccc",          "slika": **null**,          "kapacitet": "10"      }  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Brisanje dogadjaja |
| HTTP metoda | DELETE |
| URL | /api/dogadjaji |
| URL parametri | 7 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "message": "USPEH"  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Prikaz svih ulaznica |
| HTTP metoda | GET |
| URL | /api/ulaznice |
| URL parametri | nema |
| HTTP body parametri | nema |
| Format HTTP body parametara | JSON |
| Izlazni parametri | {      "data": [          {              "id": 1,              "dogadjaj": {                  "id": 1,                  "naziv": "New York Marathon",                  "datumVreme": "2023-11-05 09:00:00",                  "mesto": "New York, USA",                  "tip": {                      "id": 3,                      "naziv": "Sportski događaj"                  },                  "organizator": "NYRR",                  "slika": "images/ny-marathon.jpg",                  "kapacitet": 50000              },              "korisnik": {                  "name": "pera",                  "email": "pera@gmail.com",                  "kontakt": "061254781",                  "uloga": "korisnik"              },              "tip": {                  "id": 1,                  "naziv": "Standard"              },              "datumKupovine": "2023-12-12 15:14:56",              "cena": "456.00"          },          {              "id": 3,              "dogadjaj": {                  "id": 1,                  "naziv": "New York Marathon",                  "datumVreme": "2023-11-05 09:00:00",                  "mesto": "New York, USA",                  "tip": {                      "id": 3,                      "naziv": "Sportski događaj"                  },                  "organizator": "NYRR",                  "slika": "images/ny-marathon.jpg",                  "kapacitet": 50000              },              "korisnik": {                  "name": "pera",                  "email": "pera@gmail.com",                  "kontakt": "061254781",                  "uloga": "korisnik"              },              "tip": {                  "id": 1,                  "naziv": "Standard"              },              "datumKupovine": "2023-11-05 09:00:00",              "cena": "850.00"          }      ]  } |
| Format izlaznih parametara | Application/json |
| Opis funkcije | Kreiranje ulaznice |
| HTTP metoda | POST |
| URL | /api/ulaznice |
| URL parametri | ?tip=1&dogadjaj=1&korisnik=1&datumKupovine=2023-11-05 09:00:00&cena=850 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "message": "Ulaznica uspešno kreirana",      "ulaznica": {          "id": 4,          "dogadjaj": {              "id": 1,              "naziv": "New York Marathon",              "datumVreme": "2023-11-05 09:00:00",              "mesto": "New York, USA",              "tip": {                  "id": 3,                  "naziv": "Sportski događaj"              },              "organizator": "NYRR",              "slika": "images/ny-marathon.jpg",              "kapacitet": 50000          },          "korisnik": {              "name": "pera",              "email": "pera@gmail.com",              "kontakt": "061254781",              "uloga": "korisnik"          },          "tip": {              "id": 1,              "naziv": "Standard"          },          "datumKupovine": "2023-11-05 09:00:00",          "cena": "850"      }  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Ažuriranje ulaznice |
| HTTP metoda | PUT |
| URL | /api/ulaznice |
| URL parametri | 4? tip=1&dogadjaj=1&korisnik=1&datumKupovine=2023-11-05 09:00:00&cena=850 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "message": "Ulaznica uspešno ažurirana"  } |
| Format izlaznih parametara | application/json |
| Opis funkcije | Brisanje ulaznice |
| HTTP metoda | DELETE |
| URL | /api/dogadjaji |
| URL parametri | 4 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "message": "Ulaznica uspešno obrisana"  } |
| Format izlaznih parametara | application/json |
| Opis funkcije | Registracija korisnika |
| HTTP metoda | POST |
| URL | /api/register |
| URL parametri | password=perapera&name=peraa&email=pera1@gmail.com&kontakt=0612974473 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "message": "Korisnik uspešno registrovan",      "user": {          "name": "peraa",          "email": "pera1@gmail.com",          "kontakt": "0612974473",          "updated\_at": "2023-12-21T22:43:26.000000Z",          "created\_at": "2023-12-21T22:43:26.000000Z",          "id": 5      },      "token": "5|3gbLBWWXgjyuZT1pYdM6MIgOEZjlA7LgbDQ5V6ZA"  } |
| Format izlaznih parametara | application/json |
| Opis funkcije | Prijava korisnika |
| HTTP metoda | POST |
| URL | /api/login |
| URL parametri | ?password=perapera&email=pera1@gmail.com |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "message": "Korisnik uspešno prijavljen",      "user": {          "id": 5,          "name": "peraa",          "email": "pera1@gmail.com",          "email\_verified\_at": **null**,          "created\_at": "2023-12-21T22:43:26.000000Z",          "updated\_at": "2023-12-21T22:43:26.000000Z",          "kontakt": "0612974473",          "uloga": "korisnik"      },      "token": "6|rTQ1KMWoJMnbd4iqfCCnbhNQv1pbAZ0svX3UzhxD"  } |
| Format izlaznih parametara | application/json |
| Opis funkcije | Odjava korisnika |
| HTTP metoda | POST |
| URL | /api/logout |
| URL parametri | ?password=perapera&email=pera1@gmail.com |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "message": "Korisnik uspešno odjavljen"  } |
| Format izlaznih parametara | application/json |

