Univerzitet u Beogradu  
Fakultet organizacionih nauka  
Katedra za elektronsko poslovanje

Web aplikacija za rezervacije u salonu lepote

Seminarski rad iz predmeta Internet tehnologije

Mentor: Studenti:

Beograd, 2023.

Sadržaj

[Korisnički zahtev 1](#_Toc158253474)

[Opis sistema 1](#_Toc158253475)

[Opis slučajeva korišćenja 1](#_Toc158253476)

[Opis arihtekture aplikacije 4](#_Toc158253477)

[Opis procesa slučajeva korišćenja 5](#_Toc158253478)

[Model podataka 8](#_Toc158253479)

[Specifikacija REST API-ja 9](#_Toc158253480)

[Opis tehnologija korišćenih u aplikaciji 17](#_Toc158253481)

[Laravel 17](#_Toc158253482)

[React 18](#_Toc158253483)

[Axios 18](#_Toc158253484)

[Korisničko uputstvo 18](#_Toc158253485)

[Prikaz delova koda 23](#_Toc158253486)

[Link ka github repozitorijumu 44](#_Toc158253487)

[Reference 45](#_Toc158253488)

# Korisnički zahtev

Potrebno je kreirati aplikaciju koja će olakšati rad salona za lepotu. Aplikacija treba da obezbedi tri korisničke uloge – admin, šminker i korisnik. Za svaku od ovih uloga je potrebno obezbediti rad sa sesijom, tj. Autentifikaciju preko korisničkog imena i lozinke.

U zavisnosti od uloge, korisnici imaju različite mogućnosti:

* Neulogovani korinisci imaju mogućnost da se uloguju ili registruju, kao i da pregledaju cenovnik usluga. Zbog velikog broja stranih državljana potrebno je obezbediti prikaz cena u raznim valutama
* Ulogovani korisnici imaju mogućnost da kreiraju rezervaciju za odabranu uslugu kod odabranog zaposlenog. Ukoliko je odabrani zaposleni već zauzet u tom terminu, sistem ne sme da dopusti da se desi dvostruko bukiranje. Takođe, korisnicima stiže mejl kao potvrda o kreiranoj rezervaciji.
* Ulogovani šminkeri imaju mogućnost pregleda svog današnjeg i sutrašnjeg rasporeda, kao i mogućnost vršenja CRUD operacijama nad uslugama koje se nude u salonu.
* Administratorima sistema je potrebno omogućiti pregled statistika vezanih za poslovanje salona.

# Opis sistema

U ovom poglavlju ćemo dati kratak pregled opisa sistema kroz dijagrame sekvenci, slučajeva korišćenja, dijagram klasa I specifikaciju RESTfull APIja.

## Opis slučajeva korišćenja

### Prijava korisnika

Slučaj korišćenja: Prijava korisnika na sistem

Akteri:

* Korisnik
* Sistem (program)Preduslov:
* Sistem je uključen, a korisnik ima kreiran i verifikovan nalog za pristup.

Osnovni scenario:

1. Korisnik bira opciju za prijavu na sistem.
2. Sistem prikazuje formu za prijavu koja traži korisničko ime i lozinku.
3. Korisnik unosi svoje korisničko ime i lozinku u predviđena polja.
4. Korisnik potvrđuje unos i šalje podatke sistemu klikom na dugme "Prijavi se".
5. Sistem proverava da li su uneti podaci validni i da li se podudaraju sa podacima u bazi.
6. Ukoliko su podaci ispravni, sistem autorizuje korisnika i daje mu pristup odgovarajućim funkcionalnostima na osnovu njegovog nivoa pristupa.
7. Sistem prikazuje poruku "Uspešno ste prijavljeni na sistem" i preusmerava korisnika na početnu stranu.

Alternativni scenariji: 5.1 Ukoliko korisničko ime i/ili lozinka nisu ispravni:

* Sistem prikazuje poruku o grešci: "Neispravno korisničko ime ili lozinka. Molimo pokušajte ponovo."

### Kreiranje rezervacije

Slučaj korišćenja: Kreiranje rezervacije

Akteri:

• Korisnik (Klijent)

• Sistem (aplikacija za rezervacije)

Preduslov: Korisnik je prijavljen na sistem i ima pristup funkcionalnostima za kreiranje rezervacije.

Osnovni scenario:

1. Korisnik pristupa opciji za kreiranje nove rezervacije unutar sistema.
2. Sistem prikazuje formu za unos podataka o rezervaciji, uključujući datum, vreme, uslugu i druge relevantne informacije.
3. Korisnik unosi tražene informacije o rezervaciji u formu.
4. Korisnik pregleda unete podatke i proverava da li su sve informacije ispravne i potpune.
5. Korisnik potvrđuje unos klikom na dugme "Potvrdi rezervaciju" ili slično.
6. Sistem proverava unete podatke na validnost i potpunost.
7. Ukoliko su svi podaci ispravni, sistem kreira novu rezervaciju i registruje je u bazi.
8. Sistem obaveštava korisnika porukom: "Nova rezervacija je uspešno kreirana."

Alternativni scenario (korak 7): 7a. Ukoliko neki od unetih podataka nisu ispravni ili potpuni, sistem obaveštava korisnika o grešci i vraća ga na korak 2 kako bi ispravio podatke.

### Kreiranje usluge

Slučaj korišćenja: Kreiranje usluge

Akteri:

• Administrator

• Sistem (administrativni sistem za upravljanje uslugama)

Preduslov: Administrator je prijavljen na sistem i ima privilegije za kreiranje novih usluga.

Osnovni scenario:

1. Administrator pristupa opciji za kreiranje nove usluge unutar sistema.
2. Sistem prikazuje formu za unos podataka o usluzi, uključujući naziv usluge, opis, cena i druge relevantne informacije.
3. Administrator unosi tražene informacije o usluzi u formu.
4. Administrator pregleda unete podatke i proverava da li su sve informacije ispravne i potpune.
5. Administrator potvrđuje unos klikom na dugme "Kreiraj uslugu" ili slično.
6. Sistem proverava unete podatke na validnost i potpunost.
7. Ukoliko su svi podaci ispravni, sistem kreira novu uslugu i registruje je u bazi.
8. Sistem obaveštava administratora porukom: "Nova usluga je uspešno kreirana."

