Univerzitet u Beogradu  
Fakultet organizacionih nauka  
Katedra za elektonsko poslovanje

NAZIV TEME

Seminarski rad iz predmeta Internet tehnologije

Mentor: Studenti:

Beograd, 2023.

# Sadržaj

[Sadržaj 1](#_Toc127376267)

[1. Korisnički zahtev 1](#_Toc127376268)

[2. Opis sistema 2](#_Toc127376269)

[1. Opis slučajeva korišćenja 2](#_Toc127376270)

[1.1. SK1: Slučaj korišćenja – Prijava korisnika 2](#_Toc127376271)

[1.2. SK2: Slučaj korišćenja – Kreiranje karte 3](#_Toc127376272)

[1.3. SK3: Slučaj korišćenja – Brisanje dogadjaja 4](#_Toc127376273)

[2. Opis arhitekture aplikacije 6](#_Toc127376274)

[3. Opis procesa slučajeva korišćenja (dijagrami sekvenci) 7](#_Toc127376275)

[3.1. DS1: Dijagram sekvenci – Kreiranje naloga 7](#_Toc127376276)

[3.2. DS2: Dijagram sekvenci – Kreiranje karte 9](#_Toc127376277)

[3.3. DS3: Dijagram sekvence slučaja korišćenja – Brisanje dogadjaja 11](#_Toc127376278)

[4. Model podataka 14](#_Toc127376279)

[5. Specifikacija REST API-ja 15](#_Toc127376280)

[3. Opis tehnologija korišćenih u aplikaciji 21](#_Toc127376281)

[1. JavaScript 21](#_Toc127376282)

[2. React JS 22](#_Toc127376283)

[3. PHP 22](#_Toc127376284)

[4. Bootstrap 22](#_Toc127376285)

[5. Laravel 22](#_Toc127376286)

[4. Korisničko uputstvo 24](#_Toc127376287)

[5. Prikaz reprezentativnih delova koda 27](#_Toc127376288)

[6. GitHub link ka repozitorijumu 41](#_Toc127376289)

[7. Reference 42](#_Toc127376290)

# Korisnički zahtev

Potrebno je kreirati aplikaciju koja će omogućiti građanima uvid u dešavanja u Beogradu. Kako bi korisnici mogli da koriste aplikaciju neophodno je da se prvo uloguju. Tek nakon uspešnog prijavljivanja omogućiti korisnicima aplikacije pregled aktuelnih dešavanja u Beogradu. Radi lakšeg snalaženja neophodno je napraviti stranicu na kojoj će korisnici moći jasno i pregledno da vide detalje o svakom događaju koji ih interesuje. To podrazumeva prikaz naziva, vremena, datuma, mesta, kao i kategorije određenog događaja. Korisnicima treba omogućiti mogućnost rezervacije karata, kao i prikaz svih karata koje su kupljene do sada.

Osim toga, neophodno je napraviti i admin stranicu na kojoj će se nalaziti statiskike u vezi kupovina karata za događaje. Potrebno je omogućiti administratorima mogućnost dodavanja novih događaja, brisanja i ažuriranja postojećih.

Za razvoj aplikacije koristiti React I Laravel.

# Opis sistema

## Opis slučajeva korišćenja

Aplikacija se sastoji od velikog broja SK,pa ćemo izdvojiti nekoliko:

1. Prijava korisnika
2. Kreiranje karte
3. Brisanje dogadjaja

### 1.1. SK1: Slučaj korišćenja – Prijava korisnika

Naziv SK:  
Prijava korisnika

Aktori SK:  
Korisnik (administrator)

Učesnici SK:  
Korisnik i sistem (program)

Preduslov: Sistem je uključen i korisniku prikazuje formu za prijavu.

Osnovni scenario SK

1. Korisnik unosi podatke za prijavu. (APUSO)
2. Korisnik proverava da li je korektno uneo podatke u nalog. (ANSO)
3. Korisnik poziva sistem da zapamti pronadje nalog. (APSO)
4. Sistem trazi nalogu. (SO)
5. Sistem prikazuje korisniku nalog i poruku: “Sistem je pronasao nalog“. (IA)

Alternativna scenarija

5.1 Ukoliko sistem ne može da nadje nalog on prikazuje korisniku poruku: “Sistem ne može da nadje nalog”. Prekida se izvršenje scenarija. (IA)

### 1.2. SK2: Slučaj korišćenja – Kreiranje karte

Naziv SK:  
Kreiranje karte

Aktori SK:  
Korisnik

Učesnici SK:  
Korisnik i sistem (program)

Preduslov: Sistem je uključen i korisniku prikazuje formu za rad sa kartama

Osnovni scenario SK

1. Korisnik unosi podatke o karti. (APUSO)
2. Korisnik proverava da li je korektno uneo podatke. (ANSO)
3. Korisnik poziva sistem da zapamti podatke o karti. (APSO)
4. Sistem pamti podatke o karti. (SO)
5. Sistem prikazuje korisniku zapamćenu kartui poruku: “Sistem je zapamtio kartu “. (IA)

Alternativna scenarija

5.1 Ukoliko sistem ne može da zapamti podatke o karti on prikazuje korisniku poruku “Sistem ne može da zapamti zakazivanje”. (IA)

### 1.3. SK3: Slučaj korišćenja – Brisanje dogadjaja

Naziv SK:  
Brisanje dogadjaja

Aktori SK:  
Administrator

Učesnici SK:  
Administrator i sistem (program)

Preduslov: Sistem je uključen i administrator je ulogovan pod svojom šifrom. Sistem prikazuje formu za rad sa dogadjajima. Učitana je lista svih dogadjaja.

Osnovni scenario SK

1. Administrator unosi vrednost po kojoj pretražuje dogadjaje. (APUSO)

2. Administrator poziva sistem da nađe dogadjaje po zadatoj vrednosti. (APSO)

3. Sistem traži dogadjaje po zadatoj vrednosti. (SO)

4. Sistem prikazuje administratoru podatke o dogadjajima i poruku: “Sistem je našao dogadjaje po zadatoj vrednosti”. (IA)

5. Administrator bira dogadjaj. (APUSO)

6. Administrator poziva sistem da nađe odabrani dogadjaj. (APSO)

7. Sistem traži dogadjaje po zadatoj vrednosti. (SO)

8. Sistem prikazuje podatke o školi i poruku: “Sistem je našao dogadjaje po zadatoj vrednosti”. (IA)

9. Administrator poziva sistem da obriše dogadjaj. (APSO)

10. Sistem briše dogadjaj. (SO)

11. Sistem prikazuje administratoru poruku: “Sistem je obrisao dogadjaj.” (IA)

Alternativna scenarija

4.1 Ukoliko sistem ne može da pronađe dogadjaje po zadatoj vrednosti, prikazuje poruku: “Sistem ne može da pronađe dogadjaje po zadatoj vrednosti“. Prekida se izvršavanje scenaria. (IA)

8. 1 Ukoliko sistem ne može da učita podatke o odabranom dogadjaju, prikazuje poruku: “Sistem ne može da učita podatke o odabranom dogadjaju“. (IA)

11.1 Ukoliko sistem ne može da obriše dogadjaj, prikazuje poruku: “Sistem ne može da obriše dogadjaj“. (IA)

## Opis arhitekture aplikacije

Aplikacija se sastoji iz dva dela

* Backend – Laravel projekat
* Frontend – react projekat

Komponente aplikacije koje su vidljive na frontendu su:

Slika na kojoj se nalazi tekst

Opis je automatski generisan

Slika - komponente aplikacije

Glavna komponenta je Example.js. U njoj se nalazi spisak svih ruta koje postoje u aplikaciji I ona ih koordiniše.

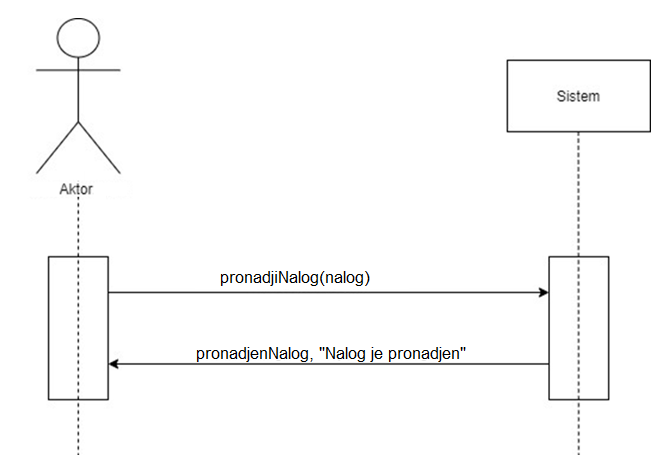
Komponente se povezuju sa bazom podataka i komuniciraju uz korišćenje axios-a.