# Kratak opis tehnologija korišćenih u aplikaciji

JavaScript (JS) je široko prihvaćen programski jezik namenjen razvoju interaktivnih web stranica i aplikacija. Izvodi se na klijentskoj strani, što znači da se kod procesuira direktno u web pregledaču korisnika, umesto na web serveru. Ova karakteristika omogućava JavaScriptu da unese dinamičke i interaktivne elemente u web stranice, poput animacija i automatskog osvežavanja sadržaja.

React predstavlja jedan od vodećih JavaScript okvira za kreiranje korisničkih interfejsa (UI) za web, koristeći JSX - sintaksnu ekstenziju koja omogućava inkorporiranje HTML i CSS koda unutar JavaScript datoteka. React se često koristi za razvoj jednostraničnih aplikacija (SPA) i raznih dinamičkih UI komponenti, uključujući liste, dijagrame i interaktivne kontrolne elemente.

PHP je programski jezik orijentisan ka serveru, koji se masovno koristi za razvoj web aplikacija. Zahvaljujući svojoj širokoj dostupnosti i bogatom setu ugrađenih funkcionalnosti, PHP olakšava izradu kompleksnih web aplikacija. Izvršava se na serveru, omogućavajući procesiranje podataka pre slanja klijentima, i koristi se za razne zadatke poput obrade formi, kreiranja sesija i slanja elektronske pošte.

Laravel je visoko cenjen PHP okvir koji nudi raznovrsne ugrađene funkcije i alate za efikasan i pojednostavljen proces razvoja web aplikacija. Laravel olakšava rad sa bazama podataka, autentifikaciju korisnika, rutiranje, kao i slanje e-pošte, među ostalim funkcionalnostima.

CSS (Cascading Style Sheets) je jezik za stilizovanje koji definiše vizuelni izgled i formatiranje web stranica. Koristi se za određivanje boja, fontova, rasporeda i ostalih vizuelnih aspekata stranica, omogućavajući razvojnim timovima da kreiraju estetski privlačne i konzistentne korisničke interfejse.

HTML (Hypertext Markup Language) je temeljni jezik za struktuiranje i prezentaciju sadržaja na webu. Služi za definisanje osnovne strukture stranice, uključujući njene segmente poput zaglavlja, tela i podnožja, kao i za umetanje teksta, slika, obrazaca i drugih elemenata.

Axios je priznata JavaScript biblioteka namenjena za upravljanje HTTP zahtevima između klijenta i servera, često korišćena u kombinaciji sa React aplikacijama za komunikaciju s API-ima i obradu podataka. Nudi bogat skup funkcionalnosti, uključujući automatsko konvertovanje odgovora u JSON format, konfiguraciju zahteva, obradu grešaka i podršku za moderne asinhrone operacije poput Promise i async/await.

# Korisničko uputstvo

Prilikom pokretanja aplikacije otvara se login forma koja je prikazana na slici ispod

Slika na kojoj se nalazi tekst, snimak ekrana, Četvorougao, dizajn

Opis je automatski generisan

Iz navigacionog menija, korisnici takodje mogu da odaberu i opciju da se registruju čime se otvara komponenta na kojoj je prikazana forma za registraciju. Ovo je prikazano na slici ispod. Slika na kojoj se nalazi tekst, snimak ekrana, dizajn

Opis je automatski generisan

Nakon što se admin uspešno prijavi otvara mu se stranica na kojoj može da dodaje nove događaje ili da briše postojeće, da ih ažurira, kao i da pregleda statistike.