Alternativni scenario (korak 7): 7a. Ukoliko neki od unetih podataka nisu ispravni ili potpuni, sistem obaveštava administratora o grešci i vraća ga na korak 2 kako bi ispravio podatke.

## Opis arihtekture aplikacije

Naša aplikacija je izgrađena na modernom web razvojnom stack-u koji uključuje Laravel, React i Axios, što omogućava agilan i efikasan razvoj složenih web aplikacija. Laravel je odabran kao backend framework zbog svoje visoke performanse, bezbednosti i jednostavnosti korišćenja. On predstavlja osnovu naše aplikacije, upravljajući bazom podataka, autentifikacijom korisnika, poslovnom logikom i API endpointima. Zahvaljujući Laravelovoj MVC arhitekturi, logika aplikacije je jasno odvojena od prezentacionog sloja, što olakšava održavanje koda i poboljšava skalabilnost aplikacije. API resursi su kreirani pomoću Laravel-a, omogućavajući aplikaciji da komunicira sa frontend-om putem JSON odgovora, što osigurava efikasnu razmenu podataka.

Za izgradnju korisničkog interfejsa koristimo React, JavaScript biblioteku koja omogućava dinamične i interaktivne web stranice. React se koristi za kreiranje komponenata koje definišu elemente i logiku korisničkog interfejsa, omogućavajući brzu izradu single-page aplikacija (SPA). Korišćenjem React-a, aplikacija može dinamički ažurirati i prikazivati odgovarajuće komponente na korisničkom interfejsu u realnom vremenu, bez potrebe za ponovnim učitavanjem stranice, što značajno unapređuje korisničko iskustvo.

Za komunikaciju između React frontend-a i Laravel backend-a koristimo Axios, popularnu JavaScript biblioteku za izvršavanje HTTP zahteva. Axios nam omogućava jednostavno upravljanje zahtevima i odgovorima između klijenta i servera, uključujući konfigurisanje zahteva, prenos tokena za autentifikaciju i obradu odgovora servera. Zahvaljujući Axios-u, naša aplikacija efikasno dohvata podatke, šalje podatke na server i implementira kompleksne funkcionalnosti poput CRUD operacija (Create, Read, Update, Delete) na siguran i pouzdan način. Ova kombinacija tehnologija omogućava nam da razvijemo visoko funkcionalnu i performantnu web aplikaciju koja zadovoljava moderne standarde u razvoju softvera.

## Opis procesa slučajeva korišćenja

### Dijagram sekvenci – prijava korisnika

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

Opis je automatski generisan

Slika 1 ds - prijava korisnika

### Dijagram sekvenci – kreiranje rezervacije

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

Opis je automatski generisan

Slika 2 ds -kreiranje rezervacije

### Dijagram sekvenci – kreiranje usluge

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

Opis je automatski generisan

Slika 3 ds -kreiranje usluge

## Model podataka

Slika na kojoj se nalazi dijagram

Opis je automatski generisan

Slika 4 PMOV

## Specifikacija REST API-ja

U ovom poglavlju ce biti prikazani detalji o api rutama za kontrolere.

|  |  |
| --- | --- |
| Opis funkcije | Prikaz svih rezervacija ulogovanog korisnika |
| HTTP metoda | GET |
| URL | /api/rezervacije |
| URL parametri | nema |
| HTTP body parametri | nema |
| Format HTTP body parametara | JSON |
| Izlazni parametri | {      "data": [          {              "id": 11,              "datum": "2024-02-10 00:00:00",              "vreme": "14:30"          },          {              "id": 12,              "datum": "2024-02-10 00:00:00",              "vreme": "14:30"          },  .. |
| Format izlaznih parametara | Application/json |
|  |  |
| Opis funkcije | Kreiranje rezervacije |
| HTTP metoda | POST |
| URL | /api/rezervacije |
| URL parametri | usluga\_id=1&datum=2024-02-10&vreme=19:55&zaposleni\_id=1 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "data": {          "id": 21,          "datum": "2024-02-10",          "vreme": "19:55",          "usluga": {              "id": 1,              "naziv": "Šminkanje za posebne prilike",              "opis": "Profesionalno šminkanje za sve vrste posebnih prilika.",              "cena": "50.00"          }      }  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Ažuriranje rezervacije |
| HTTP metoda | PUT |
| URL | /api/rezervacije |
| URL parametri | 21?usluga\_id=1&datum=2024-02-10&vreme=19:55&zaposleni\_id=1 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {  {      "data": {          "id": 21,          "datum": "2024-02-10",          "vreme": "19:55"      }  }  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Brisanje rezervacije |
| HTTP metoda | DELETE |
| URL | /api/rezervacije |
| URL parametri | 21 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "message": "Rezervacija deleted"  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Prikaz svih usluga |
| HTTP metoda | GET |
| URL | /api/usluge |
| URL parametri | nema |
| HTTP body parametri | nema |
| Format HTTP body parametara | JSON |
| Izlazni parametri | "data": [          {              "id": 1,              "naziv": "Šminkanje za posebne prilike",              "opis": "Profesionalno šminkanje za sve vrste posebnih prilika.",              "cena": "50.00"          },          {              "id": 2,              "naziv": "Svadbeno šminkanje",              "opis": "Šminkanje za venčanja i svadbeni makeup.",              "cena": "60.00"          },  .. |
| Format izlaznih parametara | Application/json |
|  |  |
| Opis funkcije | Kreiranje usluge |
| HTTP metoda | POST |
| URL | /api/usluge |
| URL parametri | naziv=sminkanje za dan zaljubljenih&cena=50 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "data": {          "id": 6,          "naziv": "sminkanje za dan zaljubljenih",          "opis": **null**,          "cena": "50"      }  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Ažuriranje usluge |
| HTTP metoda | PUT |
| URL | /api/ usluge |
| URL parametri | 6?naziv=sminkanje za dan zaljubljenih&cena=55 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "data": {          "id": 6,          "naziv": "sminkanje za dan zaljubljenih",          "opis": **null**,          "cena": "55"      }  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Brisanje usluge |
| HTTP metoda | DELETE |
| URL | /api/ usluge |
| URL parametri | 6 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "message": "Usluga deleted successfully"  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Prijava korisnika |
| HTTP metoda | POST |
| URL | /api/login |
| URL parametri | email=katarina@gmail.com&password=katarina1234 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "user": {          "id": 11,          "name": "katarina",          "email": "katarina@gmail.com",          "role": "korisnik"      },      "token": "3|M05pQw220BJYDuiOBSkMziLGTpW8EFOlUvBBPRny8381ae3f"  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Odjava korisnika |
| HTTP metoda | POST |
| URL | /api/logout |
| URL parametri | nema |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "message": "You have been successfully logged out"  } |
| Format izlaznih parametara | application/json |