## Opis procesa slučajeva korišćenja (dijagrami sekvenci)

### DS1: Dijagram sekvenci – Kreiranje naloga

Osnovni scenario SK

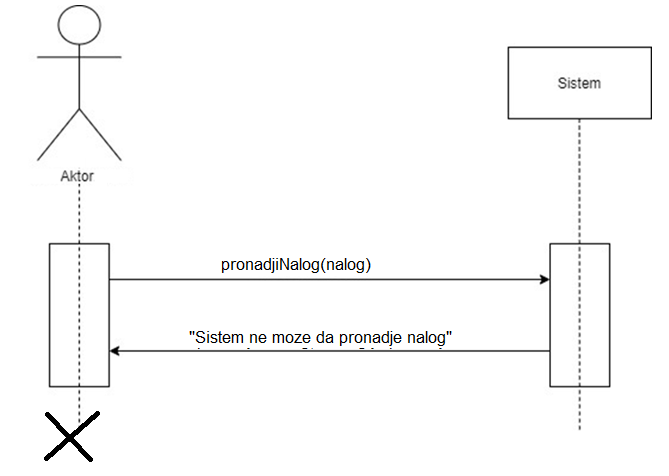
1. Korisnik unosi podatke za prijavu. (APUSO)
2. Korisnik proverava da li je korektno uneo podatke u nalog. (ANSO)
3. Korisnik poziva sistem da zapamti pronadje nalog. (APSO)
4. Sistem trazi nalogu. (SO)
5. Sistem prikazuje korisniku nalog i poruku: “Sistem je pronasao nalog“. (IA)



Slika . Dijagram sekvenci – Prijava korisnika

Alternativna scenarija

* 1. Ukoliko sistem ne može da pronadje nalog on prikazuje korisniku poruku: “Sistem ne može da pronadje nalog”. Prekida se izvršenje scenarija. (IA)



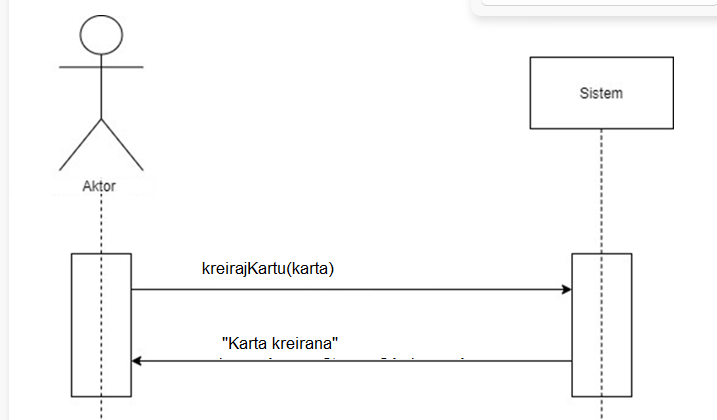
Slika . Dijagram sekvenci – Prijava korisnika/alternativni scenario 1

4.1 Ukoliko sistem ne može da zapamti podatke o nalogu on prikazuje korisnikuocu poruku “Sistem ne može da zapamti nalog”. (IA)

### DS2: Dijagram sekvenci – Kreiranje karte

Osnovni scenario SK

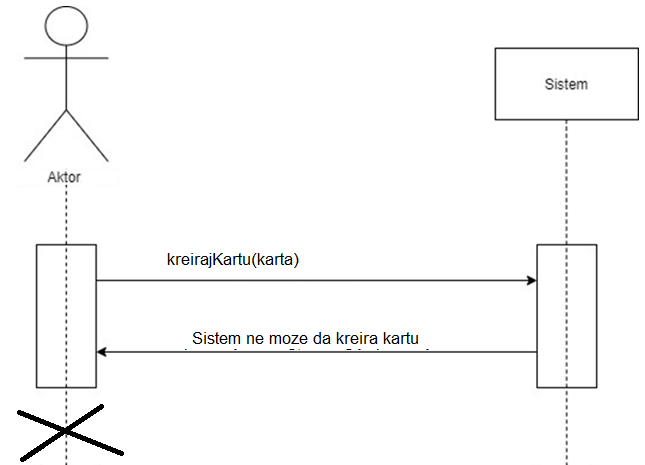
1. Korisnik unosi podatke o karti. (APUSO)
2. Korisnik proverava da li je korektno uneo podatke. (ANSO)
3. Korisnik poziva sistem da zapamti podatke o karti. (APSO)
4. Sistem pamti podatke o karti. (SO)
5. Sistem prikazuje korisniku zapamćenu kartui poruku: “Sistem je zapamtio kartu “. (IA)



Slika . Dijagram sekvenci – kreiranje karte

Alternativna scenarija

3.1 Ukoliko sistem ne može da kreira kartu prikazuje korisniku poruku: “Sistem ne može da kreira kartu”. Prekida se izvršenje scenarija. (IA)



Slika . Dijagram sekvenci – kreiranje karte alternativni scenario

### DS3: Dijagram sekvence slučaja korišćenja – Brisanje dogadjaja

Osnovni scenario SK

1. Forma poziva sistem da učita listu dogadjaja. (APSO)

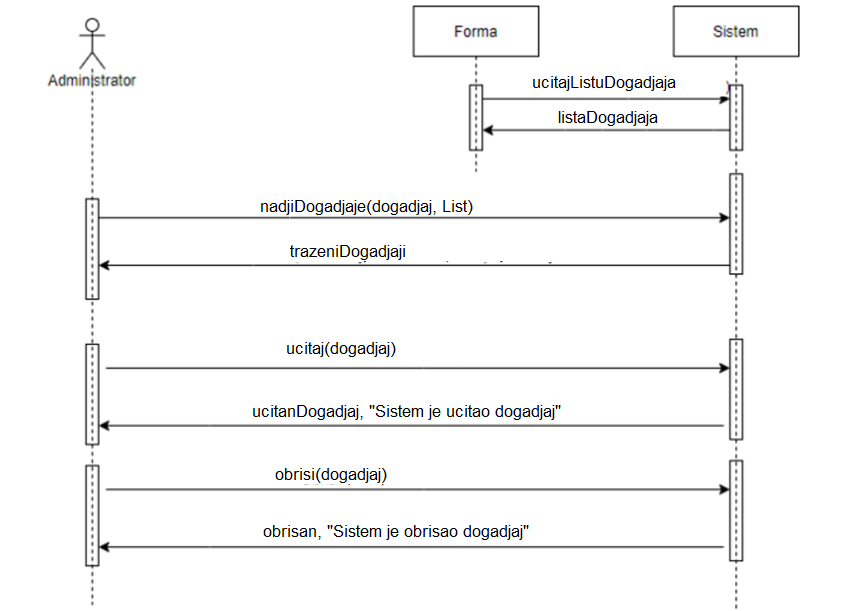
2. Sistem vraća formi listu dogadjaja. (IA)

3. Administrator poziva sistem da nađe dogadjaje po zadatoj vrednosti (APSO)

4. Sistem prikazuje administratoru podatke o dogadjajima i poruku: “Sistem je našao skole po dogadjaje vrednosti”. (IA)

5. Administrator poziva sistem da obriše dogadjaj. (APSO)

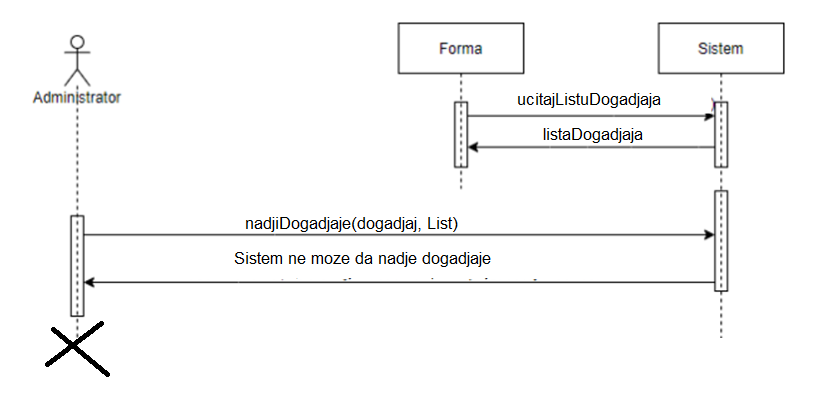
6. Sistem briše dogadjaj. (SO)



Slika . Dijagram sekvenci – Brisanje dogadjaja

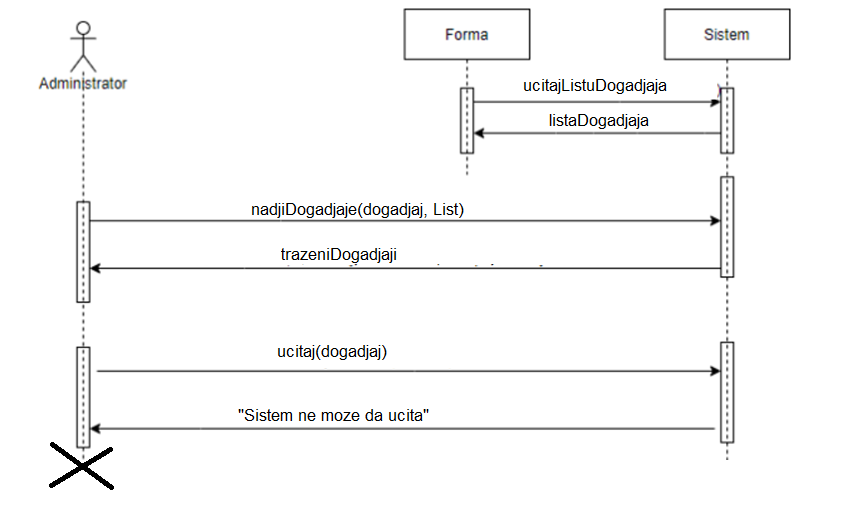
Alternativna scenarija

4.1 4.1 Ukoliko sistem ne može da pronađe dogadjaje po zadatoj vrednosti, prikazuje poruku: “ Sistem ne može da pronađe dogadjaje po zadatoj vrednosti“. Prekida se izvršavanje scenaria. (IA)