Slika na kojoj se nalazi tekst, snimak ekrana, softver, Font

Opis je automatski generisan

Klikom na dugme dodaj otvara se forma za unos novog dogadjaja u bazu podataka

Slika na kojoj se nalazi tekst, snimak ekrana, broj, softver

Opis je automatski generisan

Klikom na dugme ažuriraj otvara se forma za ažuriranje podataka o događaju

Slika na kojoj se nalazi tekst, snimak ekrana, Font, broj

Opis je automatski generisan

Klikom na dugme detalji otvara se stranica preko koje admin može da vidi detalje o određenom događaju

Slika na kojoj se nalazi tekst, snimak ekrana, krug, dizajn

Opis je automatski generisan

Ukoliko korisnik odabere opciju Dogadjaji iz navigacionog menija otvara mu se stranica na kojoj su prikazani događaji učitani sa spoljnog apija. Na ovoj stranici je implementirana paginacija, filteri i pretraga događaja.

Slika na kojoj se nalazi tekst, snimak ekrana, softver, Multimedijski softver

Opis je automatski generisan

Kada se uloguje korisnik, njemu se otvara stranica za kupovinu ulaznice

Slika na kojoj se nalazi tekst, snimak ekrana, Vebsajt, Brošura

Opis je automatski generisan

Klikom na dugme kupi ulaznice, korisniku se otvara modal u kom može da unese podatke vezane za kupovinu

Slika na kojoj se nalazi tekst, snimak ekrana, broj, ekran

Opis je automatski generisan

Na osnovu tipa ulaznice I kolicine, se formira I cena. Korisnik može da vidi sve svoje kupljene ulaznice na stranici Moje ulaznice

Slika na kojoj se nalazi tekst, snimak ekrana, Font, broj

Opis je automatski generisan

Klikom na opciju Stampaj korisniku se preuzima pdf fajl kao potvrda o kupovini ulaznice

Slika na kojoj se nalazi tekst, snimak ekrana, ekran, Font

Opis je automatski generisan

# Prikaz reprezentativnih delova koda

Sledeće dve komponente, Login i Register, predstavljaju ključne elemente u upravljanju korisničkim pristupom u React aplikaciji. Komponenta Login omogućava korisnicima da se prijave, koristeći email i lozinku, a autentifikacija se vrši putem zahteva prema API-ju. Nakon uspešne prijave, korisnik se preusmerava na admin stranicu. Komponenta Register omogućava novim korisnicima da kreiraju nalog, zahtevajući osnovne informacije kao što su ime, email, lozinka i kontakt. Po uspešnoj registraciji, korisnik se preusmerava na stranicu za prijavu. Obe komponente koriste asinhronu komunikaciju sa serverom za obradu podataka.

import React, { useState } from 'react';

import axios from 'axios';

import { useNavigate } from 'react-router-dom';

import './Register.css';

const Register = () => {

    const [formData, setFormData] = useState({

        name: '',

        email: '',

        password: '',

        kontakt: ''

    });

    const navigate = useNavigate();

    const handleChange = (e) => {

        setFormData({ ...formData, [e.target.name]: e.target.value });

    };

    const handleSubmit = async (e) => {

        e.preventDefault();

        try {

            await axios.post('http://127.0.0.1:8000/api/register', formData);

            navigate('/login');

        } catch (error) {

            console.error('Greška prilikom registracije', error.response.data);

        }

    };

    return (

        <div className="register-container">

            <form className="register-form" onSubmit={handleSubmit}>

                <h2>Registruj se</h2>

                <input type="text" name="name" placeholder="Ime" value={formData.name} onChange={handleChange} required />

                <input type="email" name="email" placeholder="Email" value={formData.email} onChange={handleChange} required />

                <input type="password" name="password" placeholder="Lozinka" value={formData.password} onChange={handleChange} required />

                <input type="text" name="kontakt" placeholder="Kontakt" value={formData.kontakt} onChange={handleChange} required />

                <button type="submit" className="register-btn">Registruj se</button>

            </form>

        </div>

    );

};

export default Register;

import React, { useState } from 'react';

import axios from 'axios';

import { useNavigate } from 'react-router-dom';

import './Login.css';

const Login = ({setToken}) => {

    const [credentials, setCredentials] = useState({

        email: '',

        password: '',

    });

    const navigate = useNavigate();

    const handleChange = (e) => {

        setCredentials({ ...credentials, [e.target.name]: e.target.value });

        console.log(credentials)

