RESTful web aplikacija za rezervaciju aktivnosti

Buršić, Luka

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Prijediplomski studij računarstva

Završni rad

RESTful web aplikacija za rezervaciju aktivnosti

Rijeka, srpanj 2023.

Luka Buršić
0069089451

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Prijediplomski studij računarstva

Završni rad

RESTful web aplikacija za rezervaciju aktivnosti

Mentor: doc.dr.sc. Marko Gulić

Rijeka, srpanj 2023.

Luka Buršić
0069089451

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA ZAVRŠNE ISPITE

Rijeka, 8. ožujka 2023.

Zavod: **Zavod za računarstvo**
Predmet: **Razvoj web aplikacija**
Grana: **2.09.06 programsko inženjerstvo**

ZADATAK ZA ZAVRŠNI RAD

Pristupnik: **Luka Buršić (0069089451)**
Studij: Sveučilišni prijediplomski studij računarstva

Zadatak: **RESTful web aplikacija za rezervaciju aktivnosti / RESTful web application for booking activities**

Opis zadatka:

Razviti RESTful web aplikaciju za rezervaciju dostupnih aktivnosti u studentskom domu. Aplikacija mora podržavati uslugu rezervacije termina na igralištu, teretani i praonici za rublje. Također, treba implementirati popis rezerviranih aktivnosti kao i njihovo moguće ažuriranje ili brisanje od strane korisnika koji ih je rezervirao. Za razvoj poslužiteljskog dijela web aplikacije treba koristiti Laravel radni okvir uz proizvoljno odabran sustav za upravljanje bazama podataka. Također, treba koristiti jedan od Laravel paketa za realizaciju autentifikacije RESTful aplikacije. Za razvoj klijentskog dijela aplikacije treba koristiti React JavaScript knjižicu za razvoj korisničkog sučelja uz učinkovito renderiranje aplikacije na uređajima s različitim veličinama zaslona.

Rad mora biti napisan prema Uputama za pisanje diplomskih / završnih radova koje su objavljene na mrežnim stranicama studija.

Zadatak uručen pristupniku: 26. ožujka 2023.

Mentor:

Doc. dr. sc. Marko Gulić

Predsjednik povjerenstva za završni ispit:

Prof. dr. sc. Miroslav Joler

Izjava o samostalnoj izradi rada

Izjavljujem da sam samostalno izradio ovaj rad.

Rijeka, srpanj 2023.

_____
Ime Prezime

Zahvala

Zahvaljujem mentoru, doc.dr.sc. Marku Guliću na podršci tijekom pisanja ovoga rada i korisnim raspravama i savjetima. Zahvaljujem svojoj obitelji i prijateljima na podršci tijekom studiranja.

Sadržaj

Popis slika

Poglavlje 1

Uvod

Studentsko naselje 'Ivan Goran Kovačić' je studentsko naselje u Rijeci koje nudi smještaj za preko 600 studenata, a osim smještaja pruža i razne druge usluge. Osim standardnih usluga kao što su menza i studentski servis, studentima se omogućuje korištenje teretane, praone rublja te igrališta. Za teretanu i praonu potrebno je na recepciji unaprijed rezervirati termin, zbog čega se nerijetko zna stvoriti gužva na recepciji. Također, studenti često moraju doći do recepcije da bi rezervirali termin za određeni dan, samo da bi onda bili obaviješteni kako su za taj dan termini već popunjeni. Upravo su to neki od razloga stvaranja ove aplikacije, kojom se digitalizira proces rezervacije termina. Osim rezervacije, moguć je uvid u popis rezerviranih termina. Za termine na igralištu stvorena je platforma koja korisnicima omogućuje kreiranje vlastitih termina, ali i prijavu na termine koje su kreirali drugi korisnici. Aplikacija je, osim na hrvatskom, napravljena i na engleskom jeziku, što omogućava i stranim studentima koji žive u domu da koriste aplikaciju.

Za razvoj poslužiteljskog dijela aplikacije korišten je **Laravel**, poznati PHP okvir koji nudi elegantnu sintaksu, bogat skup značajki, kao što su usmjeravanje, komunikcaija s bazom podataka i autentifikacija, te olakšava razvoj aplikacija. Laravel također ima veliku zajednicu korisnika i obilje resursa, što olakšava učenje i rješavanje problema.

Klijentski dio aplikacije razvijen je pomoću **Reacta**, Javascript biblioteke otvorenog koda koja nudi komponentnu arhitekturu koja olakšava izgradnju modularnih

i ponovno upotrebljivih korisničkih sučelja. React također koristi virtualni DOM (*document object model*) za učinkovito ažuriranje samo promijenjenih dijelova korisničkog sučelja, što rezultira brzim i odzivnim korisničkim iskustvom.

Za pristup i upravljanje podacima koristi se **MySQL**, široko korišteni sustav za upravljanje relacijskim bazama podataka. MySQL ima snažne performanse, pouzdanost i skalabilnost, što ga čini odličnim izborom za aplikacije koje zahtijevaju obradu velikog broja podataka. Također, podržava standardni SQL jezik i ima široku podršku i resurse u zajednici, olakšavajući razvoj i održavanje aplikacija.

Poglavlje 2

Korištene tehnologije i alati

2.1 Laravel

Laravel je besplatan PHP okvir namijenjen razvoju web aplikacija prateći MVC (*model - view - controller*) arhitekturu. Prva verzija Laravela postala je dostupna 2011. godine, a najnovija dostupna verzija jest Laravel 10. [1] Prema djelovanju, MVC arhitektura dijeli se u 3 cjeline: modeli, pogledi i upravljači. Takva podjela omogućuje jednostavnije razumijevanje odnosa između dijelova poslužiteljske strane aplikacije, veću modularnost, a samim time i jednostavnije održavanje. Modeli omogućavaju upravljanje podacima te komunikaciju s bazom podataka te se pomoću njih definira koji podaci se koriste u aplikaciji. Pomoću pogleda se definira kako će se ti podaci prikazati te se omogućuje komunikacija s korisnikom. Upravljače možemo definirati kao poveznicu između modela i pogleda - oni prosljeđuju podatke između modela i pogleda te ih istovremeno obrađuju pomoću definiranih metoda.

U ovoj aplikaciji, MVC arhitektura nije u potpunosti implementirana u Laravelu, već se za prikaz podataka i komunikaciju s korisnikom koristi React. React omogućuje kreiranje kuno dinamičnijih komponenti u odnosu na Blade predloške, koji se inače koriste kod implementacije pogleda unutar Laravela. React i Laravel se izvršavaju na različitim TCP portovima, stoga je potrebno definirati API kako bi se omogućila razmjena podataka između njih.

2.1.1 Laravel Sanctum

Laravel Sanctum je alat koji programerima pruža jednostavan sustav autentikacije za jednostranične aplikacije, mobilne aplikacije te API-jeve bazirane na tokenima. [2] U ovoj aplikaciji, pomoću Laravel Sanctuma je kreirana SPA autentikacija, odnosno autentikacija jednostranične aplikacije. Kada se korisnik uspješno autenticira, dodjeljuje mu se *Bearer* token koji mu omogućuje da preko API-a komunicira s poslužiteljem. Da bi se podaci mogli dohvatiti na klijentskoj strani, potrebno je u zaglavlju zahtjeva poslati taj token. Nedostatak korištenja ovih tokena je to što ništa ne sprječava aplikacije i korisnike treće strane da koriste te tokene, ako ih uspiju pribaviti. To može rezultirati neovlaštenim pristupom povjerljivim ili zaštićenim podacima, odnosno CSRF (*Cross-Site Request Forgery*) napadima.

Iz tog razloga se, osim *Bearer* tokena, korisniku dodjeljuje i CSRF token. CSRF token je tajna, jedinstvena i nepredvidiva vrijednost koju poslužitelj generira i dodjeljuje korisniku kako bi se zaštitio od CSRF napada. Taj se token na klijentskoj strani najčešće pohranjuje u obliku kolačića.

2.1.2 Komunikacija s bazom

Migracije

Nakon uspostavljanja veze s bazom podataka, što je kasnije prikazano na slici 2.11, potrebno je odrediti koje će se tablice nalaziti u bazi te koje će podatke svaka tablica sadržavati. Za upravljanje strukturom baze u Laravelu se koriste migracije. Migracije omogućuju dodavanje i brisanje tablica i stupaca, mijenjanje tipova podataka te postavljanje ograničenja za podatke. Migracije se mogu kreirati korištenjem *php artisan make:migration* unutar *artisan* konzole, kao u primjeru na slici 2.1a. Izvršavanjem te naredbe, kreira se migracijska datoteka s osnovnim izgledom, kao što je prikazano na slici 2.1b. Nakon toga, programer može definirati podatke koji će se nalaziti u tablici, njihove tipove i ograničenja. Da bi se tablica kreirala u bazi, potrebno je izvršiti naredbu *php artisan:migrate*.

```
php artisan make:migration create_example_table
```

(a) Naredba za kreiranje migracije

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('example', function (Blueprint $table) {
            $table->id();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('example');
    }
};
```

(b) Osnovni izgled migracije

Slika 2.1 Primjer kreiranja migracije.

Eloquent

Laravel uključuje Eloquent, objektno-relacijski maper (ORM) koji olakšava interakciju s bazom podataka. Kod korištenja Eloquenta, svaka tablica ima odgovarajući model koji omogućava interakciju s tablicom. U svakom modelu potrebno je definirati ime tablice s kojom je taj model povezan, a mogu se definirati i druga svojstva, na primjer koja polja se mogu popuniti, koja su zaštićena te koja predstavljaju primarni ključ. Također, u modelu se mogu definirati tipovi veza između tablica te polja preko kojih su tablice povezane, kao što je prikazano na slici 2.2. Eloquent pojednostavljuje izvršavanje SQL upita pružajući metode kojima se mogu dodavati, brisati te ažurirati podaci u tablici.

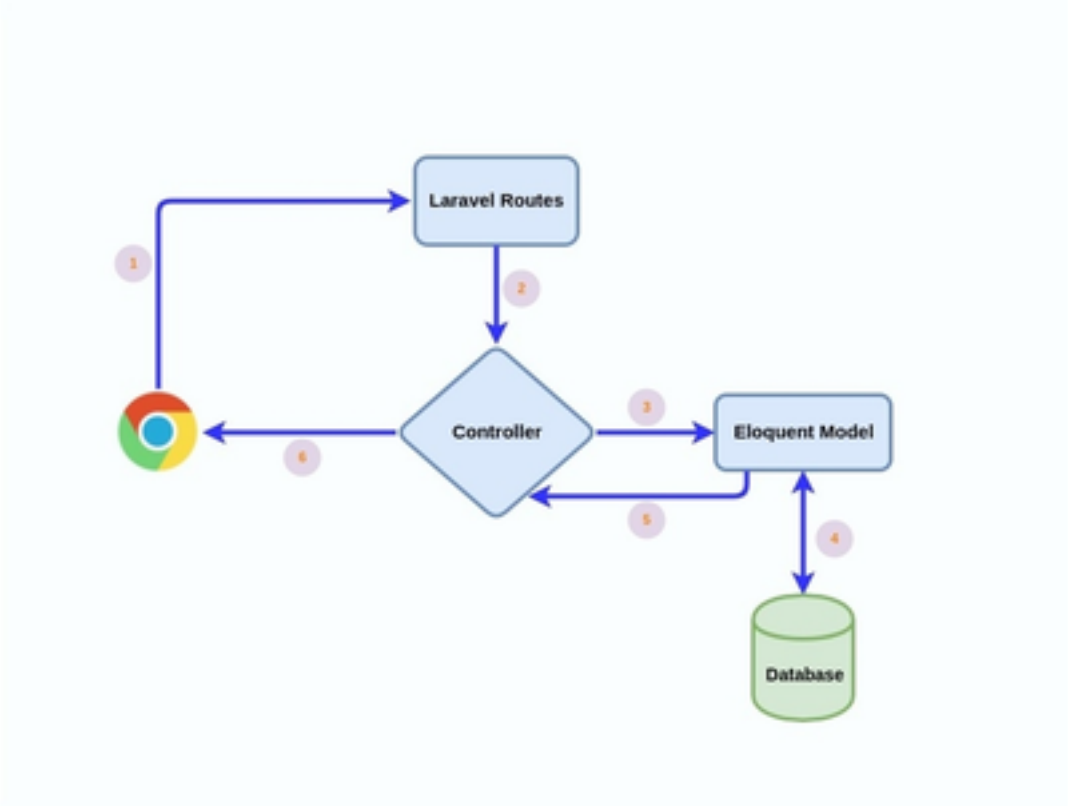
```php
class Tweet extends Model
{