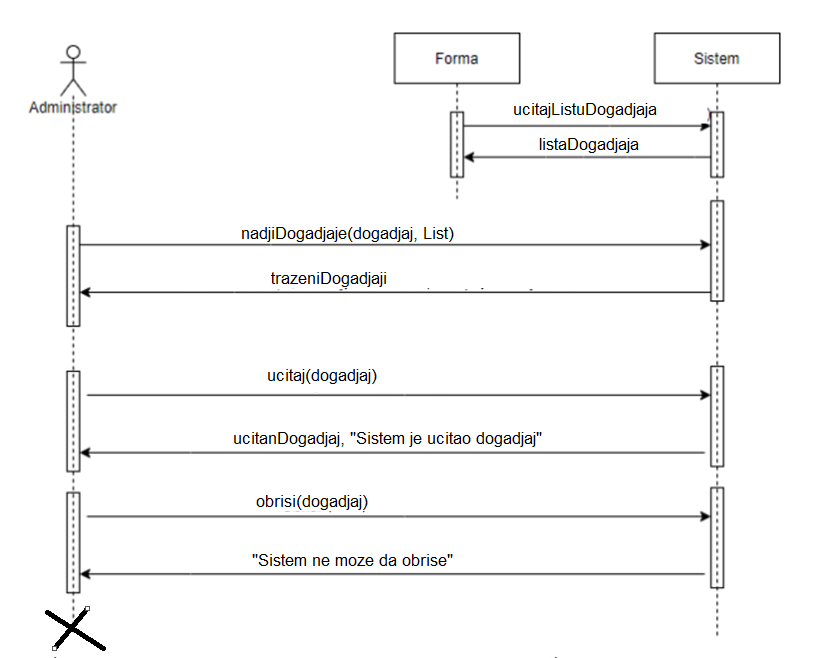
Slika . Dijagram sekvenci – Brisanje dogadjaja alt. scenario

8. 1 Ukoliko sistem ne može da učita podatke o odabranom dogadjaju, prikazuje poruku: “Sistem ne može da učita podatke o odabranom dogadjaju“. (IA)



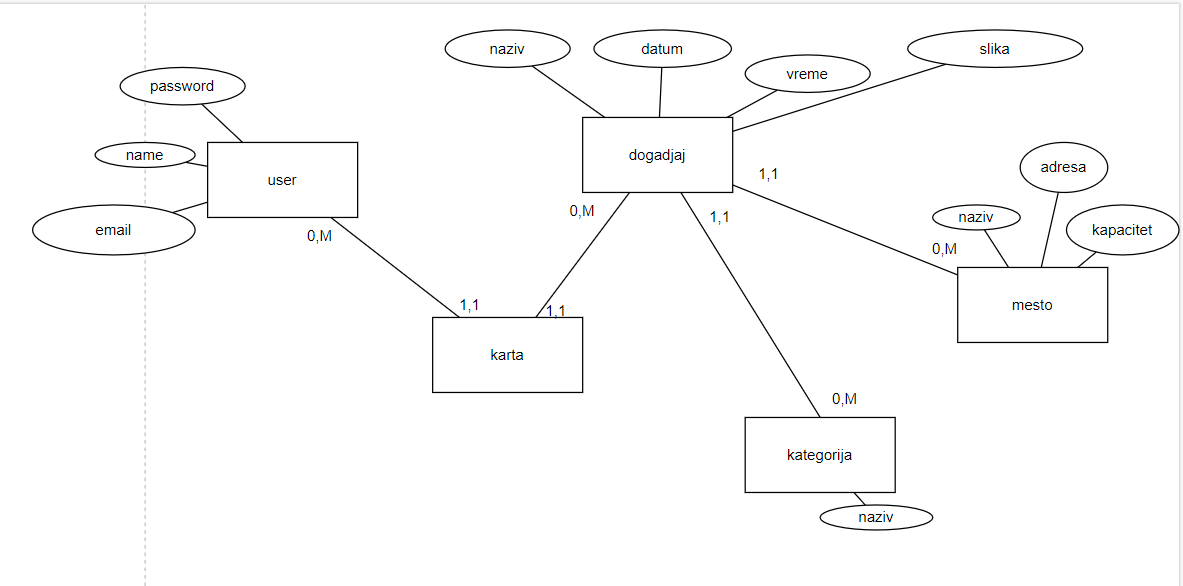
Slika 8. Dijagram sekvenci – Brisanje dogadjaja alt. scenario

11.1 Ukoliko sistem ne može da obriše dogadjaj, prikazuje poruku: “Sistem ne može da obriše dogadjaj“. (IA)



Slika 9. Dijagram sekvenci – Brisanje dogadjaja alt. scenario

## Model podataka



Slika . PMOV

## Specifikacija REST API-ja

|  |  |
| --- | --- |
| Opis funkcije | Registracija |
| HTTP metoda | POST |
| URL | /api/register |
| URL parametri | ?name=Ivona&email=ivona@gmail.com&password=ivona |
| HTTP body parametri | {      "name": "ivona",      "email": "ivona@gmail.com",      "password": "ivona",    } |
| Format HTTP body parametara | JSON |
| Izlazni parametri | {      "data": {          "name": "Ivona",          "email": "ivona@gmail.com",          "updated\_at": "2022-12-22T21:05:37.000000Z",          "created\_at": "2022-12-22T21:05:37.000000Z",          "id": 12      },      "acess\_token": "3|ReEm0L0olyoHbyKfcrcFIfBMlCUQsPhxS7yjEnKS",      "token\_type": "Bearer"  } |
| Format izlaznih parametara | application/json |
|  |  |
| Opis funkcije | Login |
| HTTP metoda | POST |
| URL | /api/login |
| URL parametri | ? email=ivona@gmail.com&password=ivona |
| HTTP body parametri | {        "email": "ivona@gmail.com",      "password": "ivona",    } |
| Format HTTP body parametara | JSON |
| Izlazni parametri | {      "user": {          "id": 12,          "name": "Ivona",          "email": "ivona@gmail.com",          "email\_verified\_at": **null**,          "created\_at": "2022-12-22T21:05:37.000000Z",          "updated\_at": "2022-12-22T21:05:37.000000Z"      },      "token": "4|KkCj615Az1cQcjtisSGGFICuyPFb8bJ7dTRs2j3I"  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Odjava |
| HTTP metoda | POST |
| URL | /api/logout |
| URL parametri | nema |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {  "status": "success",    }, |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Dodavanje novog dogadjaja |
| HTTP metoda | POST |
| URL | /api/dogadjaji |
| URL parametri | ?naziv=AAA&kategorija\_id=1&mesto\_id=1&vreme=10h&datum=2022-12-12 |
| HTTP body parametri | {        " naziv ": " AAA",      " kategorija ": 1,  " mesto ": 1,  " vreme ": "10h",  " datum ": "2022-12-12",    } |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | [      "Dogadjaj kreiran!",      {          "naziv": "AAA",          "vreme": "10h",          "datum": "2022-12-12",          "kategorija\_id": "1",          "mesto\_id": "1",          "updated\_at": "2022-12-22T21:08:54.000000Z",          "created\_at": "2022-12-22T21:08:54.000000Z",          "id": 12      }  ] |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Brisanje dogadjaja |
| HTTP metoda | DELETE |
| URL | /api/dogadjaji/{id} |
| URL parametri | 1 |
| HTTP body parametri | nema |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | [      "Uspesno obrisano!",      {          "id": 1,          "naziv": "cumque",          "vreme": "16:44",          "datum": "1995-02-22",          "kategorija\_id": 1,          "mesto\_id": 2,          "created\_at": "2022-08-27T20:56:36.000000Z",          "updated\_at": "2022-08-27T20:56:36.000000Z"      }  ] |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Izmena dogadjaja |
| HTTP metoda | PUT |
| URL | /api/dogadjaji/{id} |
| URL parametri | 2?http://127.0.0.1:8000/api/dogadjaji/2?naziv=AAA&kategorija\_id=1&mesto\_id=1&vreme=10h&datum=2022-12-12 |
| HTTP body parametri | nema |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | [      "Uspesno izmenjeno!",      {          "id": 2,          "naziv": "AAA",          "vreme": "10h",          "datum": "2022-12-12",          "kategorija\_id": "1",          "mesto\_id": "1",          "created\_at": "2022-08-27T20:56:36.000000Z",          "updated\_at": "2022-12-22T21:12:03.000000Z"      }  ] |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Prikaz svih dogadjaja |
| HTTP metoda | GET |
| URL | /api/dogadjaji |
| URL parametri | nema |
| HTTP body parametri | nema |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | [   {              "id": 1,              "naziv": "Benito Roman",              "vreme": "10:00",              "datum": "2023-02-20",              "kategorija": {                  "id": 5,                  "naziv": "Umetnost i kultura",                  "created\_at": "2023-02-15T14:38:42.000000Z",                  "updated\_at": "2023-02-15T14:38:42.000000Z"              },              "mesto": {                  "id": 1,                  "naziv\_mesta": "Institut Servantes",                  "kapacitet": 100,                  "created\_at": "2023-02-15T14:38:42.000000Z",                  "updated\_at": "2023-02-15T14:38:42.000000Z",                  "adresa": "Čika-Ljubina 19"              },              "slika": "https://belgrade-beat.rs/photos/activities/3353/t-1669822980.jpg",              "omiljen": 0          },          {              "id": 2,              "naziv": "Radovi na papiru",              "vreme": "12:00",              "datum": "2023-02-20",              "kategorija": {                  "id": 4,                  "naziv": "Aktivnost i dozivljaj",                  "created\_at": "2023-02-15T14:38:42.000000Z",                  "updated\_at": "2023-02-15T14:38:42.000000Z"              },              "mesto": {                  "id": 1,                  "naziv\_mesta": "Institut Servantes",                  "kapacitet": 100,                  "created\_at": "2023-02-15T14:38:42.000000Z",                  "updated\_at": "2023-02-15T14:38:42.000000Z",                  "adresa": "Čika-Ljubina 19"              },              "slika": "https://belgrade-beat.rs/photos/activities/3426/t-1673982062.jpg",              "omiljen": 0          },          {              "id": 3,              "naziv": "Ledeno carstvo",              "vreme": "14:00",              "datum": "2023-02-20",              "kategorija": {                  "id": 3,                  "naziv": "Za decu",                  "created\_at": "2023-02-15T14:38:42.000000Z",                  "updated\_at": "2023-02-15T14:38:42.000000Z"              },              "mesto": {                  "id": 3,                  "naziv\_mesta": "Savski park",                  "kapacitet": 55,                  "created\_at": "2023-02-15T14:38:42.000000Z",                  "updated\_at": "2023-02-15T14:38:42.000000Z",                  "adresa": "Savski park"              },              "slika": "https://belgrade-beat.rs/photos/activities/3445/t-1674316371.jpg",              "omiljen": 0          },  …  ] |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Prikaz jednog dogadjaja |
| HTTP metoda | GET |
| URL | /api/dogadjaji/{id} |
| URL parametri | 2 |
| HTTP body parametri | nema |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {              "id": 2,              "naziv": "Radovi na papiru",              "vreme": "12:00",              "datum": "2023-02-20",              "kategorija": {                  "id": 4,                  "naziv": "Aktivnost i dozivljaj",                  "created\_at": "2023-02-15T14:38:42.000000Z",                  "updated\_at": "2023-02-15T14:38:42.000000Z"              },              "mesto": {                  "id": 1,                  "naziv\_mesta": "Institut Servantes",                  "kapacitet": 100,                  "created\_at": "2023-02-15T14:38:42.000000Z",                  "updated\_at": "2023-02-15T14:38:42.000000Z",                  "adresa": "Čika-Ljubina 19"              },              "slika": "https://belgrade-beat.rs/photos/activities/3426/t-1673982062.jpg",              "omiljen": 0          }, |
| Format izlaznih parametara | application/json |