    };

    const handleSubmit = async (e) => {

        e.preventDefault();

        try {

            const response = await axios.post('http://127.0.0.1:8000/api/login', credentials);

            localStorage.setItem('authToken', response.data.token);

            setToken(response.data.token);

            navigate('/admin');

        } catch (error) {

            console.error('Greška prilikom prijavljivanja', error.response.data);

        }

    };

    return (

        <div className="login-container">

            <form className="login-form" onSubmit={handleSubmit}>

                <h2>Prijavi se</h2>

                <input type="email" name="email" placeholder="Email" value={credentials.email} onChange={handleChange} required />

                <input type="password" name="password" placeholder="Lozinka" value={credentials.password} onChange={handleChange} required />

                <button type="submit" className="login-btn">Prijavi se</button>

            </form>

        </div>

    );

};

export default Login;

Komponenta "Dogadjaji" u React aplikaciji predstavlja suštinski deo platforme za pregled događaja. Ova komponenta koristi asinhrono dohvatanje podataka za prikazivanje događaja iz lokalne baze i eksternih izvora, pružajući korisnicima mogućnost pretrage i filtriranja događaja po kategorijama. Posebno značajna je implementacija reusable komponente "Kartica", koja se koristi za prikaz pojedinačnih događaja.

import React, { useState, useEffect } from 'react';

import axios from 'axios';

import DogadjajKartica from './DogadjajKartica';

import './Dogadjaji.css';

import Kartica from './Kartica';

import { MdLastPage ,MdFirstPage} from "react-icons/md";

import { MdNavigateNext } from "react-icons/md";

import { MdNavigateBefore } from "react-icons/md";

const Dogadjaji = () => {

    const [dogadjaji, setDogadjaji] = useState([]);

    const [dogadjaji2, setDogadjaji2] = useState([]);

    const [izabranaKategorija, setIzabranaKategorija] = useState('');

    const [trenutnaStranica, setTrenutnaStranica] = useState(1);

    const dogadjajiPoStranici = 10;

    const [pretraga, setPretraga] = useState('');

    useEffect(() => { //izvlacimo dogadjaje iz nase baze

        const fetchDogadjaji = async () => {

            try {

                const response = await axios.get('http://127.0.0.1:8000/api/dogadjaji');

                setDogadjaji(response.data.data);

                console.log(response)

            } catch (error) {

                console.error('Došlo je do greške pri dobavljanju događaja', error);

            }

        };

        fetchDogadjaji();

    }, []);

    const handlePretragaChange = (e) => {

        setPretraga(e.target.value);

    };

    const handleKategorijaChange = (e) => {

        setIzabranaKategorija(e.target.value);

    };

    const filtriraniDogadjaji = dogadjaji.filter(dogadjaj =>

        dogadjaj.naziv.toLowerCase().includes(pretraga.toLowerCase())

    );

    const filtriraniDogadjaji2 = dogadjaji2.filter(dogadjaj =>

        dogadjaj.name.toLowerCase().includes(pretraga.toLowerCase())

    );

    const filtriraniDogadjaji22 = filtriraniDogadjaji2.filter(dogadjaj =>

        izabranaKategorija === '' || (dogadjaj.classifications[0].segment.name === izabranaKategorija)

    );

    useEffect(() => {

        const fetchRandomEvents = async () => {

            try {

                const response = await axios.get('https://app.ticketmaster.com/discovery/v2/events.json', {

                    params: {

                        apikey: 'rQaLInSZHng8AiA3h8qSt41RdHFKBmd3',

                        size: dogadjajiPoStranici,

                        page: trenutnaStranica,

                    }

                });

                setDogadjaji2(response.data.\_embedded.events);

                console.log(response.data.\_embedded.events)

            } catch (error) {

                console.error('Došlo je do greške prilikom dobavljanja događaja', error);

            }

        };

        fetchRandomEvents();

    }, [trenutnaStranica]);

    const prethodnaStranica = () => {

        setTrenutnaStranica(prev => (prev > 1 ? prev - 1 : prev));

    };

    const sledecaStranica = () => {

        setTrenutnaStranica(prev => prev + 1);