    // override the id primary key

    protected $primaryKey = 'tweet_id';

    /**
     * The database table used by the model.
     *
     * @var string
     */
    protected $table = 'tweets';

    public $timestamps = false;

    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [];

    public function user()
    {
        return $this->belongsTo('App\User', 'twitter_id', 'twitter_id');
    }

}
```

Slika 2.2 Primjer modela s definiranim vezama (izvor: [3]).

2.1.3 Usmjeravanje

U Laravelu, svi zahtjevi su mapirani pomoću ruta. To znači da se za svaki zahtjev, na temelju adrese na koju je poslan i vrste zahtjeva, pronalazi odgovarajuća metoda određenog upravljača, koja je definirana za tu adresu. U toj metodi se zahtjev obrađuje na odgovarajući način te se po potrebi izvršavaju SQL upiti koristeći modele. Nakon izvršenog upita, rezultat upita se vraća upravljaču, koji te podatke obrađuje te prosljeđuje na klijentsku stranu. Tijek obrade zahtjeva prikazan je na slici 2.3.

Rute se mogu definirati u *web.php* ili *api.php* datoteci, ovisno o tipu aplikacije - u ovoj se aplikaciji koristi *api.php*. U većini slučajeva, priroda aplikacija zahtjeva da određene rute budu zaštićene, odnosno da im neautenticirani korisnici ne mogu pristupiti. Za zaštitu takvih ruta koristi se *Bearer* token, kojeg korisniku dodjeljuje Laravel Sanctum, principom opisanim u sekciji 2.1.1.



Slika 2.3 Tijek obrade HTTP zahtjeva (izvor: [4]).

Na slici 2.4 prikazan je primjer definiranja rute. Dakle, ako korisnik pristupi, tj. pošalje zahtjev na rutu */articles*, taj se zahtjev prosljeđuje metodi *index*, koja se nalazi unutar upravljača *ArticleController*. U toj se metodi primljeni zahtjev obrađuje na odgovarajući način te se po završetku obrade vraća odgovor klijentskoj strani aplikacije.

```
Route::get('articles', 'ArticleController@index');
```

Slika 2.4 Tijek obrade HTTP zahtjeva (izvor: [4]).

2.2 React

React je besplatna Javascript biblioteka otvorenog koda koja se koristi za kreiranje korisničkih sučelja temeljenih na komponentama. Komponente omogućuju podjelu korisničkog sučelja na neovisne, ponovno upotrebljive dijelove. Komponente mogu primati određene ulazne podatke (*props*) na temelju kojih se mogu mijenjati svojstva komponente, a vraćaju odgovarajuće elemente koji će se prikazati korisniku.

React je deklarativan jezik. Općenito, deklarativnost znači da se prilikom pisanja programa definira što se treba izvršiti, a ne kako se to treba izvršiti. U kontekstu Reacta, deklarativnost se odnosi na to da programer definira React komponente te tako deklarira željeno stanje korisničkog sučelja, a React na temelju toga ažurira DOM. [5]

React omogućava usmjeravanje na strani klijenta (*client side routing*). Kod klasičnih web stranica, prilikom bilo kakve promjene sadržaja na stranici, koliko god ona bila mala, preglednik ponovno šalje zahtjev poslužitelju za dohvaćanje stranice, uz pripadne CSS i JavaScript datoteke. Ovaj proces može biti mrežno zahtjevan te uzrokovati kašnjenja u prikazu sadržaja. S druge strane, kada se koristi React, ne mora se cijela stranica ponovno učitavati, već samo ona komponenta kod koje je došlo do promjene sadržaja. To se postiže korištenjem virtualnog DOM-a, u kojem se pomoću Javascript objekata pohranjuju kopije svih elemenata u stvarnom DOM-u. Manipuliranje virtualnog DOM-a je puno brže od manipuliranja "običnog" DOM-a zbog toga što se tijekom manipulacije virtualnog DOM-a promjene ne prikazuju na zaslonu. Dakle, kada se dogodi nekakva promjena na stranici, virtualni DOM se u potpunosti ažurira. Zatim se novo stanje virtualnog DOM-a uspoređuje sa stanjem prije promjene. Procesom zvanim *diffing*, React utvrđuje koji objekti u DOM-u su se promijenili. React nakon toga u DOM-u ažurira isključivo one objekte u kojim je došlo do promjena, te se zatim te promjene prikazuju na zaslonu. [6]

Također, još jedan faktor koji utječe na efikasnije osvježavanje sadržaja jest taj da se komponente nalaze na strani klijenta pa nije potrebno slati zahtjeve poslužitelju.

2.2.1 React-i18next

React-i18next je popularna React knjižnica koja se koristi za lokalizaciju web aplikacija, odnosno omogućuje prilagodbu aplikacije različitim jezicima. Osim prijevoda riječi, ovim alatom mogu se izvršiti i druge metode lokalizacije, kao što su formatiranje datuma i brojeva. Na slici 2.5 prikazano je kako se unutar *i18n.js* datoteke postavlja ovaj alat.

```
i18n
  .use(LanguageDetector)
  .use(initReactI18next)
  .use(Backend)
  .init({
    debug: true,
    fallbackLng: 'hr'
  });
```

Slika 2.5 Postavljanje i18n alata.

Ova aplikacija podržava hrvatski i engleski jezik. Da bi to bilo moguće, kreirane su odgovarajuće datoteke unutar *public* direktorija. Unutar tih datoteka, definirani su ključevi koji označuju dio sadržaja koji se prikazuje korisniku te odgovarajući prijevodi za te ključeve. Također, moguće je kreiranje podključeva za određeni ključ, čime se postiže bolja organiziranost prijevoda te njihovo jednostavnije održavanje. Na slici 2.6 prikazano je na koji način se definiraju ključevi i odgovarajući prijevodi na engleski jezik.

```
"already-have-account": "Already have an account?",
"logout": "Logout",
"logout-success": "Logged out successfully!",
"gym": {
    "title": "Gym" ,
    "homepage": "Gym - homepage"
},
```

Slika 2.6 Primjer definiranja ključeva za prijevod na engleski jezik

Da bi se ti ključevi mogli koristiti u kodu, potrebno je uvesti funkciju *useTranslation()*, kao što je prikazano na slici 2.7a. Ta funkcija omogućava da se jednom linijom koda definira sadržaj za oba jezika. Pri pozivu te funkcije, potrebno je definirati ključ čiju vrijednost želimo dohvatiti. Odluka o tome koji jezik će se koristiti donosi se na temelju vrijednosti *i18nextLng*, zapisane unutar lokalne pohrane u korisnikovom pregledniku.

```
const { t } = useTranslation();
```

(a) Uvođenje funkcije *useTranslation()*.

```
<button onClick={handleSubmit}>{t('session.confirm')}</button>
```

(b) Primjer korištenja funkcije za definirani ključ.

Slika 2.7 Korištenje funkcije *useTranslation()*.

2.2.2 SweetAlert

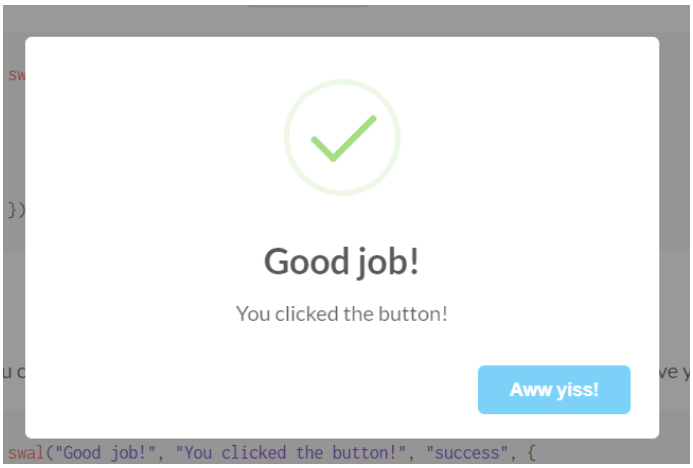
SweetAlert je knjižnica koja programerima omogućuje prilagodbu izgleda prozora za obavijesti (eng. *alert*) i kreiranje prozora koji će korisnicima biti vizualno privlačni. U ovom se projektu koristi SweetAlert 2.1.2 [7].

Prozori se kreiraju pomoću *swal()* funkcije. Pri pozivanju te funkcije, mogu se definirati razni parametri, među kojima su osnovni naslov, tekst unutar prozora, tip ikone te tekst koji će se prikazati unutar tipke za zatvaranje prozora. Na slici 2.8 prikazan je primjer pozivanja *swal()* funkcije s definiranim parametrima.