# Opis tehnologija korišćenih u aplikaciji

### JavaScript

JavaScript je skriptni programski jezik, koji se izvršava u web pregledaču na strani korisnika. Napravljen je da bude sličan Javi, zbog lakšega korišćenja, ali nije objektno orijentiran kao Java, već se temelji na prototipu i tu prestaje svaka povezanost s programskim jezikom Java. Izvorno ga je razvila tvrtka Netscape (www.netscape.com). JavaScript je primjena ECMAScript standarda.

JavaScript s AJAX (Asynchronous JavaScript and XML) tehnikom omogućuje web stranicama komunikaciju sa serverskim programom, što čini web aplikaciju interaktivnijom i lakšom za korištenje.

### React JS

**React** (poznat i kao **React.js** ili **ReactJS**) je Javaskrpit biblioteka otvorenog koda koja obezbeđuje pregled podataka zapisanih preko  [HTMLa](https://sr.wikipedia.org/wiki/HTML" \o "HTML). React pregledi su obično obezbeđeni korišćenjem komponenti koje sadrže dodatne komponente definisane kao prilagođene HTML oznake. React obezbeđuje programeru model u kojem podkomponente ne mogu direktno da utiču na spoljašnje komponente, efikasno ažuriranje HTML dokumenta pri promeni podataka i jasno razdvajanje komponenti na današnjim jednostraničnim aplikacijama.

### PHP

PHP (Hypertext Preprocessor) je široko rasprostranjen jezik za skriptovanje opšte namene otvorenog koda koji je posebno pogodan za web razvoj i može se ugraditi u HTML.

PHP je stekao popularnost zbog svoje jednostavnosti i sintakse nasleđene iz programskog jezika [C](https://sr.wikipedia.org/wiki/C_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%81%D0%BA%D0%B8_%D1%98%D0%B5%D0%B7%D0%B8%D0%BA)). Tokom vremena jezik se proširivao i sticao mogućnosti za [objektno orijentisano programiranje](https://sr.wikipedia.org/wiki/%D0%9E%D0%B1%D1%98%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D1%98%D0%B5%D0%BD%D1%82%D0%B8%D1%81%D0%B0%D0%BD%D0%BE_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B8%D1%80%D0%B0%D1%9A%D0%B5" \o "Objektno-orijentisano programiranje), naročito od verzije 5.0. Nalikuje jeziku [C++](https://sr.wikipedia.org/wiki/C%2B%2B) u smislu što dozvoljava i čisto-proceduralno programiranje, ali istovremeno omogućava i korišćenje [klasa](https://sr.wikipedia.org/sr-el/PHP" \l "%D0%9A%D0%BB%D0%B0%D1%81%D0%B5_%D0%B8_%D0%BE%D0%B1%D1%98%D0%B5%D0%BA%D1%82%D0%B8) i drugih koncepata objektno orijentisanog programiranja ([nasleđivanje](https://sr.wikipedia.org/sr-el/PHP" \l "%D0%9D%D0%B0%D1%81%D0%BB%D0%B5%D1%92%D0%B8%D0%B2%D0%B0%D1%9A%D0%B5), [apstraktne klase](https://sr.wikipedia.org/sr-el/PHP" \l "%D0%90%D0%BF%D1%81%D1%82%D1%80%D0%B0%D0%BA%D1%82%D0%BD%D0%B5_%D0%BA%D0%BB%D0%B0%D1%81%D0%B5) i metode, [interfejse](https://sr.wikipedia.org/sr-el/PHP" \l "%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D1%98%D1%81%D0%B8) itd.).

### Bootstrap

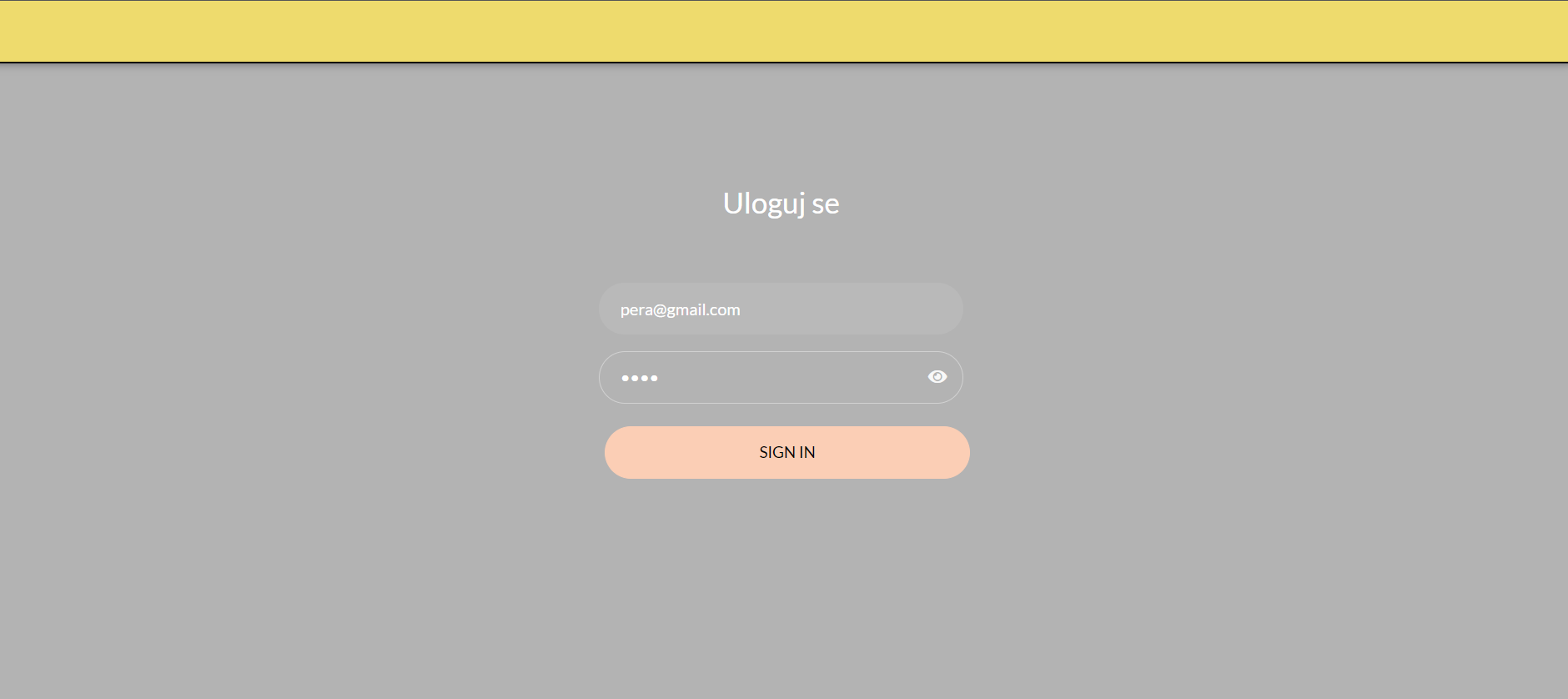
Bootstrap je besplatni web frejmvork otvorenog koda, za kreiranje veb sajtova i veb aplikacija. Baziran je na HTML i CSS šablonima za tipografiju, kreiranju formulara, dugmadi, navigacionim i ostalim komponentama interfejsa, kao i opcionim JavaSkript dodacima. Cilj Bootstrap-a je olakšavanje programiranja za veb.  
Koristi se i za web aplikacije, tj. podržava razvoj dinamičkih web sajtova i web aplikacija.