# Opis tehnologija korišćenih u aplikaciji

U ovom poglavlju opisaćemo ključne tehnologije koje su korišćene u izgradnji naše aplikacije, uključujući Laravel za backend, React za frontend i Axios za upravljanje HTTP zahtevima.

### Laravel

Laravel je moderni PHP web framework dizajniran za olakšavanje razvoja web aplikacija kroz intuitivnu sintaksu, bogat set alata i robustne funkcionalnosti. Sa svojom elegantnom MVC arhitekturom, Laravel omogućava jasnu organizaciju koda, gde Modeli predstavljaju strukturu podataka, View-i definiraju korisnički interfejs, a Controller-i upravljaju logikom aplikacije. Ovaj framework dolazi sa mnogim ugrađenim funkcionalnostima, uključujući rutiranje, autentifikaciju, autorizaciju, validaciju podataka i rad sa bazom podataka putem Eloquent ORM-a. Laravel takođe pruža Blade templating engine za efikasno generisanje HTML-a, kao i Artisan komandnu liniju za automatizaciju rutinskih zadataka. Sa svojom aktivnom zajednicom korisnika i kontinuiranim razvojem, Laravel ostaje lider u PHP razvoju aplikacija, pružajući programerima alate potrebne za kreiranje sigurnih, skalabilnih i modernih web aplikacija.

### React

React je JavaScript biblioteka za izgradnju korisničkih interfejsa, razvijena od strane Facebook-a. Odabrali smo React zbog njegove sposobnosti da omogući brz razvoj interaktivnih UI komponenata, koristeći koncept virtualnog DOM-a za minimiziranje potrebe za manipulacijom stvarnog DOM-a i time poboljšanje performansi aplikacije. React takođe podržava JSX, sintaksno proširenje za JavaScript, koje omogućava pisanje UI komponenata koristeći sintaksu sličnu HTML-u unutar JavaScript koda, čineći kod čistijim i lakšim za razumevanje.

### Axios

Axios je obećavajuća JavaScript biblioteka koja se koristi za izvršavanje HTTP zahteva iz browsera i Node.js aplikacija. Izabrali smo Axios zbog njegove jednostavnosti upotrebe, podrške za Promise API, i mogućnosti da se lako konfiguriše za potrebe naše aplikacije. Axios omogućava slanje asinhronih HTTP zahteva ka RESTful API endpointima, podržavajući globalne konfiguracije, presretanje zahteva i odgovora, te automatsko transformisanje JSON podataka. Korišćenje Axiosa u našoj aplikaciji olakšava komunikaciju između frontend-a i backend-a, čineći razmenu podataka brzom i efikasnom.

### HTML (Hypertext Markup Language)

HTML je osnovni jezik koji se koristi za strukturiranje i oblikovanje sadržaja web stranica. Koristili smo HTML za definisanje strukture naših web stranica, uključujući elemente poput <div>, <header>, <footer>, <nav>, itd. HTML je ključan za kreiranje semantički ispravnih stranica, što poboljšava SEO i pristupačnost naše aplikacije.

### CSS (Cascading Style Sheets)

CSS se koristi za stilizovanje i oblikovanje HTML elemenata, što omogućava da naša aplikacija izgleda estetski privlačno i korisnički prijatno. Definisali smo stilove kao što su boje, fontovi, pozadine, margini i padding koristeći CSS. Takođe smo koristili CSS za implementaciju responsivnog dizajna kako bi naša aplikacija bila prilagodljiva na različitim uređajima i ekranima.

### JavaScript

JavaScript je programski jezik koji se koristi za dodavanje interaktivnosti i dinamičnosti na web stranice. Koristili smo JavaScript za manipulaciju DOM-om (Document Object Model), validaciju formi, reagovanje na korisničke interakcije kao što su klikovi i hover, kao i za upravljanje asinhronim zahtevima ka backend serveru. Korišćenjem JavaScripta unapređujemo korisničko iskustvo i funkcionalnost naše aplikacije.

# Korisničko uputstvo

Kako bi korisnici mogli da koriste ovu aplikaciju potrebno je da se uloguju

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

Opis je automatski generisan

Slika 5 login

Ukoliko korisnici nisu ulogovani imaju mogućnost pregleda cenovnika. Zbog velikog broja stranih državljana koji koriste naše usluge, dodata je opcija za prikaz cena u raznim valutama.

Slika na kojoj se nalazi tekst, snimak ekrana, Font

Opis je automatski generisan

Slika 6 cenovnik

Nakon prijave, korisnicima se prikazuju zakazivanja koja su oni kreirali

Slika na kojoj se nalazi tekst, snimak ekrana, Font

Opis je automatski generisan

Slika 7 prikaz mojih rezervacija

Odabirom opcije Dodaj iz navigacionog menija korisnicima se otvara forma za kreiranje nove rezervacije, nakon cega im stize mejl kao potvrda rezervacije

Slika na kojoj se nalazi tekst, snimak ekrana, dizajn

Opis je automatski generisan

Slika 8 kreiranje rezervacije

Potvrda mejla

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

Opis je automatski generisan

Slika 9 potvrda mejla

Korisnici koji nemaju nalog, mogu da ga naprave putem forme za registraciju.