```
swal({
  title: "Good job!",
  text: "You clicked the button!",
  icon: "success",
  button: "Aww yiss!",
});
```

Slika 2.8 Primjer kreiranja prozora pomoću *swal()* funkcije (izvor: [7])

Pozicije elemenata unutar prozora su automatski postavljene, pa se programer ne treba brinuti oko toga, no moguće ih je promijeniti ukoliko je to potrebno. Također, neki od parametara imaju postavljene zadane vrijednosti, pa ih nije nužno definirati. Na slici 2.9 vidljiv je prozor kreiran kodom sa slike 2.8.



Slika 2.9 Prozor kreiran *swal()* funkcijom (izvor: [7])

2.2.3 Material UI

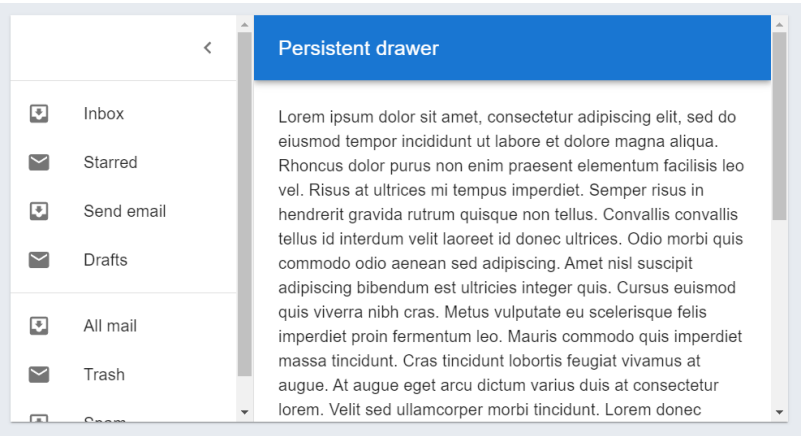
Material UI (MUI) je React knjižnica otvorenog koda koja implementira Googleov Material Design, sustav smjernica, komponenti i alata koji podržava najbolje prakse dizajna korisničkog sučelja te olakšava suradnju između dizajnera i programera. [8]

MUI pruža programerima veliku kolekciju visoko funkcionalnih komponenti koje omogućavaju da aplikacija bude vizualno ugodna korisnicima, a implementacija tih komponenti vrlo je jednostavna. Također, programerima i dizajnerima se pruža mogućnost prilagodbe izgleda tih komponenti, ovisno o njihovim potrebama. Još jedna prednost ovog alata jest to da je zajednica koja ga koristi vrlo velika, što garantira njegovo stalno unaprijeđivanje i držanje koraka s najnovijim trendovima i tehnologijama. [9]

Jedna od često korištenih MUI komponenti jest *Drawer* komponenta, koja korisniku omogućava navigaciju kroz aplikaciju. U tu se komponentu dodaju kartice ili

tipke koje korisnicima omogućavaju pristup određenim funkcionalnostima i dijelovima aplikacije. Ta se komponenta može nalaziti na lijevoj ili desnoj strani zaslona, a pri njenom vrhu nalazi se tipka kojom se ta komponenta otvara ili zatvara.

Na slici 2.10 prikazan je primjer *Drawer* komponente na lijevoj strani zaslona. Unutar te komponente nalazi se nekoliko kartica koje korisniku omogućuju pristup određenim funkcionalnostima, a pri vrhu se nalazi tipka za otvaranje i zatvaranje ove komponente.



Slika 2.10 Primjer korisničkog sučelja s *Drawer* komponentom (izvor: [10])

2.3 MySQL

MySQL je sustav za upravljanje relacijskim bazama podataka, čija je prva verzija objavljena 1995. godine. Kreatori ovog sustava, Allan Larson i Michael Widenius, razvili su ovaj sustava iz postojećeg mSQL sustava, kojeg su smatrali sporim i nefleksibilnim. Kreirano je novo SQL sučelje, dok je API koji se koristio u mSQL-u zadržan, čime se programerima omogućio jednostavniji prijelaz na novi sustav. [11]

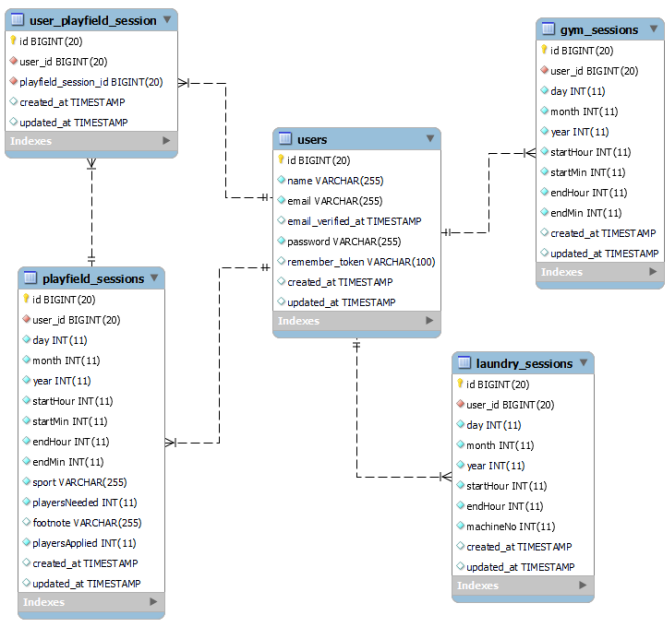
MySQL pruža jednostavan i intuitivan prikaz podataka u relacijskim tablicama. Relacijske tablice omogućuju organizaciju pohranjenih podataka na adekvatan način te definiranje veza između tablica i podataka u tablicama. Da bi se Laravel aplikacija

povezala s bazom, potrebno je konfigurirati *.env* datoteku te definirati parametre za povezivanje, kao što su IP adresa, port te ime baze, kao što je prikazano na slici 2.11.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=snIgkDb
DB_USERNAME=luka
DB_PASSWORD=password
```

Slika 2.11 Definiranje parametara povezivanja sa bazom.

Za provjeru i upravljanje podacima u bazi korišten je MySQL Workbench. Ovaj program programerima pruža opciju reverznog inženjeringa (eng. *reverse engineering*) koja, između ostalog, omogućava izradu E-R dijagrama. Na slici 2.12 prikazan je E-R dijagram ove aplikacije, na kojem se jasno mogu vidjeti koja polja se nalaze u tablicama te preko kojih ključeva su te tablice povezane.



Slika 2.12 E-R dijagram aplikacije

U tablici *users* pohranjeni su podaci za identifikaciju korisnika - ime, lozinka te email adresa. Tablica *gym_sessions* sadrži osnovne podatke kojim se definiraju datum (dan, mjesec, godina) i vrijeme termina u teretani(sati i minute kada termin započinje, odnosno završava). Tablica *laundry_sessions* za termine u praoni slična je prethodnoj, razlika je u tome što ova tablica sadržava i stupac u kojem se definira perilica koja se koristi tijekom termina. Također, u ovoj tablici nema stupaca kojim se definiraju minute početka i završetka jer termini počinju i završavaju na puni sat. Naposljetku, tablica *playfield_sessions* sadrži sve stupce kojim se definiraju datum i vrijeme termina na igralištu. Osim toga, u ovoj tablici postoje i stupci za definiranje sporta, određivanje očekivanog broja igrača (*playersNeeded*), za napomenu koju korisnik može navesti prilikom kreiranja termina (*footnote*), te stupac kojim se prati koliko korisnika je prijavljeno na termin (*playersApplied*).

Poglavlje 3

Opis aplikacije

U ovom poglavlju detaljno su opisane sve funkcionalnosti koje ova aplikacija pruža s fokusom na klijentski dio aplikacije.

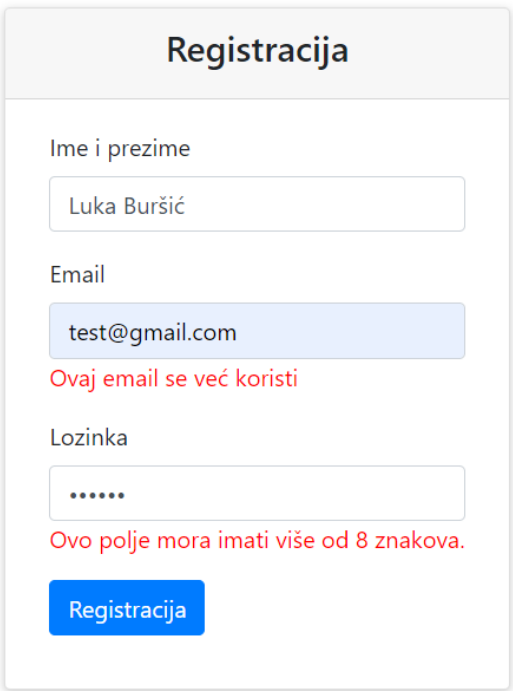
3.1 Autentikacija

Da bi korisnici mogli koristiti usluge ove aplikacije, prvo se moraju autenticirati. U Reactu su za tu svrhu kreirane komponente za registraciju i prijavu korisnika. Komponenta za registraciju sadrži jednostavan obrazac u kojem se od korisnika zahtijeva unos imena, email adrese te lozinke. Nakon što se unesu podaci, na poslužiteljskoj strani provjerava validnost podataka - je li email adresa jedinstvena te je li lozinka dovoljno dugačka. U slučaju krivo unesenih podataka, korisniku se prikazuje odgovarajuća poruka, kao što je prikazano na slici 3.1, a ako su podaci ispravni, prikazuje se SweetAlert prozor s potvrdnom porukom te se korisnik usmjerava na početnu stranicu.

Prijava korisnika funkcionira na sličan način - korisnici unose svoj email te lozinku, koji se zatim provjeravaju u bazi podataka aplikacije. Ukoliko uneseni podaci nisu ispravni, korisniku se prikazuje odgovarajuća poruka o pogrešci. U suprotnom, korisnik prima potvrdnu poruku te se usmjerava na početnu stranicu.

Ako je korisnik uspješno autenticiran, dodjeljuje mu se *Bearer* token, koji omogućava slanje zahtjeva i pristupanje poslužiteljskom dijelu aplikacije. Osim toga,

taj se token koristi za omogućavanje pristupa zaštićenim rutama i komponentama na klijentskoj strani aplikacije. Ako korisnik ima dodijeljen token, učitati će se odgovarajuća komponenta, a ako nema, korisnik će biti preusmjeren na stranicu za prijavu.



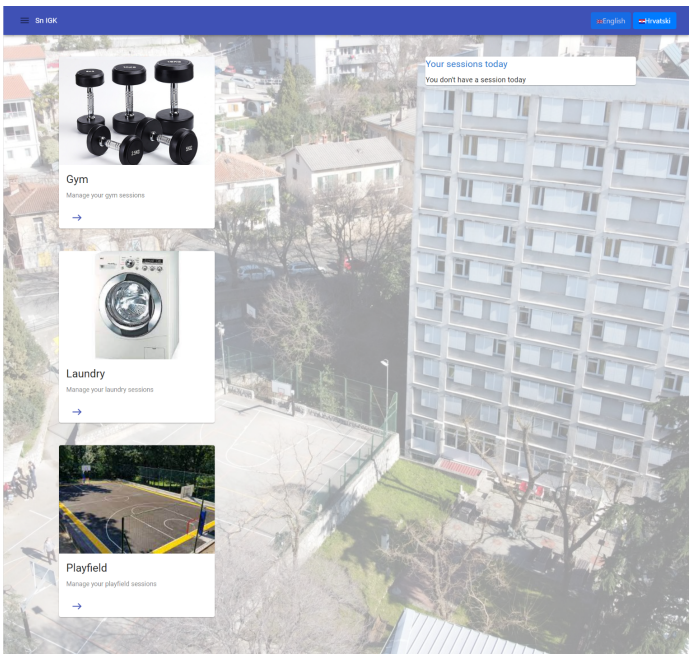
Slika 3.1 Primjer validacijskih poruka kod neispravno unesenih podataka

3.2 Lokalizacija

Kako je već prije napomenuto, u ovom domu žive i strani studenti, pa je jedan od ciljeva ove aplikacije bio da aplikacija bude namijenjena i njima. To je postignuto korištenjem React-i18next knjižnice, opisane u sekciji 2.2.1. Ovaj alat omogućuje jednostavan i brz prijevod sadržaja komponenti. Da bi prijevod funkcionirao, bilo je potrebno za oba jezika kreirati *translation.json* unutar *public* mape klijentskog dijela aplikacije te u njima za sve ključeve koji se koriste u komponentama definirati

odgovarajući tekst koji će se prikazati.