### Laravel

Laravel je besplatni PHP web okvir otvorenog koda, koji je namenjen razvoju web aplikacija koje prate arhitektonski obrazac model–pogled–kontrolor (MVC) i zasnovan na Symfony-ju. Neke od karakteristika Laravel-a su modularni sistem pakovanja sa namenskim menadžerom zavisnosti, različiti načini za pristup relacionim bazama podataka, uslužni programi koji pomažu u primeni i održavanju aplikacija, i njegova orijentacija ka syntactic sugar.  
Izvorni kod Laravel-a se nalazi na GitHub-u.

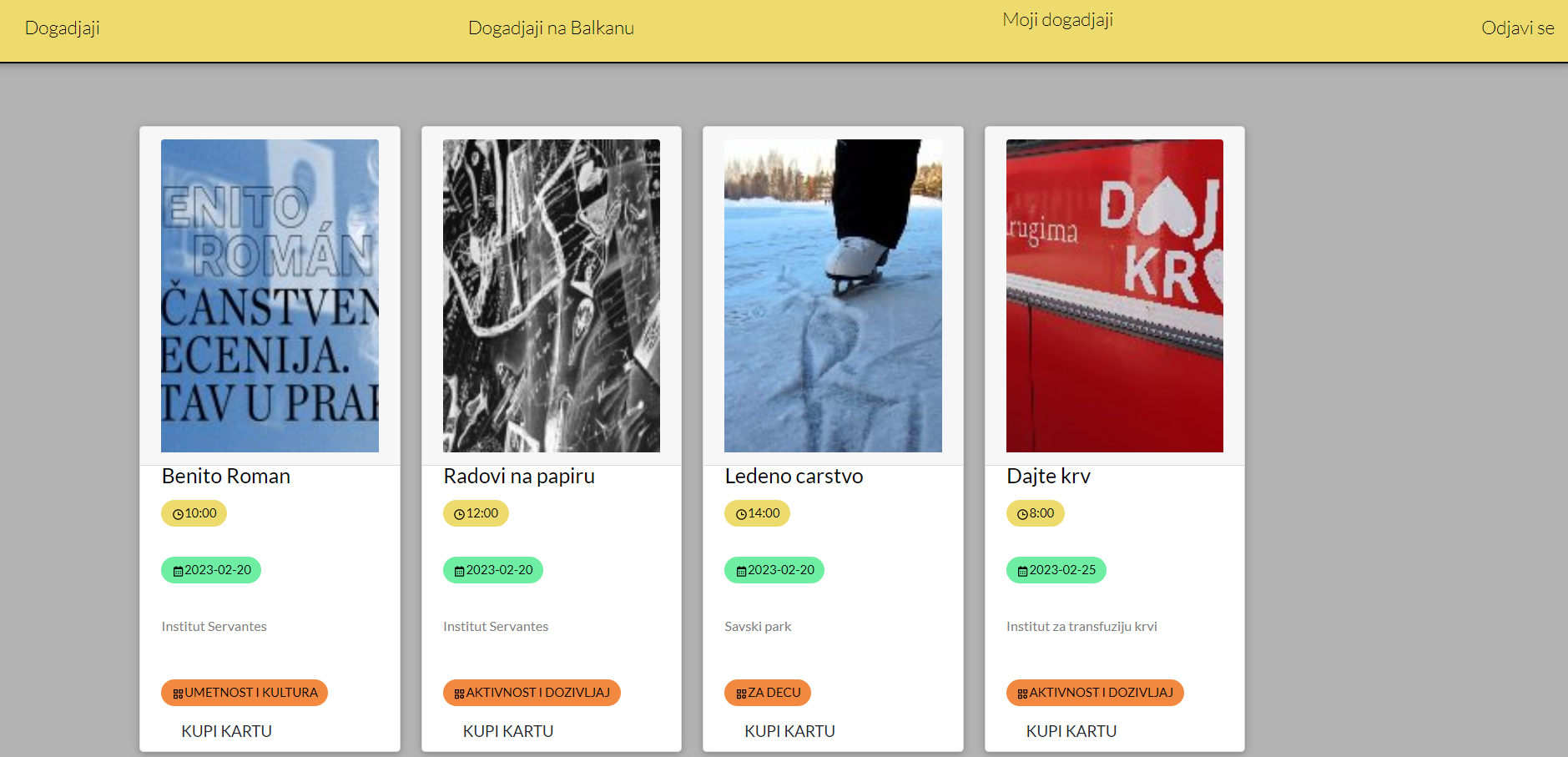
# Korisničko uputstvo

Prilikom pokretanja aplikacije otvara se stranica na kojoj se nalazi login forma.



Slika login stranica

Ukoliko se ulogujemo kao običan korisnik, prikazuje nam se stranica na kojoj su prikazani svi događaji.



Slika dogadjaji

Za svaki dogadjaj su ispisane osnovne informacije, a klikom na KUPI KARTU korisnik može da rezevriše kartu za sebe. Sve kupljene karte korisnik može da vidi na stranici Moji događaji.



Slika moji dogadjaji

Na ovoj stranici su prikazani svi događaji koje je kupio određen korisnik u vidu DataTabele. Korisnici mogu da sortiraju događaje po bilo kom kriterijumu, kao I da ih pretražuju.

Ukoliko se uloguje administrator, njemu se prikazuje adminpage kao na slici ispod

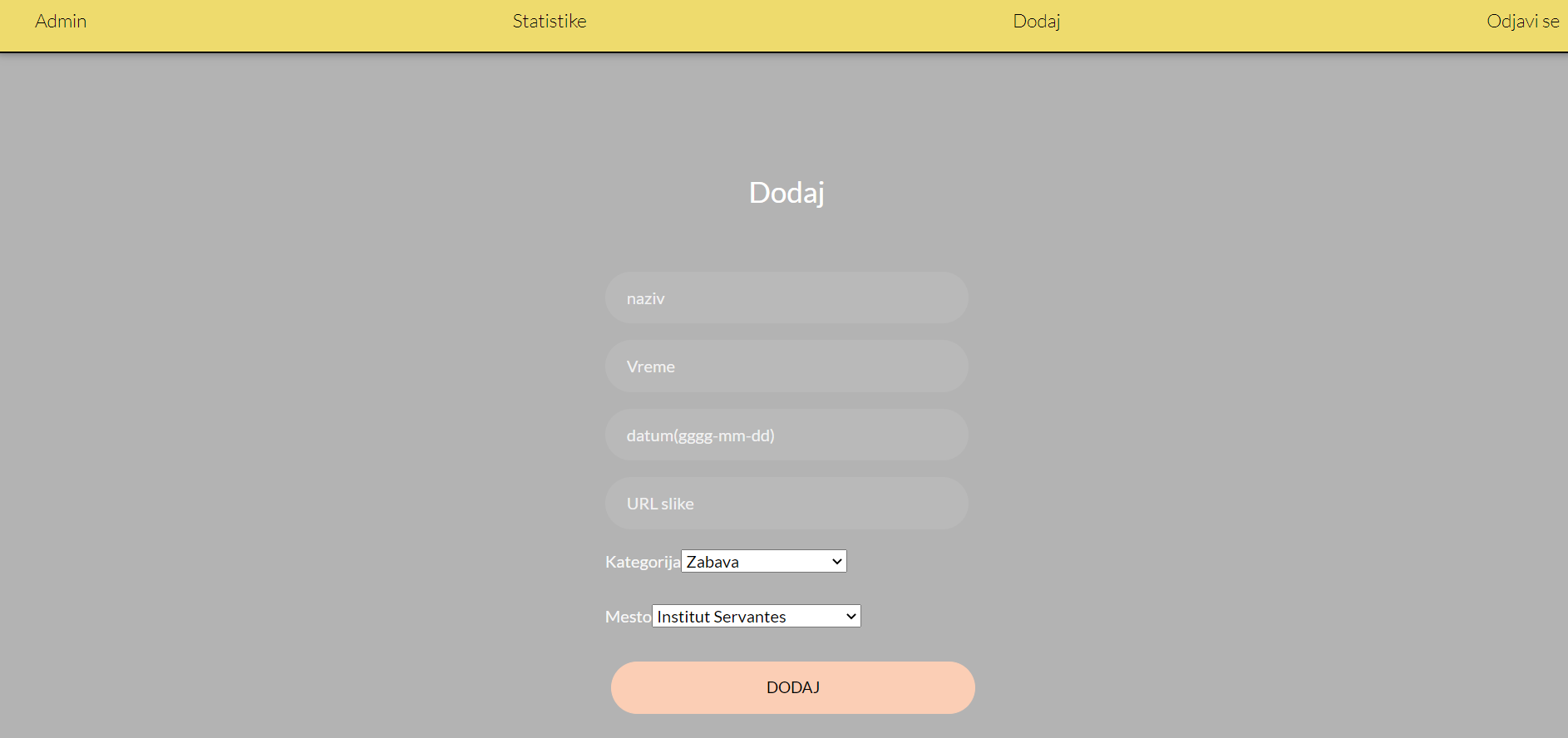
Slika na kojoj se nalazi tekst, snimak ekrana, zatvoreni prostor, nekoliko