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

Opis je automatski generisan

Slika 10 forma za registracij

Kada se uloguje korisnik koji je zaposleni, tj šminker, njemu se otvara stranica na kojoj može da vidi svoj raspored zakazivanja za danas I sutra, kaon a slici ispod.

Slika na kojoj se nalazi tekst, snimak ekrana, Font

Opis je automatski generisan

Slika 11 prikaz stranice za šminkera

Zaposleni takođe imaju mogućnost vršenja CRUD opreacija nad uslugama, na stranici Usluge, kaon a slici ispod.

Slika na kojoj se nalazi tekst, snimak ekrana, softver, Veb stranica

Opis je automatski generisan

Slika 12 CRUD operacije za usluge

Kada se na system prijavi administrator, njemu se prikazuje stranica sa statistikama.

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

Opis je automatski generisan

Slika 13 admin statistike

# Prikaz delova koda

**RezervacijaController** sadrži nekoliko metoda

1. **index**: Metoda **index** koristi se za prikazivanje liste svih rezervacija koje je napravio trenutno ulogovani korisnik. Koristeći **Rezervacija::where** upit, filtrira rezervacije tako što upoređuje **korisnik\_id** iz tabele rezervacija sa ID-em trenutno ulogovanog korisnika. Rezultati se zatim vraćaju kao kolekcija **RezervacijaResource**

Slika na kojoj se nalazi tekst, snimak ekrana, Font

Opis je automatski generisan

1. **store**: Metoda **store** omogućava kreiranje nove rezervacije. Prvo se vrši validacija dolaznih podataka iz HTTP zahteva kako bi se osiguralo da su svi neophodni podaci prisutni i ispravni. Nakon validacije, kreira se nova rezervacija sa podacima trenutno ulogovanog korisnika kao kreatora rezervacije. Nakon toga, korisniku se šalje e-mail sa potvrdom rezervacije koristeći **Mail::to** i **RezervacijaConfirmation** mailable klasu. Na kraju, vraća se instance **RezervacijaResource** sa detaljima o kreiranoj rezervaciji.

Slika na kojoj se nalazi tekst, snimak ekrana, Font

Opis je automatski generisan

1. **update**: Metoda **update** se koristi za ažuriranje postojeće rezervacije. Nakon što identifikuje rezervaciju preko njenog ID-a, metoda proverava da li trenutno ulogovani korisnik ima pravo da izmeni tu rezervaciju. Ako ima, validacija se ponovo vrši za sve promenjive podatke. Ako nema grešaka, vrše se izmene nad rezervacijom i vraća se ažurirani **RezervacijaResource**.

Slika na kojoj se nalazi tekst, snimak ekrana, softver

Opis je automatski generisan

1. **destroy**: Metoda **destroy** omogućava brisanje rezervacije. Metoda prvo proverava da li rezervacija sa datim ID-em postoji i da li pripada trenutno ulogovanom korisniku. Takođe proverava da li je datum rezervacije u budućnosti, jer prošle rezervacije ne bi trebalo da se brišu. Ako su svi uslovi ispunjeni, rezervacija se briše i korisniku se vraća potvrdni odgovor.

Slika na kojoj se nalazi tekst, snimak ekrana, softver

Opis je automatski generisan

**AuthController** sadrži metode koje upravljaju autentifikacijom korisnika

1. **login**: Metoda **login** se koristi za prijavljivanje korisnika. Najpre se vrši validacija zahteva da bi se osiguralo da su e-mail i lozinka prosleđeni. Zatim, koristi se Eloquent upit **User::where** da bi se pronašao korisnik sa odgovarajućim e-mailom. Ako korisnik postoji i uneta lozinka se poklapa sa onom u bazi podataka (koristeći Laravel-ov **Hash::check**), generiše se token pomoću metode **createToken** i vraća se JSON odgovor sa podacima korisnika i tokenom. U slučaju greške, baca se **ValidationException** sa odgovarajućom porukom.

Slika na kojoj se nalazi tekst, snimak ekrana, softver

Opis je automatski generisan

1. **logout**: Metoda **logout** omogućava korisniku da se odjavi. Ona briše trenutni pristupni token koji se koristi za autentifikaciju korisničkih zahteva. Nakon brisanja tokena, korisnik više neće moći da pristupi zaštićenim rutama dok se ponovo ne prijavi.

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

Opis je automatski generisan

1. **register**: Metoda **register** služi za registraciju novih korisnika. Koristi Laravel validator za proveru da li su svi potrebni podaci poslati i da li su validni (uključujući proveru da li je e-mail jedinstven unutar tabele **users**). Ako su svi podaci ispravni, novi korisnik se kreira u bazi podataka sa šifriranom lozinkom (koristeći **Hash::make**). Nakon toga, generiše se token pomoću **createToken** metode, i vraćaju se podaci o korisniku i token kao JSON odgovor.

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

Opis je automatski generisan

Login komponenta je odgovorna za formu za prijavu korisnika. Koristi se useState hook za praćenje stanja unetih podataka (email adrese i lozinke) i za praćenje tokena koji se dobija nakon uspešne prijave. Takođe, koristi se axios za slanje HTTP zahteva na server kako bi se izvršila prijava. Nakon uspešne prijave, korisnik se preusmerava na stranicu "/rezervacije". U slučaju greške, prikazuje se greška na ekranu.

InputField komponenta je jednostavna komponenta za unos teksta koja prikazuje ikonu (ikonu korisnika ili ikonu katanca, zavisno od toga da li se unosi email adresa ili lozinka) i input polje za unos teksta.

import React, { useState } from 'react';

import axios from 'axios';

import InputField from './InputField';

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

function Login({setToken}) {

    let navigate = useNavigate();

  const [credentials, setCredentials] = useState({

    email: 'adonis.lind@example.org',

    password: 'password',

  });

  const handleChange = (e) => {

    const { name, value } = e.target;

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

  };

  const handleSubmit = async (e) => {

    e.preventDefault();

    try {

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

      console.log(response.data);

        sessionStorage.setItem("token",response.data.token)

        sessionStorage.setItem("id",response.data.user.id)

        setToken(response.data.token);

        navigate('/rezervacije');

    } catch (error) {

      console.error('Login error', error);

        alert(error);