Na slici 3.2, prikazan je izgled početne stranice kada je jezik aplikacije postavljen na engleski. Tekstualni sadržaj je generiran pomoću ključeva koji su postavljeni na odgovarajuća mjesta unutar komponenti, što omogućuje definiranje sadržaja za oba jezika. Odabir jezika na kojem će se sadržaj prikazivati vrši se na temelju *i18nextLng* vrijednosti pohranjene u lokalnoj pohrani web preglednika. Promjena jezika sadržaja može se izvršiti pritiskom na gumbe za odabir jezika, koji se nalaze u gornjem desnom kutu.

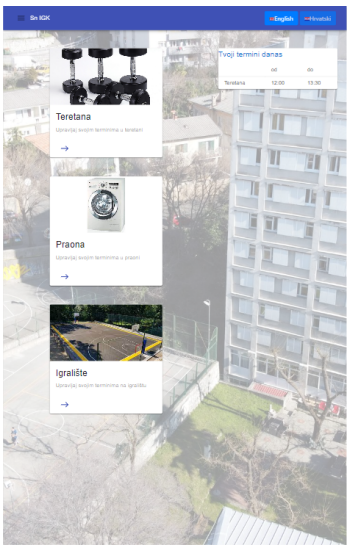


Slika 3.2 Izgled početne stranice na engleskom jeziku

3.3 Responzivnost

Kako studenti svakodnevno osim laptopa koriste i mobitele i tablete, bilo je potrebno dizajn aplikacije prilagoditi različitim dimenzijama zaslona. Drugim riječima, cilj je bio osigurati da elementi na stranicama dinamički prilagođavaju svoju poziciju i veličinu. Kroz implementaciju responzivnog dizajna, omogućeno je da se sadržaj i funkcionalnosti prilagode veličini zaslona, što olakšava preglednost i upotrebljivost

aplikacije na različitim uređajima. Na slici 3.3 prikazana je početna stranica na zaslonu tableta *Surface Pro 7*, na kojoj se može vidjeti kako elementi prilagođavaju svoju poziciju ovisno o zaslonu.

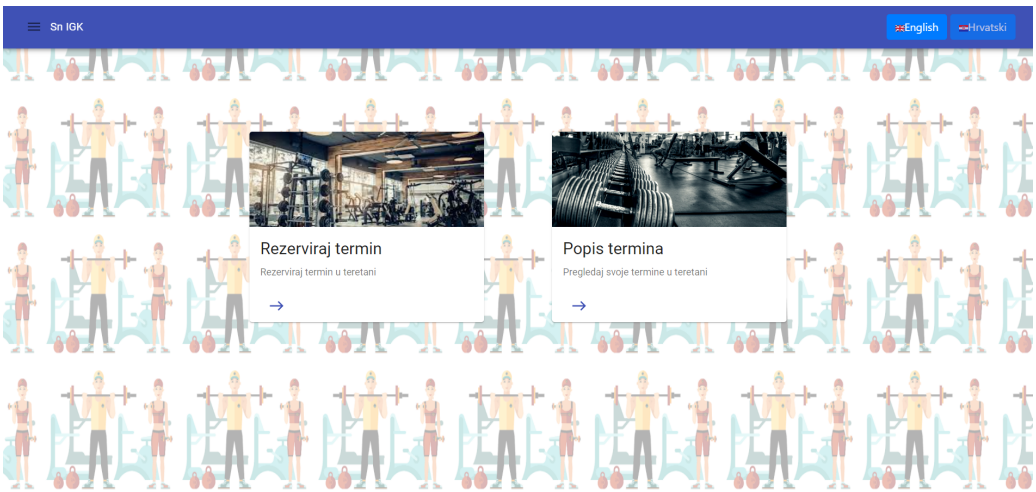


Slika 3.3 Izgled početne stranice na zaslonu *Surface Pro 7* (dimenzije 912x1368)

3.4 Teretana

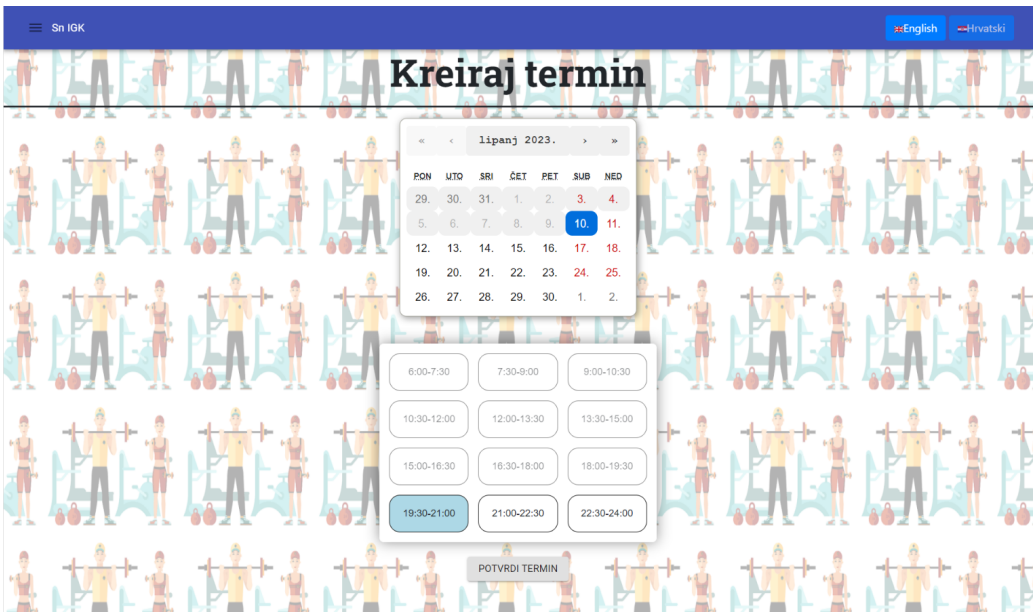
3.4.1 Rezervacija termina

Kada korisnik na početnoj stranici aplikacije odabere teretanu, preusmjerava se na početnu stranicu teretane, koja je prikazana na slici 3.4. Na toj se stranici nalaze dvije kartice - jedna za rezervaciju termina u teretani te druga za pregled svih budućih termina koje je korisnik rezervirao.



Slika 3.4 Izgled početne stranice za teretanu

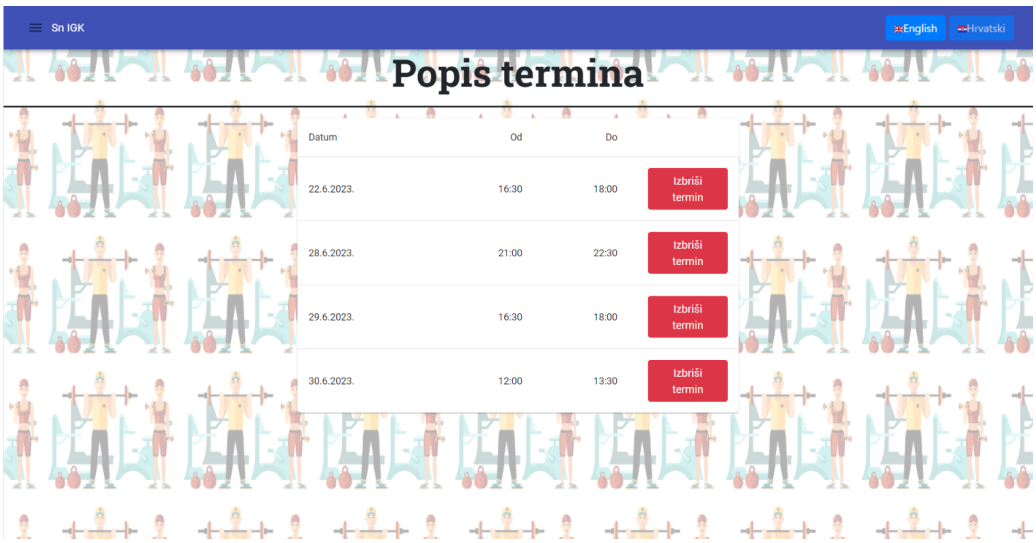
Ukoliko korisnik odabere rezervaciju termina, preusmjerava se na stranicu za definiranje detalja termina. Na toj se stranici od korisnika traži odabir datuma za koji želi rezervirati termin te odabir jednog od točno definiranih perioda iz tablice termina. Ukoliko se korisnik, kao u situaciji na slici 3.5, odluči odabrati termin za taj isti dan, na raspolaganju ima samo one termine koji još nisu započeli. Tek nakon što su definirani dan i period, korisniku se omogućava rezervacija. Nakon što korisnik pritisne gumb za rezervaciju, provjerava se mogućnost rezervacije odabranog termina. Ukoliko korisnik već ima taj dan rezerviran termin u teretani ili ima pet termina u budućnosti, njegov se zahtjev odbija. Također, u teretani istovremeno mogu biti maksimalno tri osobe. Stoga, ukoliko je odabrani termin već rezerviran od strane tri osobe, zahtjev se odbija. Ukoliko nijedan od ovih uvjeta nije narušen, izvršava se rezervacija te korisnik dobiv potvrdnu poruku.



Slika 3.5 Primjer rezerviranja termina za isti dan

3.4.2 Popis termina

Ukoliko korisnik na početnoj stranici za teretanu odabere opciju popisa termina, preusmjerava se na stranicu gdje se prikazuje tablica sa svim korisnikovim budućim terminima u teretani, kao što se može vidjeti na slici 3.6. U toj su tablici prikazani datum, početak te kraj termina, a korisniku se pruža i mogućnost brisanja termina. Prikazani termini sortirani su kronološki.