Opis je automatski generisan

Slika - admin stranica

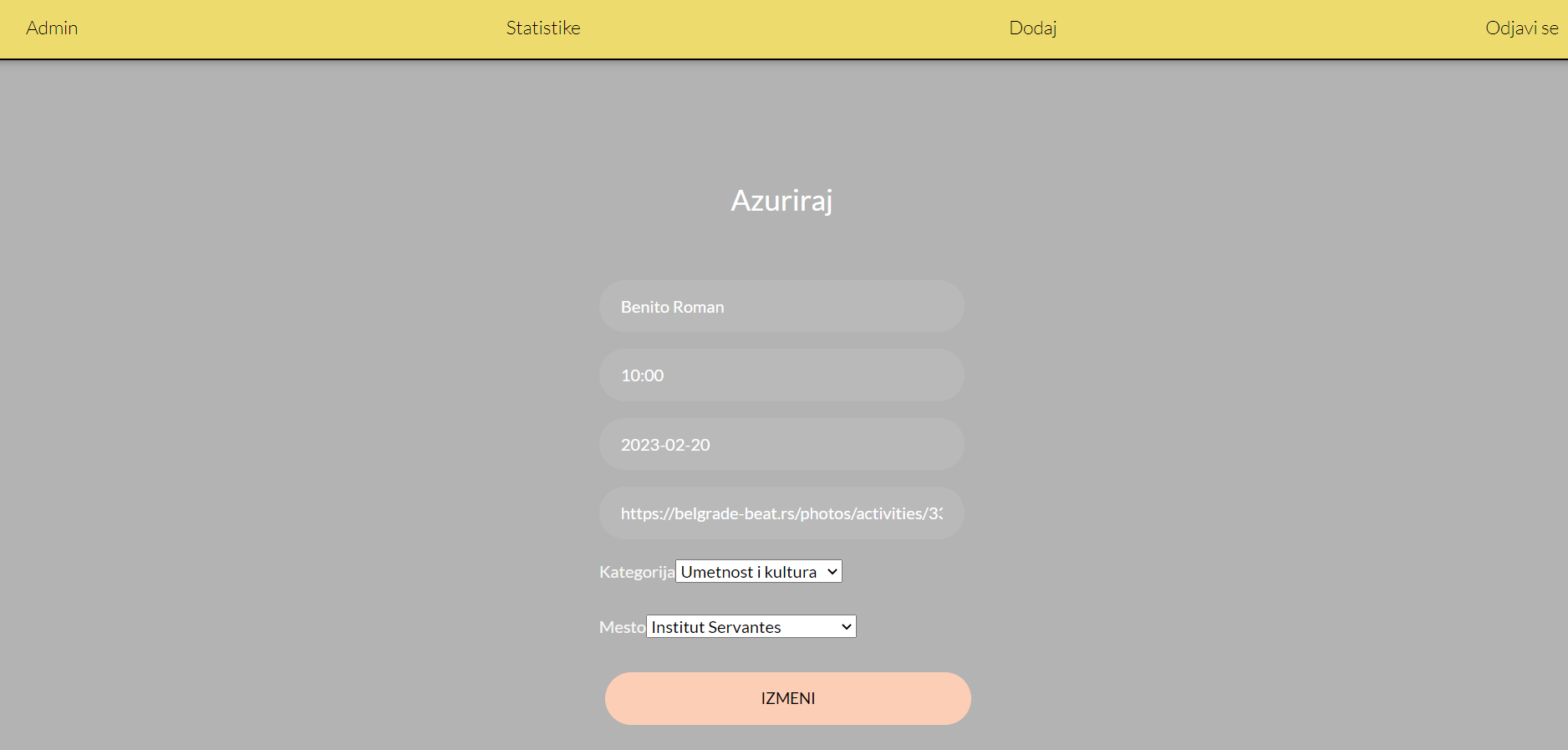
Na ovoj stranici administrator ima mogućnost pregleda svih događaja, kao I mogućnost brisanja/ažuriranja odgovarajućeg događaja klikom na ikonicu u poslednjoj koloni.

Odabirom opcije Dodaj iz navigacionog menija, administrator se otvara forma preko koje može da unese podatke o novom događaju.



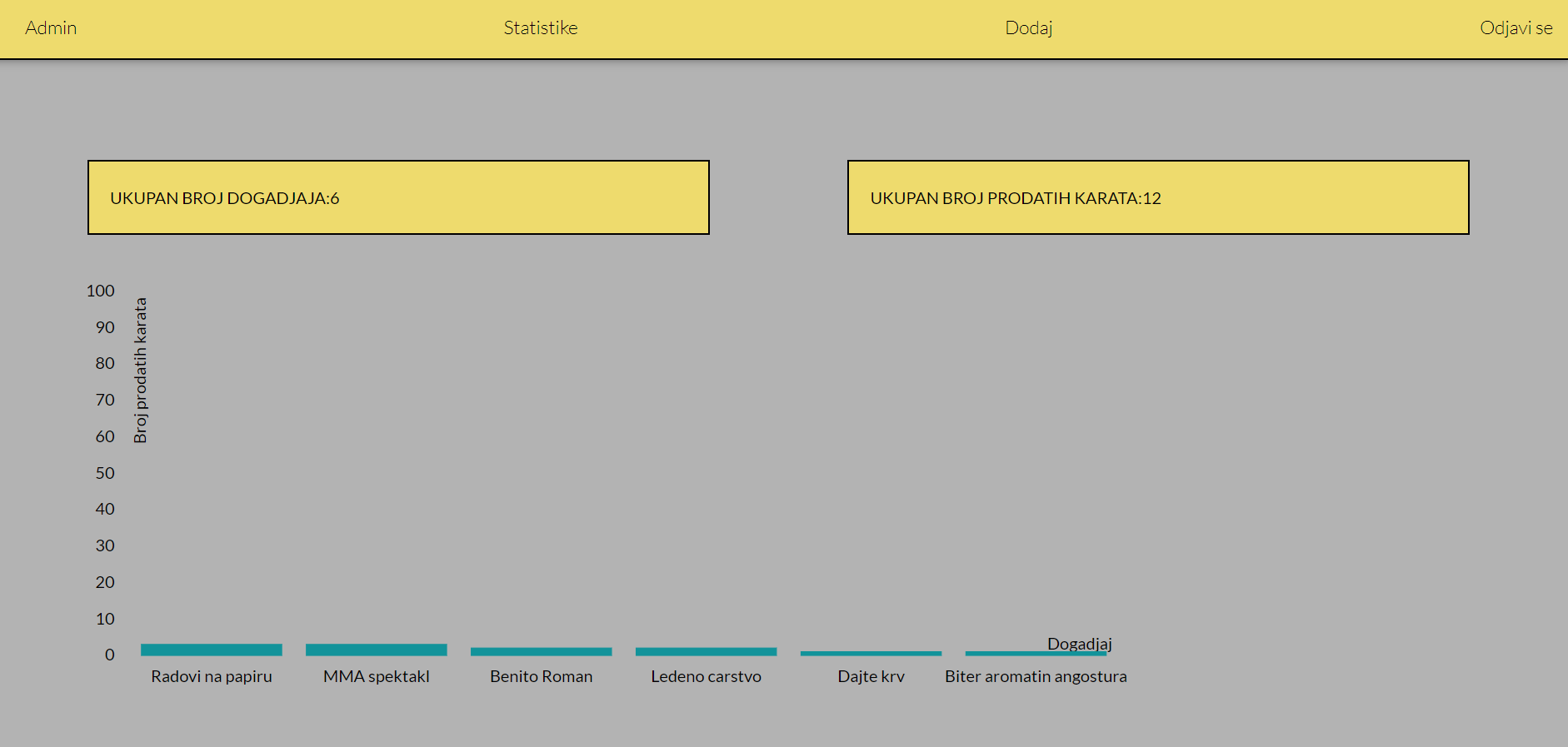
Slika Dodaj

Klikom na ikonu sa olovkicom u tabeli korisniku se otvara ista ovakva forma, ali popunjena podacima odgovarajućeg događaja.