    }

  };

  return (

    <div className="login-container">

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

        <h2>Sign In</h2>

        <InputField

          name="email"

          value={credentials.email}

          onChange={handleChange}

          placeholder="Username"

          type="email"

        />

        <InputField

          name="password"

          value={credentials.password}

          onChange={handleChange}

          placeholder="Password"

          type="password"

        />

        <div className="options">

          <button type="submit" className="login-button">Login</button>

        </div>

        <div className="footer">

          <a href="/signup">Don't have an account? Sign Up</a>

          <a href="/forgot-password">Forgot Password</a>

        </div>

      </form>

    </div>

  );

}

export default Login;

import React from 'react';

import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';

import { faUser, faLock } from '@fortawesome/free-solid-svg-icons';

const InputField = ({ name, value, onChange, placeholder, type }) => {

  const icon = name === 'username' ? faUser : faLock;

  return (

    <div className="input-group">

      <FontAwesomeIcon icon={icon} className="input-icon" />

      <input

        type={type}

        name={name}

        value={value}

        onChange={onChange}

        placeholder={placeholder}

        required

      />

    </div>

  );

};

export default InputField;

Ovaj kod predstavlja stranicu "Rezervacije" u aplikaciji. Na ovoj stranici se prikazuju rezervacije korisnika, omogućeno je pretraživanje tih rezervacija, a podaci se dohvataju iz baze podataka. Na vrhu stranice se prikazuje naslov "Lista Rezervacija", ispod toga se nalazi polje za unos teksta kako bi se filtrirale rezervacije, a zatim se prikazuje tabela sa detaljima rezervacija.

Stranica koristi hookove poput useState i useEffect za upravljanje stanjem komponente i dohvatanje podataka sa servera. Takođe, koristi se i biblioteka lodash za implementaciju odložene pretrage kako bi se smanjilo opterećenje servera pri svakom unosu teksta.

Komponenta Rezervacije dohvata podatke o rezervacijama sa servera i prikazuje ih u tabeli. Takođe, omogućava korisniku da pretražuje rezervacije na osnovu unetog teksta. Ukoliko se podaci još uvek učitavaju sa servera, prikazuje se poruka "Loading...", a ako dođe do greške, prikazuje se odgovarajuća greška.

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

import axios from 'axios';

import { debounce } from 'lodash';

import RedTabele from './RedTabele';

const Rezervacije = () => {

  const [reservations, setReservations] = useState([]);

  const [filteredReservations, setFilteredReservations] = useState([]);

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

  const [isLoading, setIsLoading] = useState(true);

  const [searchText, setSearchText] = useState('');

  useEffect(() => {

    const token = sessionStorage.getItem('token');

    const config = {

      headers: {

        Authorization: `Bearer ${token}`,

      },

    };

    axios.get('http://127.0.0.1:8000/api/rezervacije', config)

      .then((response) => {

        setReservations(response.data.data);

        setFilteredReservations(response.data.data);

        setIsLoading(false);

      })

      .catch((err) => {

        setError('Greška pri učitavanju rezervacija');

        setIsLoading(false);

      });

  }, []);

  const searchReservations = (text) => {

    setSearchText(text);

    const lowercasedText = text.toLowerCase();

    const filtered = reservations.filter(reservation => {

      return searchInObject(reservation, lowercasedText);

    });

    setFilteredReservations(filtered);

  };

  const debouncedSearch = debounce(searchReservations, 300);

  const handleSearchChange = (e) => {

    debouncedSearch(e.target.value);

  };

  const searchInObject = (obj, searchText) => {

    if (obj === null) return false;

    if (typeof obj === 'object') {

      return Object.values(obj).some(value => searchInObject(value, searchText));

    }

    return String(obj).toLowerCase().includes(searchText);

  };

  if (isLoading) {

    return <div>Loading...</div>;

  }

  if (error) {

    return <div>{error}</div>;

  }

  return (

    <div className='glavniContainer'>

      <h2>Lista Rezervacija</h2>

      <input

        type="text"

        placeholder="Pretraži rezervacije..."

        onChange={handleSearchChange}

      />

      <table>

        <thead>

          <tr>

            <th>ID</th>

            <th>Datum</th>

            <th>Vreme</th>

            <th>Korisnik</th>

            <th>Zaposleni</th>

            <th>Usluga</th>

          </tr>

        </thead>

        <tbody>

          {filteredReservations.map((reservation) => (

            <RedTabele key={reservation.id} reservation={reservation} />

          ))}

        </tbody>

      </table>

    </div>

  );

};

export default Rezervacije;

U ovoj aplikaciji imamo dve korisnicki definisane kuke, useUsluge i useZaposleni, koje se koriste za dohvatanje podataka o uslugama i zaposlenima sa servera. Ove funkcije koriste axios za slanje HTTP zahteva i useState i useEffect hookove za upravljanje stanjem komponenti.

Prva funkcija, useUsluge, dohvata podatke o uslugama i vraća ih kao rezultat. Slično tome, druga funkcija, useZaposleni, dohvata podatke o zaposlenima sa servera i takođe ih vraća kao rezultat.

import axios from "axios";

import { useEffect, useState } from "react";

const useUsluge = () => {

    const [usluge, setUsluge] = useState([]);

    useEffect(() => {

      axios.get('http://127.0.0.1:8000/api/usluge')

        .then((response) => {

          setUsluge(response.data.data);

        })

        .catch((error) => {

          console.error('Greška pri učitavanju usluga', error);

        });

    }, []);

    return [usluge, setUsluge];

  };

  export default useUsluge;

import axios from "axios";

import { useEffect, useState } from "react";

const useZaposleni = () => {

    const [zaposleni, setZaposleni] = useState([]);

    useEffect(() => {

      axios.get('http://127.0.0.1:8000/api/zaposleni')

        .then((response) => {

          setZaposleni(response.data.data);

        })

        .catch((error) => {

          console.error('Greška pri učitavanju zaposlenih', error);

        });

    }, []);

    return [zaposleni, setZaposleni];

  };

  export default useZaposleni;

Komponenta "CreateReservation" u aplikaciji koja omogućava korisnicima da kreiraju nove rezervacije.