    };

    const prvaStranica = () => {

        setTrenutnaStranica(1);

    };

    const poslednjaStranica = () => {

        setTrenutnaStranica(25);

    };

    return (

        <div style={{backgroundColor:"#95A78D", display: 'flex'}}>

            <div className="filteri">

            <label>

                    <input

                        type="radio"

                        value=""

                        checked={izabranaKategorija === ''}

                        onChange={handleKategorijaChange}

                    /> Svi

                </label>

                <label>

                    <input

                        type="radio"

                        value="Sports"

                        checked={izabranaKategorija === 'Sports'}

                        onChange={handleKategorijaChange}

                    /> Sports

                </label>

                <label>

                    <input

                        type="radio"

                        value="Music"

                        checked={izabranaKategorija === 'Music'}

                        onChange={handleKategorijaChange}

                    /> Music

                </label>

                <label>

                    <input

                        type="radio"

                        value="Miscellaneous"

                        checked={izabranaKategorija === 'Miscellaneous'}

                        onChange={handleKategorijaChange}

                    /> Miscellaneous

                </label>

            </div>

            <div style={{flexGrow: 1}}>

                <input

                    type="text"

                    placeholder="Pretraži događaje"

                    value={pretraga}

                    onChange={handlePretragaChange}

                    className="pretraga-input"

                />

                <h2>Naši Događaji</h2>

                <div className="paginacija">

                    <button onClick={prvaStranica}><MdFirstPage /></button>

                    <button onClick={prethodnaStranica}><MdNavigateBefore /></button>

                    <span>Stranica {trenutnaStranica}</span>

                    <button onClick={sledecaStranica}><MdNavigateNext /></button>

                    <button onClick={poslednjaStranica}><MdLastPage /></button>

                </div>

                <div className="dogadjaji-wrapper">

                     {/\* <div className="dogadjaji-container">

                        {filtriraniDogadjaji.map(dogadjaj => (

                            <DogadjajKartica key={dogadjaj.id} dogadjaj={dogadjaj} />

                        ))}

                    </div>   \*/}

                    <h2>Spoljni Događaji</h2>

                    <div className="dogadjaji-container">

                        {filtriraniDogadjaji22.map(dogadjaj => (

                            <Kartica dogadjaj={dogadjaj}></Kartica>

                        ))}

                    </div>

                </div>

            </div>

        </div>

    );

}

export default Dogadjaji;

import React from 'react';

import './DogadjajKartica.css';

import image1 from './images/1.jpg';

// {dogadjaj.slika}

const DogadjajKartica = ({ dogadjaj }) => {

    return (

        <div className="dogadjaj-kartica">

            <div className="dogadjaj-slika-container">

                <img src={image1} alt={dogadjaj.naziv} className="dogadjaj-slika" />

                <span className="dogadjaj-tip">{dogadjaj.tip.naziv}</span>

            </div>

            <div className="dogadjaj-info">

                <h3>{dogadjaj.naziv}</h3>

                <p>{new Date(dogadjaj.datumVreme).toLocaleString()}</p>

                <p>Mesto: {dogadjaj.mesto}</p>

                <p>Organizator: {dogadjaj.organizator}</p>

                <p>Kapacitet: {dogadjaj.kapacitet}</p>

            </div>

        </div>

    );

}

export default DogadjajKartica;

Komponenta "TabelaDogadjaja" je ključna u React aplikaciji za upravljanje događajima. Ona koristi custom hook "useDogadjaji" za dohvatanje i prikaz podataka o događajima, uz funkcionalnosti za dodavanje i brisanje događaja. Aplikacija uključuje modalni prozor za unos novog događaja, sa poljima za naziv, datum, mesto, organizatora i kapacitet. Takođe, sadrži tabelu koja prikazuje sve događaje, uz opciju brisanja pojedinačnih događaja. Ova komponenta omogućava efikasno upravljanje događajima i unapređuje korisničko iskustvo.

import React, { useState, useEffect } from 'react';

import axios from 'axios';

import RedTabele from './RedTabele';

import './TabelaDogadjaja.css';

import useDogadjaji from './useDogadjaji';

const TabelaDogadjaja = () => {

    const { dogadjaji, loading, error, setDogadjaji } = useDogadjaji('http://127.0.0.1:8000/api/dogadjaji');

    const [prikaziModal, setPrikaziModal] = useState(false);

    const [noviDogadjaj, setNoviDogadjaj] = useState({

        naziv: '',

        datumVreme: '',

        mesto: '',

        organizator: '',

        kapacitet: ''