Slika Ažuriraj

Odabirom opcije Statiskike iz navigacionog menija, administratoruj se prikazuje stranica na kojoj se nalazi kratak pregled statistika u vezi sa događajima.



Na ovoj stranici se nalazi informacija o tome koliko imamo događaja, koliko imamo kupljenih karata, kao i bar dijagram na kom možemo da vidimo koji događaji su među najpopularnijjim.

# Prikaz reprezentativnih delova koda

<?php

namespace App\Http\Resources;

use App\Models\Kategorija;

use App\Models\Mesto;

use App\Models\User;

use Illuminate\Http\Resources\Json\JsonResource;

class DogadjajResource extends JsonResource

{

    /\*\*

     \* Transform the resource into an array.

     \*

     \* @param  \Illuminate\Http\Request  $request

     \* @return array|\Illuminate\Contracts\Support\Arrayable|\JsonSerializable

     \*/

    public function toArray($request)

    {

        return  [

            'id' => $this->resource->id,

            'naziv' => $this->resource->naziv,

            'vreme' => $this->resource->vreme,

            'datum' => $this->resource->datum,

            'kategorija' => Kategorija::find($this->resource->kategorija\_id),

            'mesto' => Mesto::find($this->resource->mesto\_id),

            'slika' => $this->resource->slika,

            'omiljen' => $this->resource->omiljen,

        ];

    }

}

<?php

namespace App\Http\Controllers;

use App\Http\Resources\DogadjajResource;

use App\Models\Dogadjaj;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Validator;

class DogadjajController extends Controller

{

    /\*\*

     \* Display a listing of the resource.

     \*

     \* @return \Illuminate\Http\Response

     \*/

    public function index()

    {

        return DogadjajResource::collection(Dogadjaj::all());

    }

    /\*\*

     \* Show the form for creating a new resource.

     \*

     \* @return \Illuminate\Http\Response

     \*/

    public function create()

    {

        //

    }

    /\*\*

     \* Store a newly created resource in storage.

     \*

     \* @param  \Illuminate\Http\Request  $request

     \* @return \Illuminate\Http\Response

     \*/

    public function store(Request $request)

    {

        $validator = Validator::make($request->all(), [

            'naziv' => 'required|string|max:100',

            'vreme' => 'required|string',

            'datum' => 'required|string',

            'kategorija\_id'=>'required|integer|exists:kategorijas,id',

            'mesto\_id'=>'required|integer|exists:mestos,id',

            'slika' => 'required|string',

        ]);

        if ($validator->fails())

            return response()->json($validator->errors());

        $d = Dogadjaj::create([

            'naziv' => $request->naziv,

            'vreme' => $request->vreme,

            'datum' => $request->datum,

            'omiljen' =>0,

            'kategorija\_id' => $request->kategorija\_id,

            'mesto\_id' => $request->mesto\_id,

            'slika' => $request->slika,

        ]);

        $d->save();

        return response()->json(['Dogadjaj kreiran!', $d]);

    }

    /\*\*

     \* Display the specified resource.

     \*

     \* @param  \App\Models\Dogadjaj  $dogadjaj

     \* @return \Illuminate\Http\Response

     \*/

    public function show($id)

    {

        return new DogadjajResource(Dogadjaj::find($id));

    }

    /\*\*

     \* Show the form for editing the specified resource.

     \*

     \* @param  \App\Models\Dogadjaj  $dogadjaj

     \* @return \Illuminate\Http\Response

     \*/

    public function edit(Dogadjaj $dogadjaj)

    {

        //

    }

    /\*\*

     \* Update the specified resource in storage.

     \*

     \* @param  \Illuminate\Http\Request  $request

     \* @param  \App\Models\Dogadjaj  $dogadjaj

     \* @return \Illuminate\Http\Response

     \*/

    public function update(Request $request, $id)

    {

        $validator = Validator::make($request->all(), [

            'naziv' => 'required|string|max:100',

            'vreme' => 'required|string',

            'datum' => 'required|string',

            'kategorija\_id'=>'required|integer|exists:kategorijas,id',

            'mesto\_id'=>'required|integer|exists:mestos,id',

            'slika' => 'required|string',

        ]);

        if ($validator->fails())

            return response()->json($validator->errors());

        $d = Dogadjaj::find($id);

        if( $d){

            $d->naziv=$request->naziv;

            $d->vreme=$request->vreme;

            $d->datum=$request->datum;

            $d->kategorija\_id=$request->kategorija\_id;

            $d->mesto\_id=$request->mesto\_id;

            $d->slika=$request->slika;

            $d->save();

            return response()->json(['Uspesno izmenjeno!',  $d]);

        }else{

            return response()->json('Trazeni objekat ne postoji u bazi');

        }

    }

    /\*\*

     \* Remove the specified resource from storage.

     \*

     \* @param  \App\Models\Dogadjaj  $dogadjaj

     \* @return \Illuminate\Http\Response

     \*/

    public function destroy($id)

    {

        $d = Dogadjaj::find($id);

        if($d){

            $d->delete();

            return response()->json(['Uspesno obrisano!', $d]);

        }

       return response()->json('Trazeni objekat ne postoji u bazi');

    }

}

<?php

use App\Http\Controllers\AutfController;

use App\Http\Controllers\DogadjajController;

use App\Http\Controllers\KartaController;

use App\Http\Controllers\MestoController;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Route;

/\*

|--------------------------------------------------------------------------

| API Routes

|--------------------------------------------------------------------------

|

| Here is where you can register API routes for your application. These

| routes are loaded by the RouteServiceProvider within a group which

| is assigned the "api" middleware group. Enjoy building your API!

|

\*/

 Route::get("/dogadjaji",[DogadjajController::class,'index']);

 Route::get("/dogadjaji/{id}",[DogadjajController::class,'show']);

 Route::post("/register",[AutfController::class,'register']);

 Route::post("/login",[AutfController::class,'login']);

 Route::group(['middleware' => ['auth:sanctum']], function () {

    Route::get("/karte",[KartaController::class,'index']);

    Route::post("/karte",[KartaController::class,'store']);

    Route::get("/mesto",[MestoController::class,'index']);

    Route::get("/mesto/{id}",[MestoController::class,'show']);

    Route::post("/logout",[AutfController::class,'logout']);

});

Route::middleware(['auth:sanctum','isAPIAdmin'])->group(function(){ //ako je ulogovan admin

    Route::post("/dogadjaji",[DogadjajController::class,'store']);

    Route::put("/dogadjaji/{id}",[DogadjajController::class,'update']);

    Route::delete("/dogadjaji/{id}",[DogadjajController::class,'destroy']);

    Route::post("/logout",[AutfController::class,'logout']);

});

import React, { useEffect } from 'react';

import ReactDOM from 'react-dom';

import './App.css';

import Navbar from './Navbar';

import { BrowserRouter, Route, Routes } from 'react-router-dom';

import Footer from './Footer';

import Kontakt from './Kontakt';

import { useState } from 'react';

import Dogadjaji from './Dogadjaji';

import AdminPocetna from './AdminPocetna';

import Omiljeni from './Omiljeni';

import Login from './Login';

import axios from 'axios';

import DogadjajiRegion from './DogadjajiRegion';

import Dodaj from './Dodaj';

import Izmeni from './Izmeni';

import Statistike from './Statistike';

const axiosInstance = axios.create({

  baseURL: process.env.REACT\_APP\_API\_URL,

});

function Example() {

    const[token,setToken] = useState();

    const [izmenaID, setIzmenaID] = useState(0);

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

  const [karte,setKarte] = useState([ ])

  useEffect(() => {

    const getDogadjaji = async () => {

      try {

        const res = await axiosInstance.get( "http://127.0.0.1:8000/api/dogadjaji",

          {

            headers:{'Authorization': `Bearer ${ window.sessionStorage.getItem('auth\_token')}`},

          }

        );

        setDogadjaji(res.data.data);

        console.log(res.data.data)

      } catch (err) {

        console.log(err);

      }

    };

    getDogadjaji();

  }, [ axiosInstance]);

  useEffect(() => {

    const getKarte = async () => {

      try {

        const res = await axiosInstance.get( "http://127.0.0.1:8000/api/karte",

          {

            headers:{'Authorization': `Bearer ${ window.sessionStorage.getItem('auth\_token')}`},

          }

        );

        setKarte(res.data.data);

        console.log(res.data.data)