Na ovoj stranici, korisnik može da unese podatke o usluzi, datumu, vremenu i zaposlenom za koga želi da napravi rezervaciju. Kada popuni sva obavezna polja i klikne na dugme "Create Reservation", podaci se šalju na server putem HTTP zahteva sa odgovarajućim autorizacionim tokenom koji se dobija iz sesije.

Ako je kreiranje rezervacije uspešno, korisnik će biti preusmeren na stranicu sa listom rezervacija. U slučaju bilo kakve greške prilikom slanja zahteva ili kreiranja rezervacije, greška će biti prijavljena u konzoli, i može se tretirati na odgovarajući način.

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

import axios from 'axios';

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

import InputField from '../../login/InputField';

import useUsluge from '../hooks/useUsluge';

import useZaposleni from '../hooks/useZaposleni';

function CreateReservation( ) {

    const token= sessionStorage.getItem("token");

  const navigate = useNavigate();

  const [formData, setFormData] = useState({

    usluga\_id: '',

    datum: '',

    vreme: '',

    zaposleni\_id: '',

  });

  const [usluge] = useUsluge();

  const [zaposleni] = useZaposleni();

  const handleChange = (e) => {

    const { name, value } = e.target;

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

  };

  const handleSubmit = (e) => {

    e.preventDefault();

    axios

      .post('http://127.0.0.1:8000/api/rezervacije', formData, {

        headers: {

          Authorization: `Bearer ${token}`,

        },

      })

      .then((response) => {

        console.log('Rezervacija uspešno kreirana', response.data);

        navigate('/rezervacije');

      })

      .catch((error) => {

        console.error('Greška pri kreiranju rezervacije', error);

        // Tretirajte grešku pri kreiranju prema potrebi

      });

  };

  return (

    <div className="login-container">

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

        <h2>Create Reservation</h2>

        <div className="input-group">

            <select

                name="usluga\_id"

                value={formData.usluga\_id}

                onChange={handleChange}

            >

                <option value="" disabled>

                Select a service

                </option>

                {usluge.map((usluga) => (

                <option key={usluga.id} value={usluga.id}>

                    {usluga.naziv}

                </option>

                ))}

            </select>

            </div>

        <InputField

          name="datum"

          value={formData.datum}

          onChange={handleChange}

          placeholder="Date"

          type="date"

        />

        <InputField

          name="vreme"

          value={formData.vreme}

          onChange={handleChange}

          placeholder="Time"

          type="time"

        />

       <div className="input-group">

        <select

            name="zaposleni\_id"

            value={formData.zaposleni\_id}

            onChange={handleChange}

        >

            <option value="" disabled>

            Select an employee

            </option>

            {zaposleni.map((z) => (

            <option key={z.id} value={z.id}>

                {z.name}

            </option>

            ))}

        </select>

        </div>

        <div className="options">

          <button type="submit" className="login-button">

            Create Reservation

          </button>

        </div>

      </form>

    </div>

  );

}

export default CreateReservation;

Kod koji je prikazan ispod predstavlja React komponentu nazvanu Usluge, koja je odgovorna za prikaz, dodavanje, uređivanje i brisanje usluga u aplikaciji. Prvo su uvedene neophodne biblioteke, kao što su React za rad sa komponentama, useState i useEffect za upravljanje stanjem i efektima, te axios za slanje HTTP zahteva.

Glavna svrha komponente je da omogući korisnicima pregled svih usluga, kao i dodavanje novih, uređivanje postojećih ili brisanje već postojećih usluga.

Stanje komponente se prati kroz nekoliko useState hook-ova. usluge čuva listu usluga, loading označava da li su podaci o uslugama u procesu učitavanja, formData čuva podatke o trenutnoj formi (naziv, opis, cena), dok editingUsluga prati informaciju o tome da li je neka usluga u fazi uređivanja.

Pomoću useEffect hook-a, komponenta šalje GET zahtev serveru da bi dobavila sve usluge kada se komponenta prvi put montira. Nakon što se podaci uspešno dobave, stanje se ažurira i prikazuje se lista usluga.

Funkcije handleInputChange, handleSubmit, handleEdit, handleCancelEdit i handleDelete su odgovorne za rukovanje promenama u formi, slanje forme, uređivanje, otkazivanje uređivanja i brisanje usluga, redom. Svaka od ovih funkcija šalje odgovarajući HTTP zahtev ka serveru koristeći axios biblioteku.

Kada je loading stanje true, korisniku se prikazuje poruka da se usluge učitavaju, a kada se učitavanje završi, prikazuju se ili lista usluga, ili prazna forma za unos nove usluge.

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

import axios from 'axios';

function Usluge() {

  const [usluge, setUsluge] = useState([]);

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

  const [formData, setFormData] = useState({

    naziv: '',

    opis: '',

    cena: '',

  });

  const [editingUsluga, setEditingUsluga] = useState(null);

  useEffect(() => {

    // Dohvati token iz sessionStorage

    const token = sessionStorage.getItem('token');

    // Kreiraj zaglavlje sa tokenom

    const headers = {

      Authorization: `Bearer ${token}`,

    };

    // Izvrši Axios GET zahtev za dobijanje svih usluga sa dodatim zaglavlje

    axios.get('http://127.0.0.1:8000/api/usluge', { headers })

      .then((response) => {

        setUsluge(response.data.data);

        setLoading(false);

      })

      .catch((error) => {

        console.error('Greška pri dobijanju usluga', error);

        setLoading(false);

      });

  }, []);

  const handleInputChange = (e) => {

    const { name, value } = e.target;

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

  };

  const handleSubmit = (e) => {

    e.preventDefault();

    // Dohvati token iz sessionStorage

    const token = sessionStorage.getItem('token');

    // Kreiraj zaglavlje sa tokenom

    const headers = {

      Authorization: `Bearer ${token}`,

    };

    if (editingUsluga) {

      // Izvrši Axios PUT zahtev za ažuriranje usluge sa dodatim zaglavlje

      axios.put(`http://127.0.0.1:8000/api/usluge/${editingUsluga.id}`, formData, { headers })