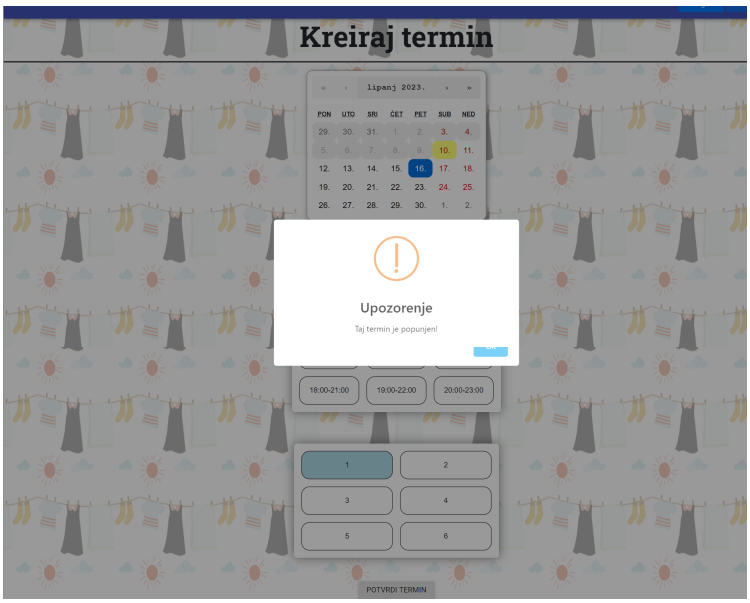
Slika 3.6 Prikaz korisnikovih termina u teretani

3.5 Praona

3.5.1 Rezervacija termina

Na početnoj stranici za praonu nalaze se, kao i kod teretane, kartice za rezervaciju i prikaz korisnikovih termina. Ukoliko korisnik odabere rezervaciju, dolazi na stranicu gdje može definirati detalje termina - datum, vrijeme te perilicu koju želi koristiti. Ponuđeni termini počinju na svaki puni sat između 6 i 20 sati te traju tri sata, a moguće je odabrati jednu od 6 ponuđenih perilica.

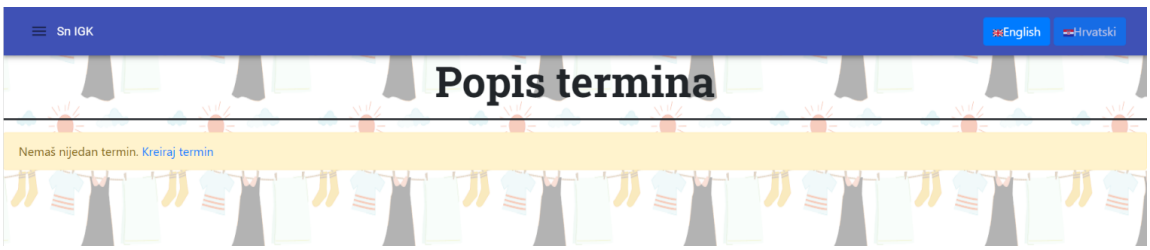
Nakon što korisnik definira sve detalje i pošalje zahtjev za rezervaciju, provjerava se nekoliko pravila koja određuju hoće li zahtjev biti uspješan. Ukoliko korisnik već ima pet rezerviranih termina u praoni ili ima termin taj dan, zahtjev će biti odbijen. Također, prilikom obrade tog zahtjeva, provjerava se je li odabrana perilica slobodna kroz sva tri sata za koja ju je korisnik odabrao. Na primjer, ukoliko student želi rezervirati perilicu od 12 do 15 sati, provjeravaju se svi termini koji se preklapaju s tim, počevši od onog koji traje od 10 do 13 sati pa sve do termina koji traje od 14 do 17 sati. Ukoliko postoji preklapanje, korisnikov zahtjev se odbija, kao što je prikazano na slici 3.7.



Slika 3.7 Poruka koju korisnik dobije kada postoji preklapanje

3.5.2 Popis termina

Kao i kod teretane, korisniku se i ovdje pruža uvid u sve buduće termine koje je rezervirao. Termini se nalaze u tablici u kojoj su prikazani datum, vrijeme termina te perilica koju je korisnik rezervirao. Ukoliko je ta tablica prazna, korisniku se prikazuje odgovarajuća poruka s poveznicom na stranica za rezervaciju termina, kao u primjeru na slici 3.8. Također, u svakom retku tablice nalazi se gumb koji korisniku omogućava brisanje termina.

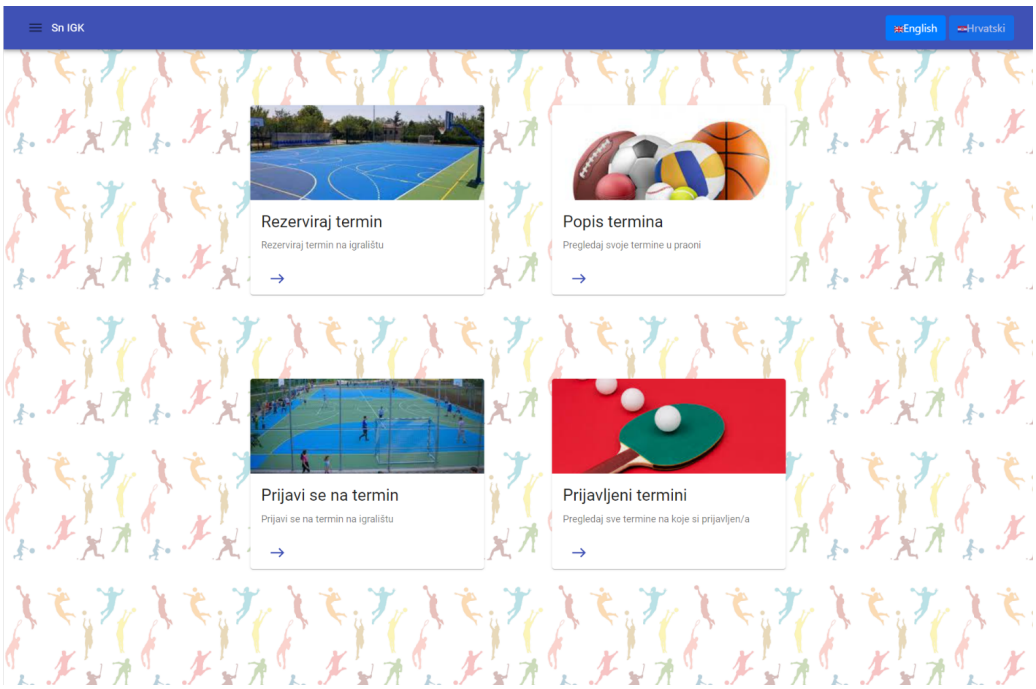


Slika 3.8 Poruka koju korisnik dobije kada postoji preklapanje s drugim terminima

3.6 Igralište

Iz svega do sada napisanog, može se vidjeti kako su funkcionalnosti vezane za teretanu i praonu međusobno veoma slične, što nije slučaj za igralište. Razlog tome je što korisnici, osim rezerviranja termina na igralištu, mogu i prijaviti učestvovanje na terminima drugih korisnika. Također, prilikom rezervacije, potrebno je definirati nešto više parametara nego kod teretane ili praone.

Na početnoj stranici za igralište nalaze se, kao što je prikazano na slici 3.9, četiri kartice - jedna za rezervaciju termina, druga za pregled rezerviranih termina, treća za prijavu na termine drugih korisnika, te posljednja za popis termina na koje se korisnik prijavio.



Slika 3.9 Izgled početne stranice za igralište

3.6.1 Rezervacija termina

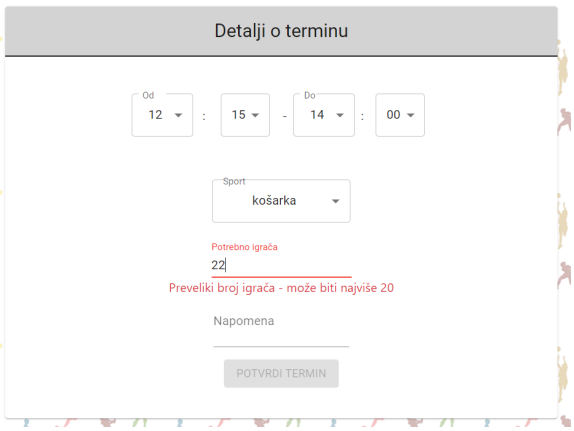
Na stranici za rezervaciju, korisnik prvo unutar kalendara odabire datum za koji želi rezervirati igralište. Ispod kalendara nalazi se obrazac u kojem korisnik unosi određene podatke.

Za početak, potrebno je iz padajućih izbornika izabrati početno i krajnje vrijeme termina. Dostupno vrijeme za rezervaciju jest od 7:00 do 23:45. Postoje ograničenja po pitanju trajanja termina - naime, termin mora trajati najmanje trideset minuta, ali ne smije trajati dulje od dva sata i trideset minuta. U slučaju da to pravilo nije zadovoljeno, korisniku se prikazuje odgovarajuća poruka, kao na slici 3.10.

Slika 3.10 Primjer definiranja predugačkog termina

Zatim, korisnik odabire sport koji će se igrati. Na igralištu studentskog doma mogu se igrati košarka, nogomet te stolni tenis, pa korisnik može odabrati jedan od ta tri sporta. Nakon toga, korisnik treba unijeti željeni broj igrača za taj termin. Taj broj ne predstavlja ograničenje broja igrača te se neće na temelju tog broja onemogućiti korisnicima prijava na termin ukoliko je već dovoljno igrača prijavljeno, već služi samo kao informacija drugim korisnicima o tome koliko igrača vlasnik tog termina očekuje. Ova vrijednost također ima svoja ograničenja - broj igrača mora biti u rasponu od dva do dvadeset, u suprotnom se korisniku prikazuje poruka kao na slici 3.11.

Posljednja, neobavezna informacija koju korisnik može unijeti jest napomena o



Slika 3.11 Primjer definiranja prevelikog broja igrača

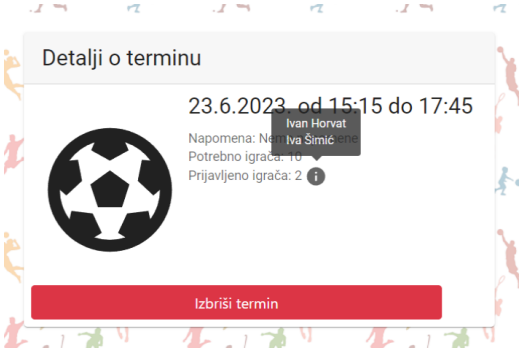
terminu. Ovdje korisnik može unijeti nekakvu dodatnu informaciju, na primjer *"Zamolio bih da netko donese loptu"*.

Nakon što je korisnik odredio sve nužne informacije, omogućava mu se slanje zahtjeva za rezervaciju. Kao i kod teretane i praone, ovdje se također provjeravaju određeni uvjeti koji određuju hoće li zahtjev biti uspješan ili ne. Ukoliko je korisnik rezervirao termin na igralištu za dan koji je definirao u zahtjevu ili već ima pet rezerviranih termina, zahtjev će biti odbijen. Također, zbog ograničenog broja terena mogu se samo dva termina iz košarke međusobno preklapati, dok termini za nogomet i stolni tenis ne smiju imati preklapanja.