    });

    const handleInputChange = (e) => {

        const { name, value } = e.target;

        setNoviDogadjaj(prevState => ({

            ...prevState,

            [name]: value

        }));

    };

    const dodajDogadjaj = () => {

        setDogadjaji(prevDogadjaji => [...prevDogadjaji, { ...noviDogadjaj, id: Date.now() }]);

        setPrikaziModal(false); // Sakrij modal nakon dodavanja

    };

    const obrisiDogadjaj = (id) => {

        setDogadjaji(prevDogadjaji => prevDogadjaji.filter(d => d.id !== id));

    };

    if (loading) return <p>Učitavanje događaja...</p>;

    if (error) return <p>Došlo je do greške: {error.message}</p>;

    return (

        <>

              <button onClick={() => setPrikaziModal(true)}>Dodaj Novi Događaj</button>

              {prikaziModal && (

                <div className="modal">

                    <div className="modal-content">

                        <span className="close" onClick={() => setPrikaziModal(false)}>&times;</span>

                        <form onSubmit={e => { e.preventDefault(); dodajDogadjaj(); }}>

                            <input

                                type="text"

                                name="naziv"

                                value={noviDogadjaj.naziv}

                                onChange={handleInputChange}

                                placeholder="Naziv događaja"

                                required

                            />

                            <input

                                type="datetime-local"

                                name="datumVreme"

                                value={noviDogadjaj.datumVreme}

                                onChange={handleInputChange}

                                placeholder="Datum i vreme"

                                required

                            />

                            <input

                                type="text"

                                name="mesto"

                                value={noviDogadjaj.mesto}

                                onChange={handleInputChange}

                                placeholder="Mesto"

                                required

                            />

                            <input

                                type="text"

                                name="organizator"

                                value={noviDogadjaj.organizator}

                                onChange={handleInputChange}

                                placeholder="Organizator"

                                required

                            />

                            <input

                                type="number"

                                name="kapacitet"

                                value={noviDogadjaj.kapacitet}

                                onChange={handleInputChange}

                                placeholder="Kapacitet"

                                min="1"

                                required

                            />

                            <button type="submit">Sačuvaj Događaj</button>

                        </form>

                    </div>

                </div>

            )}

            <table>

            <thead>

                <tr>

                    <th>Naziv</th>

                    <th>Datum i Vreme</th>

                    <th>Mesto</th>

                    <th>Organizator</th>

                    <th>Kapacitet</th>

                    <th>Obrisi</th>

                </tr>

            </thead>

            <tbody>

                {dogadjaji.map(dogadjaj => (

                        <RedTabele key={dogadjaj.id} dogadjaj={dogadjaj} obrisiDogadjaj={obrisiDogadjaj} />

                    ))}

            </tbody>

            </table>

        </>

    );

}

export default TabelaDogadjaja;

import React from 'react';

const RedTabele = ({ dogadjaj, obrisiDogadjaj }) => {

    return (

        <tr>

            <td>{dogadjaj.naziv}</td>

            <td>{new Date(dogadjaj.datumVreme).toLocaleString()}</td>

            <td>{dogadjaj.mesto}</td>

            <td>{dogadjaj.organizator}</td>

            <td>{dogadjaj.kapacitet}</td>

            <td>

                <button onClick={() => obrisiDogadjaj(dogadjaj.id)}>Obriši</button>

            </td>

        </tr>

    );

}

export default RedTabele;

import { useState, useEffect } from 'react';

import axios from 'axios';

const useDogadjaji = (initialUrl) => {

    const [dogadjaji, setDogadjaji] = useState([]);

    const [loading, setLoading] = useState(false);

    const [error, setError] = useState(null);

    useEffect(() => {

        let ignore = false;

        const fetchDogadjaji = async () => {

            try {

                setLoading(true);

                const response = await axios.get(initialUrl);

                if (!ignore) setDogadjaji(response.data.data);

            } catch (err) {

                if (!ignore) setError(err);

            } finally {

                if (!ignore) setLoading(false);

            }

        };

        fetchDogadjaji();

        return () => { ignore = true; };

    }, [initialUrl]);

    return { dogadjaji, loading, error, setDogadjaji };

};

export default useDogadjaji;

Komponentu nazvana KupiUlaznice, koja je deo web aplikacije za kupovinu ulaznica za različite događaje. Komponenta koristi nekoliko ključnih elemenata i koncepta iz React ekosistema za prikazivanje informacija o događajima i omogućavanje kupovine ulaznica.