        .then((response) => {

          // Ažuriraj uslugu u state

          setUsluge((prevUsluge) => {

            const updatedUsluge = prevUsluge.map((usluga) => {

              if (usluga.id === editingUsluga.id) {

                return response.data.data;

              }

              return usluga;

            });

            return updatedUsluge;

          });

          // Završi uređivanje

          setEditingUsluga(null);

          // Resetuj formu

          setFormData({ naziv: '', opis: '', cena: '' });

        })

        .catch((error) => {

          console.error('Greška pri ažuriranju usluge', error);

        });

    } else {

      // Izvrši Axios POST zahtev za kreiranje nove usluge sa dodatim zaglavlje

      axios.post('http://127.0.0.1:8000/api/usluge', formData, { headers })

        .then((response) => {

          // Dodaj novu uslugu u state

          setUsluge([...usluge, response.data.data]);

          // Resetuj formu

          setFormData({ naziv: '', opis: '', cena: '' });

        })

        .catch((error) => {

          console.error('Greška pri kreiranju usluge', error);

        });

    }

  };

  const handleEdit = (usluga) => {

    setEditingUsluga(usluga);

    // Popuni formu sa podacima usluge koja se uređuje

    setFormData({ naziv: usluga.naziv, opis: usluga.opis, cena: usluga.cena });

  };

  const handleCancelEdit = () => {

    setEditingUsluga(null);

    // Resetuj formu

    setFormData({ naziv: '', opis: '', cena: '' });

  };

  const handleDelete = (id) => {

    // Dohvati token iz sessionStorage

    const token = sessionStorage.getItem('token');

    // Kreiraj zaglavlje sa tokenom

    const headers = {

      Authorization: `Bearer ${token}`,

    };

    // Izvrši Axios DELETE zahtev za brisanje usluge sa dodatim zaglavlje

    axios.delete(`http://127.0.0.1:8000/api/usluge/${id}`, { headers })

      .then(() => {

        // Ukloni uslugu iz state

        setUsluge(usluge.filter((usluga) => usluga.id !== id));

      })

      .catch((error) => {

        console.error('Greška pri brisanju usluge', error);

      });

  };

  if (loading) {

    return <p>Učitavanje usluga...</p>;

  }

  return (

    <div style={{ margin: "5%" }}>

      <h2>Upravljanje uslugama</h2>

      <form onSubmit={handleSubmit}>

        <input

          type="text"

          name="naziv"

          placeholder="Naziv usluge"

          value={formData.naziv}

          onChange={handleInputChange}

        />

        <input

          type="text"

          name="opis"

          placeholder="Opis usluge"

          value={formData.opis}

          onChange={handleInputChange}

        />

        <input

          type="number"

          name="cena"

          placeholder="Cena usluge"

          value={formData.cena}

          onChange={handleInputChange}

        />

        <button type="submit">{editingUsluga ? 'Sačuvaj izmene' : 'Dodaj uslugu'}</button>

        {editingUsluga && <button onClick={handleCancelEdit}>Otkaži</button>}

      </form>

      <table>

        <thead>

          <tr>

            <th>Naziv</th>

            <th>Opis</th>

            <th>Cena</th>

            <th>Akcije</th>

          </tr>

        </thead>

        <tbody>

          {usluge.map((usluga) => (

            <tr key={usluga.id}>

              <td>{usluga.naziv}</td>

              <td>{usluga.opis}</td>

              <td>{usluga.cena}</td>

              <td>

                <button onClick={() => handleDelete(usluga.id)}>Obriši</button>

                <button onClick={() => handleEdit(usluga)}>Uredi</button>

              </td>

            </tr>

          ))}

        </tbody>

      </table>

    </div>

  );

}

export default Usluge;

Komponenta Cenovnik je osmišljena za prikaz i upravljanje cenovnikom usluga u aplikaciji. U njoj su implementirane ključne funkcionalnosti koje omogućavaju korisnicima da pregledaju usluge, sortiraju ih po ceni i valuti, navigiraju kroz stranice sa uslugama, kao i da odaberu željenu valutu za prikaz cena.

Stanje komponente se prati kroz nekoliko useState hook-ova, uključujući currentPage za praćenje trenutne stranice sa uslugama, sortDirection za praćenje smera sortiranja, valute za čuvanje dostupnih valuta, odabranaValuta za praćenje trenutno odabrane valute i konvertovaneCene za čuvanje cena usluga konvertovanih u odabranu valutu.

Korišćenjem useEffect hook-ova, komponenta izvršava određene akcije nakon što se mountuje. Prvi useEffect se koristi za učitavanje dostupnih valuta sa eksternog API-ja, dok drugi useEffect vrši konverziju cena usluga u odabranu valutu.

Usluge se sortiraju u zavisnosti od odabrane opcije sortiranja (rastuće ili opadajuće) i prikazuju se u tabeli. Takođe, omogućena je paginacija kako bi se prikazivao samo određeni broj usluga po stranici, uz mogućnost prelaska na prethodnu ili sledeću stranicu.

Korisnicima se pruža mogućnost odabira valute putem select elementa, pri čemu se cene usluga prikazuju u toj valuti. Opcija za sortiranje omogućava korisnicima da sortiraju usluge po ceni u odabranom smeru (rastuće ili opadajuće).