3.6.2 Popis termina

Na stranici za popis termina, korisniku se prikazuje tablica sa datumom, početkom i krajem svakog budućeg termina kojeg je rezervirao. Kao i kod teretane i praone, korisniku se pruža mogućnost brisanja termina, pritiskom na gumb na kraju retka. Također, u svakom retku tablice nalazi se i gumb koji korisnika vodi na stranicu za prikaz detalja o terminu, prikazanu na slici 3.12. Na toj se stranici nalazi kartica sa svim detaljima koje je korisnik definirao, a također se može vidjeti i broj korisnika koji su se prijavili na termin. Pri prelasku preko ikone koja se nalazi pored broja prijavljenih igrača, korisniku se prikazuju imena prijavljenih korisnika. Na dnu kartice

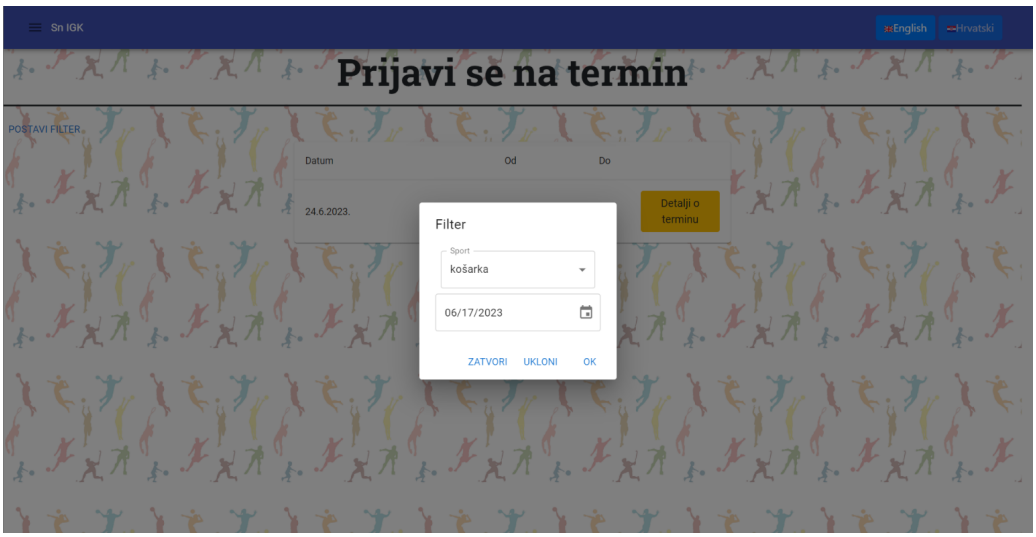
nalazi se gumb za brisanje termina.



Slika 3.12 Kartica s detaljima termina

3.6.3 Prijava na termin

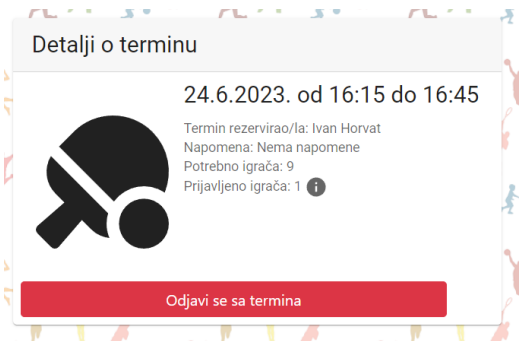
Na stranici za prijavu termina nalazi se tablica u kojoj se prikazuju svi budući termini drugih korisnika na koje se korisnik može prijaviti, odnosno na koje već nije prijavljen. Iznad tablice nalazi se tipka koja otvara prozor u kojem korisnik može definirati filtar za termine, kao što je prikazano na slici 3.13. Termini se mogu filtrirati po sportu ili po datumu. Na dnu prozora nalaze se tri gumba: *Zatvori* - njime se poništavaju promjene filtra i zatvara prozor, *Ukloni* - njime se uklanja prethodno postavljeni filtar, *Ok* - njime se primjenjuje definirani filtar. Nakon primjene filtra, tablica se osvježava te se u njoj prikazuju samo oni termini koji zadovoljavaju filtar. Za svaki termin prikazani su datum, početak i kraj, a na kraju retka nalazi se gumb koji korisnika vodi na stranicu s detaljima o terminu. Na toj se stranici nalazi kartica s detaljima, slična onoj opisanoj u sekciji 3.6.2, a jedina razlika je u tome što se na dnu kartice nalazi gumb za prijavu. Nakon uspješne prijave, korisnik se preusmjerava na početnu stranicu.



Slika 3.13 Primjer definiranja parametara filtriranja

3.6.4 Pregled prijavljenih termina

Na ovoj stranici korisnik može pregledati sve termine na koje se prijavio. U tablici termina prikazani su kao i u tablici opisanoj u prethodnoj sekciji, datum te početak i kraj termina, a u svakom retku nalazi se i gumb koji vodi korisnika na stranicu s detaljima o terminu. I ovdje se korisniku prikazuje već opisana kartica s detaljima, a na dnu te kartice nalazi se gumb za poništavanje prijave na termin, kao što je prikazano na slici 3.14.



Slika 3.14 Kartica termina s gumbom za odjavu

Poglavlje 4

Funkcionalnost

U ovom poglavlju detaljno je objašnjena funkcionalnost prijave na termine drugih korisnika na igralištu, iz korisničke te iz implementacijske perspektive.

4.1 Dohvaćanje dostupnih termina

Kada korisnik na početnoj stranici igrališta odabere opciju prijave na termin, usmjerava se na stranicu na kojoj će se prikazati svi dostupni termini. Prilikom učitavanja stranice, naredbom *axios.get('/api/playfieldsessions/list')* šalje se zahtjev poslužitelju za dohvaćanje svih termina na koje se korisnik može prijaviti. Putanja definirana u tom zahtjevu nije potpuna, te ova naredba sama po sebi nije dovoljna za uspješno slanje zahtjeva. No, u datoteci *App.js* definirani su osnovni parametri za slanje zahtjeva pomoću *axios* knjižnice, što je prikazano na slici 4.1. Može se vidjeti kako je definirano da se svi zahtjevi šalju na *localhost:8000*, na kojem je pokrenut poslužitelj. Također, vidljivo je kako se prilikom slanja zahtjeva postavlja *Bearer* token čime se omogućava pristup zaštićenim rutama na poslužitelju.

```
axios.defaults.baseURL = "http://localhost:8000/";
axios.defaults.headers.post['Accept'] = 'application/json';
axios.defaults.headers.post['Content-Type'] = 'application/json';
axios.defaults.withCredentials = true;
axios.interceptors.request.use(function (config) {
  const token = localStorage.getItem('auth_token');
  config.headers.Authorization = token ? `Bearer ${token}` : '';
  return config;
}
```

Slika 4.1 Definirani parametri za *axios* zahtjeve

Kada taj zahtjev stigne do poslužitelja, unutar *api.php* datoteke se na temelju definirane rute pronalazi odgovarajuća metoda za obradu zahtjeva. U ovom slučaju, to je metoda *list()* unutar *PlayfieldsessionController.php* upravljača. U toj se metodi pomoću metode *showIdsOfSessionsUserAppliedFor()*, prikazane na slici 4.2, dohvaćaju svi termini na koje je korisnik već prijavljen.

```
public function showIdsOfSessionsUserAppliedFor()
{
    $id = auth('sanctum')->user()->id;
    return User::find($id)->sessionsAppliedFor->pluck('pivot.playfield_session_id');
}
```

Slika 4.2 Funkcija *showIdsOfSessionsUserAppliedFor()*

Metoda *sessionsAppliedFor()* definirana je u modelu *User.php*, a njome se definira veza za prijavu na termin, između tablice korisnika i tablice termina na igralištu. Ta je veza tipa više-na-više (eng. *many-to-many*) pa je za potrebe te veze kreirana tablica *user_playfield_session*, u kojoj su povezani parovi korisnika i prijavljenih termina. Primjer podataka u toj tablici prikazan je na slici 4.3.

user_id	playfield_session_id
1	26
7	28
8	28
1	27

Slika 4.3 Primjer podataka u tablici *user_playfield_session*

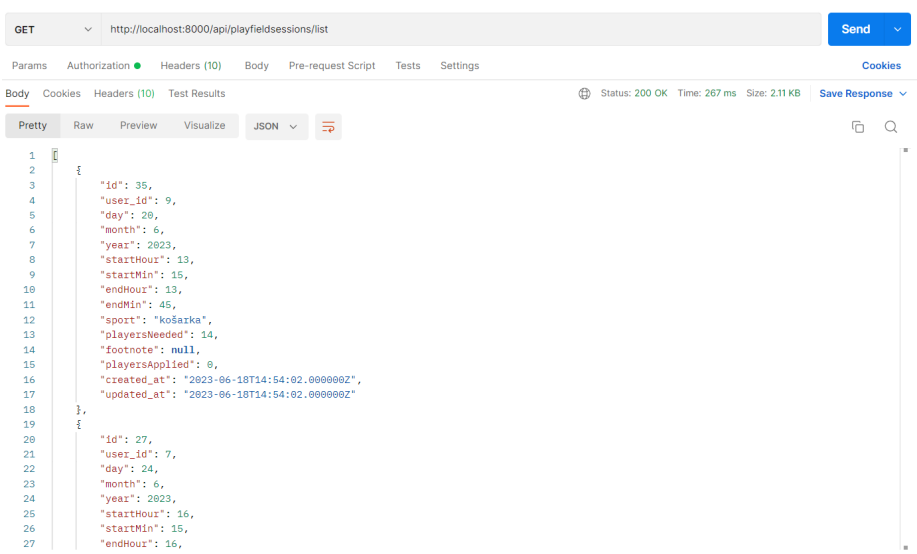
Nakon što su dohvaćeni svi termini na koje je korisnik prijavljen, u *$sessionsUserApplliedFor* se pohranjuje lista *id*-eva tih termina. Zatim se iz baze dohvaćaju svi budući termini na koje se korisnik može prijaviti, upitom prikazanim na slici 4.4. Da bi se mogli vratiti samo budući termini, potrebno je uspoređivati podatke o datumu i vremenu termina s trenutnim datumom i vremenom. To je omogućeno pomoću funkcije *date()*, ugrađene PHP funkcije koja korisniku vraća podatke o vremenu. Prilikom poziva te funkcije može se definirati parametar kojim se određuje koji podatak o trenutnom vremenu se želi dohvatiti. Na primjer, pomoću *date("n")* dohvaća se trenutni mjesec, dok se pomoću *date("G")* dohvaća trenutni sat.