Na početku, iz React biblioteke uvozimo hook useState, koji se koristi za kreiranje lokalnog stanja unutar komponente. Takođe, uvozimo i custom hook useDogadjaji koji je verovatno zadužen za dohvatanje podataka o događajima sa zadatog URL-a (http://127.0.0.1:8000/api/dogadjaji). Osim toga, uvožene su i komponente DogadjajKartica i Modal, kao i CSS stilovi za komponentu.

Komponenta KupiUlaznice koristi useDogadjaji hook da bi dobila informacije o događajima, uključujući stanje učitavanja (loading), eventualne greške (error), i podatke o događajima (dogadjaji). Na osnovu ovih informacija, komponenta uslovljeno prikazuje poruku o učitavanju, poruku o grešci, ili listu događaja koristeći komponentu DogadjajKartica.

Za svaki događaj, komponenta DogadjajKartica prikazuje relevantne informacije i omogućava korisniku da izabere događaj za kupovinu ulaznica. Kada korisnik klikne na opciju za kupovinu, aktivira se funkcija handleKupiClick, koja postavlja izabrani događaj u lokalno stanje pomoću setSelectedDogadjaj, čime se određuje koji će događaj biti prikazan u modalnom prozoru. Funkcija takođe koristi localStorage za čuvanje ID-a i cene odabranog događaja, što može biti korisno za proces kupovine.

Kada je događaj odabran, prikazuje se modalni prozor (Modal komponenta) koji sadrži detalje o odabranom događaju i omogućava korisniku da potvrdi kupovinu. Funkcija zatvoriModal omogućava korisniku da zatvori modalni prozor i vrati se na listu događaja bez kupovine.

import React, { useState } from 'react';

import useDogadjaji from './useDogadjaji';

import DogadjajKartica from './DogadjajKartica';

import './KupiUlaznice.css';

import Modal from './Modal';

const KupiUlaznice = () => {

    const { dogadjaji, loading, error } = useDogadjaji('http://127.0.0.1:8000/api/dogadjaji');

    const [selectedDogadjaj, setSelectedDogadjaj] = useState(null);

    const handleKupiClick = (dogadjaj) => {

        setSelectedDogadjaj(dogadjaj.id);

        console.log(dogadjaj.id)

        localStorage.setItem("dogadjaj\_id",dogadjaj.id);

        localStorage.setItem("dogadjaj\_cena",dogadjaj.cena);

    };

    function zatvoriModal(){

        setSelectedDogadjaj(null);

    }

    if (loading) return <p>Učitavanje događaja...</p>;

    if (error) return <p>Došlo je do greške: {error.message}</p>;

    console.log(dogadjaji)

    return (

        <div className="kupi-ulaznice-container">

            {dogadjaji.map(dogadjaj => (

                 <DogadjajKartica key={dogadjaj.id} dogadjaj={dogadjaj} onKupiClick={handleKupiClick} />

            ))}

              {selectedDogadjaj && <Modal dogadjaj={selectedDogadjaj} onClose={zatvoriModal } />}

        </div>

    );

}

export default KupiUlaznice;

# GitHub link ka repozitorijumu

<https://github.com/elab-development/internet-tehnologije-projekat-prodajaulaznica_2020_0286>

# Reference

* Vlajić, S. (2020). Projektovanje softvera. FON.
* W3C. (s.d.). HTML i CSS specifikacije. https://www.w3.org/standards/webdesign/htmlcss
* MySQL. (s.d.). Zvanična web stranica MySQL baze podataka. https://www.mysql.com/
* Lazarević, B., Marjanović, Z., Aničić, N., & Babarogić, S. (n.d.). Baze podataka. FON.
* Tešić, I. (2017, 27. decembar). Uvod u React ekosistem [Članak]. Medium. <https://medium.com/@ITesic/uvod-u-react-ekosistem-8ccfad0a1030>
* React. (n.d.). Learn React [Web stranica]. <https://react.dev/learn>
* Laravel. (n.d.). Laravel 10.x Documentation [Web stranica]. https://laravel.com/docs/10.x