      } catch (err) {

        console.log(err);

      }

    };

    getKarte();

  }, [ axiosInstance]);

  function addToken(auth\_token){

    setToken(auth\_token);

  }

  function obrisi(id){

    axios

    .delete("http://127.0.0.1:8000/api/dogadjaji/"+id,{headers:{'Authorization': `Bearer ${ window.sessionStorage.getItem('auth\_token')}`} } )

    .then((res)=>{

        console.log(res.data);

        const token = window.sessionStorage.getItem('auth\_token');d

        window.sessionStorage.set('auth\_token',token);

    })

    .catch(function (error) {

        if (error.response) {

          // Request made and server responded

          console.log(error.response.data);

          console.log(error.response.status);

          console.log(error.response.headers);

        } else if (error.request) {

          // The request was made but no response was received

          console.log(error.request);

        } else {

          // Something happened in setting up the request that triggered an Error

          console.log('Error', error.message);

        }

      });

  }

  const[dogadjaj,setDogadjaj] = useState();

  function postaviIDZaIzmenu(id){ //fja koja vraca podatke o dogadjaju koji azuriramo

    setIzmenaID(id);

       let event = dogadjaji.filter((d)=>d.id==id);

       setDogadjaj(event[0])

  }

  return (

    <div className="App">

       <BrowserRouter  >

      <Navbar token={token} ></Navbar>

      <Routes>

        <Route path="/" element={<Login addToken={addToken}></Login>}></Route>

        <Route path="/dogadjaji/balkan" element={<DogadjajiRegion ></DogadjajiRegion>}></Route>

        <Route path="/dogadjaji/moji" element={<Omiljeni karte={karte} ></Omiljeni>}></Route>

        <Route path="/dogadjaji" element={<Dogadjaji dogadjaji={dogadjaji}  ></Dogadjaji>}></Route>

        <Route path="/kontakt" element={<Kontakt></Kontakt>}></Route>

        <Route path="/admin/stats" element={<Statistike dogadjaji={dogadjaji} karte={karte}></Statistike>}></Route>

        <Route path="/admin/izmeni" element={<Izmeni id={izmenaID} dogadjaj={dogadjaj}></Izmeni>}></Route>

        <Route path="/admin/dodaj" element={<Dodaj></Dodaj>}></Route>

        <Route path="/admin" element={<AdminPocetna dogadjaji={dogadjaji} obrisi={obrisi} setIzmeniID={postaviIDZaIzmenu}></AdminPocetna>}></Route>

      </Routes>

        <Footer></Footer>

  </BrowserRouter>

    </div>

  );

}

export default Example;

if (document.getElementById('example')) {

    ReactDOM.render(<Example />, document.getElementById('example'));

}

import { BsFillTrashFill, BsPencilFill } from 'react-icons/bs';

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

function AdminPocetna({dogadjaji,obrisi,setIzmeniID}) {

  let navigate = useNavigate();

  function azuriraj(id){

      setIzmeniID(id);

       navigate("/admin/izmeni/");

  }

    return (

      <div className='adminPocetna'>

        <table id="dtBasicExample" className="table table-striped table-bordered table-sm" cellSpacing="0" width="100%">

          <thead>

            <tr>

              <th className="th-sm">ID

              </th>

              <th className="th-sm">Naziv

              </th>

              <th className="th-sm">Vreme

              </th>

              <th className="th-sm">Datum

              </th>

              <th className="th-sm">Kategorija

              </th>

              <th className="th-sm">Mesto

              </th>

              <th className="th-sm">Kapacitet

              </th>

              <th className="th-sm">Opcije

              </th>

            </tr>

          </thead>

          <tbody>

              {dogadjaji.map((d)=>(<tr key={d.id}><td>{d.id}</td><td>{d.naziv}</td><td>{d.vreme}</td><td>{d.datum}</td><td>{d.kategorija.naziv}</td><td>{d.mesto.naziv\_mesta}</td><td>{d.mesto.kapacitet}</td><td><button className="btn" onClick={() => obrisi(d.id)}><BsFillTrashFill></BsFillTrashFill></button><button className="btn" onClick={() => azuriraj(d.id)}><BsPencilFill></BsPencilFill></button></td></tr>))}

            </tbody>

      </table>

      </div>

    );

  }

  export default AdminPocetna;

import { BsFillTrashFill, BsPencilFill } from 'react-icons/bs';

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

import BarChart from './BarChart';

function Statistike({dogadjaji,karte}) {

    return (

      <div className="pocetnaAdmin">

          <div className='float-container'>

                <div className="float-child1">

                      UKUPAN BROJ DOGADJAJA:

                      {dogadjaji.length}

                </div>

                <div className="float-child2">

                      UKUPAN BROJ PRODATIH KARATA:

                      {karte.length}

                </div>

          </div>

          <div className='float-container'>

              <BarChart karte={karte} dogadjaji={dogadjaji}></BarChart>

          </div>

      </div>

    );

  }

  export default Statistike; import React from "react";

import ReactDOM from "react-dom";

// Set up any imports needed for react-vis

import {

  XYPlot,

  HorizontalGridLines,

  XAxis,

  YAxis,

  VerticalBarSeries,

} from "react-vis";

function BarChart({karte,dogadjaji}) {

  const BarSeries = VerticalBarSeries;

  function brojKupovina(id){

    var brojac=0;

    for(var i=0;i<karte.length;i++){

        if(karte[i].dogadjaj.id==id){

            brojac++;

        }

    }

    return brojac;

  }

    const data = [];

    for(var i=0;i<dogadjaji.length;i++){

        var pom  =  brojKupovina(dogadjaji[i].id);

        if(pom!=0){

          data.push({x: dogadjaji[i].naziv,y:pom})

        }

    }

    for(var i=0;i<data.length;i++){

        for(var j=0;j<data.length;j++){

            if(data[i].y>data[j].y){

                var pom = data[i];

                data[i]=data[j];

                data[j]=pom;

            }

        }

    }

  return (

    <div className="App">

      <XYPlot

        yDomain={[0,100]}

        animation

        xType="ordinal"

        width={1000}

        height={400}

        xDistance={30}

      >

        <HorizontalGridLines />

        <XAxis title="Dogadjaj"/>

        <YAxis  title="Broj prodatih karata"/>

        <BarSeries className="horizontal-bar-series-example" data={data} />

      </XYPlot>

    </div>

  );

}

export default BarChart;

import { useState } from "react";

import DogadjajKartica from "./DogadjajKartica";

 function Dogadjaji({dogadjaji}) {

    return (

        <div>

            <div className="all-events">

               { dogadjaji .map((d) => ( <DogadjajKartica dogadjaj={d} key={d.id}  />  ))}

        </div>

      </div>

    );

  }

  export default Dogadjaji;

 import { BiTime ,BiCalendar} from 'react-icons/bi';

 import { BiCategory } from 'react-icons/bi';

 function DogadjajKartica({dogadjaj}) {

    function dodaj(id){ //kreira novu kupovinu

        axios

        .post("http://127.0.0.1:8000/api/karte/?dogadjaj\_id="+id+"&user\_id="+window.sessionStorage.getItem('auth\_id'),{headers:{'Authorization': `Bearer ${ window.sessionStorage.getItem('auth\_token')}`} } )

        .then((res)=>{

            console.log(res.data);

             alert("USPESNO")

        })

        .catch(function (error) {

            if (error.response) {

              // Request made and server responded

              console.log(error.response.data);

              console.log(error.response.status);

              console.log(error.response.headers);

            } else if (error.request) {

              // The request was made but no response was received

              console.log(error.request);

            } else {

              // Something happened in setting up the request that triggered an Error

              console.log('Error', error.message);

            }

          });

    }

     return (

         <div className="card">

         <div className="card-header" >

              <img className='card-img-top'  src ={dogadjaj.slika}   />

         </div>

         <div className="card-body">

         <h4 className = "naslovKartice">  {dogadjaj.naziv}   </h4>

             <div className="tag tag-teal"> <BiTime></BiTime>{dogadjaj.vreme}    </div>  <br />

             <div className="tag tag-teal1"> <BiCalendar></BiCalendar>{dogadjaj.datum}    </div>  <br />

             <p className='cena'>  {dogadjaj.mesto.naziv\_mesta}  </p>

             <div className="tag tag-teal2"> <BiCategory></BiCategory>{dogadjaj.kategorija.naziv}    </div>

             <button className="sortbtn btn" onClick={()=>dodaj(dogadjaj.id)}>Kupi kartu</button>

         </div>

     </div>

     );

   }

   export default DogadjajKartica;

# GitHub link ka repozitorijumu

https://github.com/IvonaLy/iteh-seminarski

# Reference

* Vlajić S, „Projektovanje softvera“, FON, 2020.
* Lazarević B, Marjanović Z, Aničić N, Babarogić S, „Baze podataka“, FON.
* PHP skripta, Laravel – konačna skripta, React script
* <https://reactjs.org/blog/2013/06/05/why-react.html>
* <https://www.php.net/manual/en/intro-whatis.php>
* <https://www.w3.org/standards/webdesign/htmlcss>
* <https://www.mysql.com/>