Nakon što su termini filtrirani na temelju podataka o vremenu, uklanjaju se i termini koje je kreirao korisnik koji je trenutno prijavljen u aplikaciju jer mu se ne smije dopustiti da se prijavi na vlastiti termin. Također, iz te liste termina uklanjaju se termini čiji se *id*-evi nalaze u *$sessionsUserApplliedFor*, čime se onemogućava da se u listi termina prikažu oni termini na koje je korisnik već prijavljen. Za kraj, termini su sortirani po vremenu kako bi se korisniku olakšao pronalazak odgovarajućeg termina.

```
$query = PlayfieldSession::latest()
    ->where('year', '>', date("Y"))
    ->orWhere(function ($query) {
        $query->where('year', '=', date("Y"))->where('month', '>', date("n"))
            ->orWhere(function ($query) {
                $query->where('month', '=', date("n"))->where('day', '>', date("j"))
                    ->orWhere(function ($query) {
                        $query->where('day', '=', date("j"))->where('startHour', '>', date("G"))
                            ->orWhere(function ($query) {
                                $query->where('startHour', '=', date("G"))->where('startMin', '>', date("i"));
                            });
                    });
            });
    })
    ->where('user_id', '!=', auth()->id())->whereIn('id', $sessionsUserAppliedFor)->get()
    ->sortBy(['year', 'month', 'day', 'startHour', 'startMin']);
```

Slika 4.4 Upit kojim se dohvaćaju termini

Ti se podaci vraćaju klijentskoj strani aplikacije u JSON formatu pomoću naredbe *response()->json($query)*, čime se omogućava jednostavnije obrađivanje i prikaz tih podataka. Na slici 4.5 je u alatu Postman prikazan primjer podataka koje poslužitelj vraća klijentskoj strani.

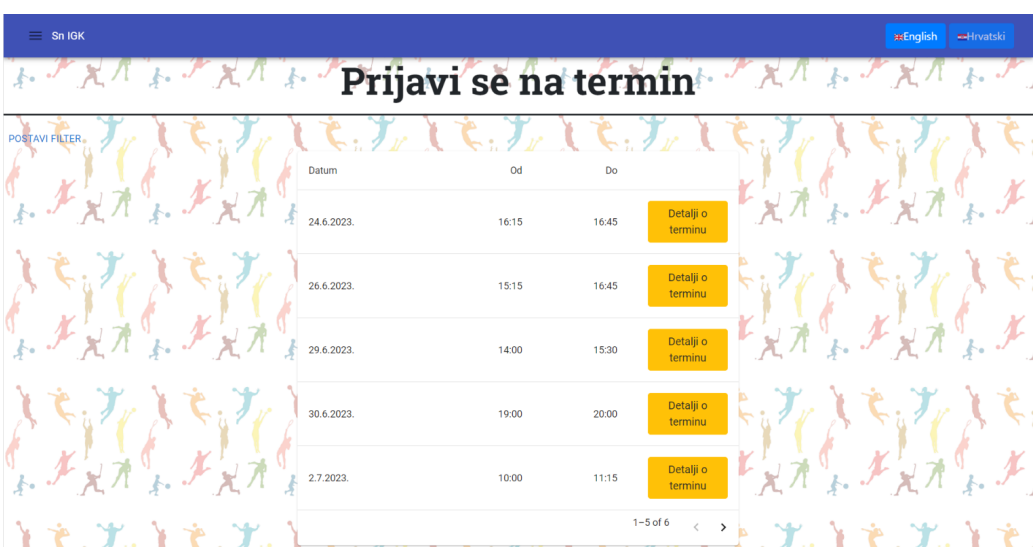


Slika 4.5 Podaci o terminima prikazani u alatu Postman

4.2 Prikaz dostupnih termina

Nakon što su dohvaćeni termini, spremaju se u listu *sessions* te u listu *filteredSessions*, koja će se koristiti za pohranu termina koji odgovaraju postavljenom filtru, što je objašnjeno u poglavlju 4.3. Ukoliko filtar nije postavljen, u toj će se listi nalaziti svi dostupni termini.

Za prikaz termina kreiran je element *SessionTable*, prikazan na slici 4.6, koji u sebi sadrži tablicu u kojoj su prikazani datum i vrijeme održavanja termina te gumb koji korisnika prosljeđuje na stranicu za prikaz detalja i prijavu na termin. Ovaj element koristi se i na stranicama za popis rezerviranih termina u teretani, praoni i na igralištu. Jedina razlika u ovom slučaju jest ta da se u ovoj tablici koristi paginacija zbog toga što je broj rezerviranih termina od strane drugih korisnika neograničen što može u određenim situacijama dovesti do prelaska tablice izvan okvira stranice. Paginacijom je postavljeno da se prikazuje pet termina po stranici, čime je osiguran ispravan prikaz tablice.



Slika 4.6 Tablica s dostupnim terminima za prijavu

4.3 Filtriranje termina

U situacijama kada postoji puno dostupnih termina, korisniku može biti teže pronaći termin koji mu odgovara. Zbog toga je omogućeno filtriranje termina, čime se omogućava selekcija termina po željenom datumu i/ili sportu. Pri vrhu stranice kreiran je element *Filter.tsx*, koji se sastoji od gumba i dijaloškog okvira koji se otvara pritiskom na gumb. Unutar tog okvira, korisniku je omogućeno postavljanje parametara filtriranja.

U *Filter.tsx* definirana je varijabla *unappliedFilter* u kojoj se pohranjuju parametri filtriranja prije nego što se filtar primijeni. Unutar dijaloškog okvira definirana su dva polja - jedno za odabir sporta, drugo za odabir datuma. U prvom polju nalazi se padajući izbornik u kojem su ponuđena sva tri sporta. Prilikom promjene odabira poziva se funkcija *handleChange*, prikazana na slici 4.7, kojom se mijenja odabrani sport unutar *unappliedFilter*.

```
const handleChange = (event: SelectChangeEvent<typeof filter.sport>) => {
  setUnappliedFilter({...unappliedFilter, sport: String(event.target.value) || ''});
};
```

Slika 4.7 Promjena odabranog sporta u filtru

U drugom polju, čija je implementacija prikazana na slici 4.8, nalazi se *DatePicker* element kojim se odabire željeni datum.

```
<LocalizationProvider dateAdapter={AdapterDayjs} >
  <DatePicker minDate={dayjs(new Date())} value={dayjs(defaultDate)}
    onChange={(date) => {
      setUnappliedFilter({...unappliedFilter, day: date && date.date(), month: date && date.month() + 1, year: date && date.year()})
    }}/>
</LocalizationProvider>
```

Slika 4.8 Element za odabir datuma

Kao što je već objašnjeno u sekciji 3.6.3, pri dnu dijaloškog okvira nalaze se tri gumba: *Zatvori*, *Ukloni* i *Ok*. Pritiskom na gumb *Ok* primjenjuje se filtar koji je korisnik definirao, pozivanjem funkcije *handleApply()*, koja je prikazana na slici 4.9. Na početku funkcije se *unappliedFilter* pohranjuje u *Filter*. Zatim se u *filSessArray*

dodaju svi termini koji imaju željenu vrijednost sporta i datuma. Ta se lista naposljetku prepisuje u *filteredSessions*, a upravo se iz te liste uzimaju podaci koji će biti prikazani u tablici dostupnih termina.

```
const handleApply = (event: React.SyntheticEvent<unknown>, reason?: string) => {
  setFilter(unappliedFilter)
  const filSessArray: Session[] = [];
      sessions.map(session => {
        if((session.sport === unappliedFilter.sport || unappliedFilter.sport === '') &&
          (!unappliedFilter.day || session.day === unappliedFilter.day) &&
          (!unappliedFilter.month || session.month === unappliedFilter.month) &&
          (!unappliedFilter.year || session.year === unappliedFilter.year)){
          filSessArray.push(session)
        }
      })
      setFilteredSessions(filSessArray)

  if (reason !== 'backdropClick') {
    setOpen(false);
  }
};
```

Slika 4.9 Funkcija za primjenu filtra

Ukoliko ne postoji niti jedan termin koji odgovara definiranim parametrima, korisniku se prikazuje odgovarajuća poruka. Također, korisniku su nudi opcija kreiranja vlasitog termina, kao što je prikazano na slici 4.10.

Slika 4.10 Prikaz poruke u slučaju kada nijedan termin ne odgovara filtru

4.4 Prijava na termin

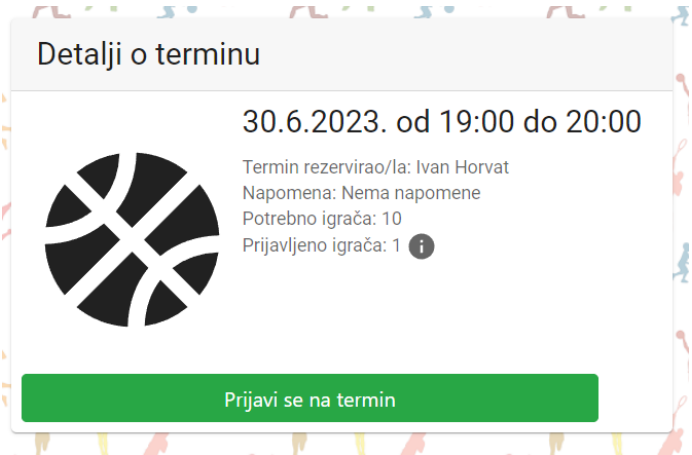
Na slici 4.6 prikazana je tablica s dostupnim terminima. Na kraju svakog retka nalazi se gumb koji korisnike vodi na stranicu na kojoj se prikazuju detalji o odabranom terminu. Na toj se stranici generira element *PlayfieldApplyDetails*, pomoću kojeg se korisniku prikazuju detalji o terminu. Pri učitavanju tog elementa dohvaćaju se podaci o terminu te se provjerava je li korisnik vlasnik tog termina. Ukoliko jest, vrši se preusmjeravanje na početnu stranicu kako bi se korisniku onemogućila prijava na vlastiti termin.

Osim podataka o terminu, dohvaćaju se i *id*-evi svih termina na koje je korisnik prijavljen, kodom prikazanim na slici 4.11. Dohvaćeni podaci spremaju se u listu, a zatim se provjerava nalazi li se u listi i trenutno prikazani termin. Na temelju toga postavlja se vrijednost varijable *alreadyApplied*, kojom se određuje hoće li se pri dnu kartice za prikaz podataka nalaziti gumb za prijavu ili gumb za poništavanje prijave.

```
axios.get('/api/playfieldsessions/applied/ids')
  .then(response => {
    Object.values(response.data).map(s => {
      sessIdArray.push(s)
    })

    setSessionIds(sessIdArray)

    if (sessIdArray.includes(sess.id)) {
      setAlreadyApplied(true)
    }

  });
```

Slika 4.11 Dohvaćanje termina na koje je korisnik prijavljen

Glavni element na ovoj stranici jest *SessionDetailsCard*, kartica u kojoj se prikazuju detalji o terminu, a prikazana je na slici 4.12.



Slika 4.12 Prikaz detalja o terminu

U primjeru na slici korisnik još nije prijavljen na termin pa se pri dnu kartice nalazi gumb za prijavu na termin. Pritiskom na taj gumb poziva se funkcija *applyForSession()*, čiji je kod prikazan na slici 4.14. U toj se funkciji šalje zahtjev prema poslužitelju, na rutu */api/playfieldsessions/apply*, a jedini podatak koji je potrebno definirati u zahtjevu jest *id* termina. Na poslužiteljskoj strani taj se zahtjev obrađuje pomoću funkcije *storeApplication*, prikazane na slici 4.13. U toj se funkciji vrši provjera ispravnosti zahtjeva, a ako je zahtjev ispravan, korisnik se prijavljuje na termin. Ispravnost zahtjeva temelji se na više uvjeta:

- Termin mora postojati
- Korisnik ne smije biti vlasnik termina
- Korisnik ne smije već biti prijavljen na termin

```php
public function storeApplication(StorePlayfieldApplicationRequest $request)
{
    $user_id = auth('sanctum')->user()->id;
    $user = User::find($user_id);
    $session = PlayfieldSession::find($request['session_id']);

    if ($session) {
        if ($user_id == $session->user_id) {
            return response()->json([
                'status' => 313
            ]);
        } else {
            $sessions_applied_for = $user->sessionsAppliedFor()->get()->pluck('pivot.playfield_session_id')->toArray();

            if (in_array($session->id, $sessions_applied_for)) {
                return response()->json([
                    'status' => 314
                ]);
            } else {
                $user->sessionsAppliedFor()->attach($session->id);

                $session->playersApplied++;
                $session->save();

                return response()->json([
                    'status' => 200
                ]);
            }
        }
    } else {
        return response()->json([
            'status' => 315
        ]);
    }
}
```

Slika 4.13 Funkcija *storeApplication*

Ovisno o postojanju pogreške te o tipu pogreške, klijentskoj strani se vraća odgovarajući status zahtjeva:

- status 200 - uspješan zahtjev
- status 313 - korisnik se pokušava prijaviti na vlastiti termin
- status 314 - korisnik je već prijavljen na taj termin
- status 315 - termin ne postoji

Na temelju statusa, korisniku se prikazuje odgovarajuća *swal* poruka, kodom prikazanim na slici 4.14. Ukoliko je zahtjev uspješan, korisnik se preusmjerava na početnu stranicu. U suprotnom, preusmjerava se na stranicu za odabir termina za prijavu.

```
const applyForSession = (e) => {
  const data = {
    'session_id': session.id
  }

  axios.post('/api/playfieldsessions/apply', data)
    .then(res => {
      if (res.data.status === 200) {
        swal(t('success-label'), t('session.apply.success'), 'success')
        navigate('/');
      }
      else if (res.data.status === 313) {
        swal(t('warning-label'), t('session.apply.fail.own-session'), 'warning')
        navigate('/playfield/apply');
      }
      else if (res.data.status === 314) {
        swal(t('warning-label'), t('session.apply.fail.already-applied'), 'warning')
        navigate('/playfield/apply');
      }
      else if (res.data.status === 315) {
        swal(t('warning-label'), t('session.apply.fail.not-exist'), 'warning')
        navigate('/playfield/apply');
      }
    })
}
```

Slika 4.14 Funkcija za prijavu na termin

4.5 Prikaz prijavljenih termina i odjava s termina

Nakon što se korisnik prijavi na određeni termin, taj će mu se termin prikazivati na stranici s prijavljenim terinima, u tablici koja je izgledom jednaka tablici prikazanoj na slici 4.6. Kao i u toj tablici, ovdje se također na kraju svakog reda nalazi gumb za prikaz detalja termina. Pritiskom na taj gumb, učitava se element *PlayfieldApplyDetails*, principom koji je već objašnjen u sekciji 4.4. Jedina razlika ovdje je što će se prilikom dohvaćanja termina na koje je korisnik prijavljen utvrditi kako je korisnik već prijavljen na taj termin, zbog čega će vrijednost *alreadyApplied* biti postavljena u *true*, što će uzrokovati da se pri dnu kartice s detaljima o terminima prikaže gumb koji korisniku omogućava odjavu s termina, kao što je već prikazano na slici 3.14.

Prilikom pritiska na taj gumb, poziva se funkcija *withdrawApplication*, prikazana na slici 4.15. U toj se funkciji šalje zahtjev prema poslužitelju, na rutu */api/playfieldsessions/apply/delete*. Jedini podatak koji je potrebno definirati u zahtjevu jest id termina, a pomoću *'_method': 'DELETE'* definira se da se tim zahtjevom žele obrisati određeni podaci.

```
const withdrawApplication = (e) => {
  const data = {
    'session_id': session.id,
    '_method': 'DELETE'
  }

  axios.post('/api/playfieldsessions/apply/delete', data)
    .then(res => {
      if (res.data.status === 200) {
        swal(t('success-label'), t('session.apply.delete.success'), 'success')
        navigate('/');
      }
      else if (res.data.status === 316) {
        swal(t('warning-label'), t('session.apply.delete.fail'), 'warning')
        navigate('/');
      }
    })
}
```

Slika 4.15 Funkcija *withdrawApplication*

Na poslužiteljskoj strani taj se zahtjev obrađuje pomoću funkcije *withdrawApplication*, prikazane na slici 4.16. Unutar te funkcije se prvo dohvaćaju svi termini na koje je korisnik prijavljen te se vrši provjera nalazi li se termin u toj listi. Ukoliko se ne nalazi, funkcija vraća odgovarajuću vrijednost varijable status pomoću koje će se na klijentskoj strani prikazati odgovarajuća poruka. U suprotnom, zahtjev se smatra uspješnim te se pomoću linije *$user->sessionsAppliedFor()->detach($id);* iz skupa termina na koje je korisnik prijavljen uklanja odabrani termin. Također, za odabrani termin smanjuje se broj korisnika koji su na njega prijavljeni.

```
public function withdrawApplication(DeletePlayfieldApplicationRequest $request)
{
    $id = $request->session_id;

    $ids = PlayfieldSessionController::showIdsOfSessionsUserAppliedFor()->toArray();

    if (!in_array($id, $ids)) {
        return response()->json([
            'status' => 316
        ]);
    } else {
        $user = User::find(auth('sanctum')->user()->id);
        $user->sessionsAppliedFor()->detach($id);

        $session = PlayfieldSession::find($id);
        $session->playersApplied--;
        $session->save();

        return response()->json([
            'status' => 200
        ]);
    }
}
```

Slika 4.16 Funkcija *withdrawApplication* unutar *PlayfieldSessionController.php*

Poglavlje 5

Zaključak

Ova aplikacija namijenjena je svim studentima koji stanuju u Studentskom naselju 'Ivan Goran Kovačić', a žele na jednostavniji način koristiti usluge koje im se pružaju te rezervirati termine u samo nekoliko klikova.

Za razvoj poslužiteljskog dijela aplikacije korišten je Laravel. Glavna prednost Laravela je to što na jednostavan i intuitivan način omogućava uspostavljanje poslužiteljskog dijela aplikacije prateći MVC arhitekturu. Također, Laravel pruža razne alate kojima se mogu postići dodatne funkcionalnosti, kao što je auntentikacija pomoću Laravel Sanctum. Za razvoj klijentskog dijela korišten je React, koji omogućuje učinkovito učitavanje modularnog sadržaja u klijentskom dijelu aplikacije. React se jednostavno nadovezuje na Laravel te se njihovim zajedničkim radom ostvaruje potpuna funkcionalnost aplikacije. Svi korišteni alati imaju veliku zajednicu korisnika te je stoga rješavanje svih problema u razvoju bilo brzo i učinkovito.

U trenutnoj verziji aplikacije implementiran je samo dio predviđenih funkcionalnosti. Neke od funkcionalnosti koje bi se potencijalno u budućnosti mogle dodati su prijava studenata pomoću *AAI@edu.hr* sustava, kao i kreiranje uloge administratora, koja bi zaposlenicima studentskog doma omogućila uvid i kontrolu nad svim terminima. Također, moguće je unaprijeđenje dizajna kako bi se poboljšalo korisničko iskustvo.

Bibliografija

[1] Laravel dokumentacija, s Interneta, https://laravel.com/docs/10.x, 20. svibnja 2023.

[2] Laravel Sanctum, s Interneta, https://laravel.com/docs/master/sanctum, 20. svibnja 2023.

[3] "Laravel hasMany and belongsTo Tutorial", s Interneta, https://vegibit.com/laravel-hasmany-and-belongsto-tutorial/, 26. lipnja 2023.

[4] "Build a React App With a Laravel RESTful Back End: Part 1, Laravel 9 API", s Interneta, https://code.tutsplus.com/tutorials/build-a-react-app-with-laravel-restful-backend-part-1-laravel-5-api--cms-29442, 21. svibnja 2023.

[5] "Why is React Declarative? A Story About Function Components", s Interneta, https://medium.com/trabe/why-is-react-declarative-a-story-about-function-components-aaae83198f79, 24. svibnja 2023.

[6] React: The Virtual DOM, s Interneta, https://www.codecademy.com/article/react-virtual-dom, 9. srpnja 2023.

[7] "SweetAlert documentation", s Interneta, https://sweetalert.js.org/guides/, 6. lipnja 2023.

[8] Material Design, s Interneta, https://m3.material.io/, 7. lipnja 2023.

[9] Material UI. , s Interneta, https://mui.com/material-ui/getting-started/overview/, 7. lipnja 2023.

[10] Material UI Drawer, s Interneta, https://mui.com/material-ui/react-drawer/, 7. lipnja 2023.

[11] MySQL wikipedia, s Interneta, https://en.wikipedia.org/wiki/MySQL, 21. svibnja 2023.

Sažetak

Tema ovoga rada je izrada *full stack* web aplikacije za upravljanje terminima u Studentskom naselju 'Ivan Goran Kovačić'. Ova aplikacija omogućuje studentima da rezerviraju, pregledavaju te uklanjaju termine u teretani, praoni ili na igralištu. Također, korisnicima se omogućuje prijava na termine drugih korisnika na igralištu. Za razvoj poslužiteljskog dijela aplikacije korišten je *Laravel*, dok je za klijentski dio korišten *React*. Od ostalih tehologija, korišteni su *Laravel Sanctum* za autentikaciju korisnika te *Material UI* i *SweetAlert* za unaprijeđenje dizajna aplikacije.

Ključne riječi — **web aplikacija, rezervacija termina, Laravel, React**

Abstract

The topic of this paper is the development of a *full stack* web application for managing sessions in the 'Ivan Goran Kovačić' Student Housing. This application enables students to reserve, view, and cancel sessions for the gym, laundry room, or playground. Additionally, users are able to sign up for sessions scheduled by other users for the playground. The server-side of the application was developed using *Laravel*, while *React* was used for the client-side. Among other technologies, *Laravel Sanctum* was used for user authentication, and *Material UI* and *SweetAlert* were utilized to enhance the application's design.

Keywords — **web application, session booking, Laravel, React**