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

import useUsluge from '../rezervacije/hooks/useUsluge';

function Cenovnik() {

  const [usluge] = useUsluge();

  const [currentPage, setCurrentPage] = useState(1);

  const [sortDirection, setSortDirection] = useState('asc');

  const uslugePerPage = 5;

  const [valute, setValute] = useState([]);

  const [odabranaValuta, setOdabranaValuta] = useState('USD');

  const [konvertovaneCene, setKonvertovaneCene] = useState({});

  useEffect(() => {

    // Učitavanje dostupnih valuta

    fetch('https://api.exchangerate-api.com/v4/latest/USD')

      .then((res) => res.json())

      .then((data) => {

        setValute(Object.keys(data.rates));

      });

  }, []);

  useEffect(() => {

    // Konverzija cena usluga u odabranu valutu

    if (odabranaValuta !== 'USD') {

      fetch(`https://api.exchangerate-api.com/v4/latest/USD`)

        .then((res) => res.json())

        .then((data) => {

          const rate = data.rates[odabranaValuta];

          const konvertovane = usluge.map(usluga => ({

            ...usluga,

            cena: (usluga.cena \* rate).toFixed(2)

          }));

          setKonvertovaneCene(konvertovane);

        });

    } else {

      setKonvertovaneCene(usluge);

    }

  }, [odabranaValuta, usluge]);

  const sortedUsluge = konvertovaneCene.slice().sort((a, b) => {

    if (sortDirection === 'asc') {

      return a.cena - b.cena;

    } else {

      return b.cena - a.cena;

    }

  });

  const indexOfLastUsluga = currentPage \* uslugePerPage;

  const indexOfFirstUsluga = indexOfLastUsluga - uslugePerPage;

  const currentUsluge = sortedUsluge.slice(indexOfFirstUsluga, indexOfLastUsluga);

  const totalPages = Math.ceil(usluge.length / uslugePerPage);

  const handlePageChange = (newPage) => {

    setCurrentPage(newPage);

  };

  const handleSortChange = () => {

    setSortDirection(sortDirection === 'asc' ? 'desc' : 'asc');

  };

  return (

    <div className='glavniContainer'>

      <h2>Cenovnik Usluga</h2>

      <div>

        <label>Odaberite valutu:</label>

        <select value={odabranaValuta} onChange={(e) => setOdabranaValuta(e.target.value)}>

          {valute.map((valuta) => (

            <option key={valuta} value={valuta}>{valuta}</option>

          ))}

        </select>

      </div>

      <button onClick={handleSortChange}>

        Sortiraj po ceni ({sortDirection === 'asc' ? 'Rastuće' : 'Opadajuće'})

      </button>

      <table>

        <thead>

          <tr>

            <th>ID</th>

            <th>Naziv Usluge</th>

            <th>Cena ({odabranaValuta})</th>

            <th>Opis</th>

          </tr>

        </thead>

        <tbody>

          {currentUsluge.map((usluga) => (

            <tr key={usluga.id}>

              <td>{usluga.id}</td>

              <td>{usluga.naziv}</td>

              <td>{usluga.cena}</td>

              <td>{usluga.opis}</td>

            </tr>

          ))}

        </tbody>

      </table>

      <div>

        <button

          onClick={() => handlePageChange(currentPage - 1)}

          disabled={currentPage === 1}

        >

          Prethodna strana

        </button>

        <button

          onClick={() => handlePageChange(currentPage + 1)}

          disabled={currentPage === totalPages}

        >

          Sledeća strana

        </button>

      </div>

    </div>

  );

}

export default Cenovnik;

Komponenta Mojerezervacije se koristi za prikaz rezervacija koje pripraju trenutno prijavljenom zaposlenom. U njoj su implementirane funkcionalnosti koje omogućavaju dobavljanje i prikaz rezervacija, sortiranje po datumu i vremenu, kao i razdvajanje rezervacija prema datumu.

Pomoću useEffect hook-a, komponenta šalje GET zahtev serveru da bi dobavila rezervacije za trenutno prijavljenog korisnika. Dobijeni podaci se sortiraju po datumu i vremenu, nakon čega se ažurira stanje komponente i učitavanje se označava kao završeno.

Ukoliko je u toku učitavanje, korisniku se prikazuje odgovarajuća poruka. Nakon završetka učitavanja, rezervacije se razdvajaju prema datumu, pri čemu se današnje i sutrašnje rezervacije filtriraju iz celokupne liste.

UI komponente prikazuju današnje i sutrašnje rezervacije u odvojenim delovima. Svaka rezervacija se prikazuje u tabeli, sa informacijama o vremenu, korisniku koji je napravio rezervaciju i nazivu usluge koju je rezervisao.

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

import axios from 'axios';

function Mojerezervacije() {

  const [reservations, setReservations] = useState([]);

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

  useEffect(() => {

    // Dohvati token iz sessionStorage

    const token = sessionStorage.getItem('token');

    // Kreiraj zaglavlje sa tokenom

    const headers = {

      Authorization: `Bearer ${token}`,

    };

    // Izvrši Axios GET zahtev za rezervacijama zaposlenog sa dodatim zaglavljem

    axios.get('http://127.0.0.1:8000/api/rezervacije/getReservationsByEmployee', { headers })

      .then((response) => {

        const sortedReservations = response.data.data.sort((a, b) => {

          // Sortiraj rezervacije po datumu i vremenu

          const dateA = new Date(a.datum + ' ' + a.vreme);

          const dateB = new Date(b.datum + ' ' + b.vreme);

          return dateA - dateB;

        });

        setReservations(sortedReservations);

        setLoading(false);

      })

      .catch((error) => {

        console.error('Greška pri dobijanju rezervacija', error);

        setLoading(false);

      });

  }, []);

  if (loading) {

    return <p>Učitavanje rezervacija...</p>;

  }

  // Razdvajamo rezervacije prema datumu

  const todayReservations = reservations.filter((reservation) => {

    const today = new Date();

    const reservationDate = new Date(reservation.datum);

    return reservationDate.toDateString() === today.toDateString();

  });

  const tomorrowReservations = reservations.filter((reservation) => {

    const tomorrow = new Date();

    tomorrow.setDate(tomorrow.getDate() + 1);

    const reservationDate = new Date(reservation.datum);

    return reservationDate.toDateString() === tomorrow.toDateString();

  });

  return (

    <div>

      <h2>Moje Rezervacije</h2>

      <div className="column">

        <h3>Današnje rezervacije</h3>

        <table>

          {/\* Prikazujemo današnje rezervacije \*/}

          {todayReservations.map((reservation) => (

            <tr key={reservation.id}>

              <td>{reservation.vreme}</td>

              <td>{reservation.korisnik.name}</td>

              <td>{reservation.usluga.naziv}</td>

            </tr>

          ))}

        </table>

      </div>

      <div className="column">

        <h3>Sutrašnje rezervacije</h3>

        <table>

          {/\* Prikazujemo sutrašnje rezervacije \*/}

          {tomorrowReservations.map((reservation) => (

            <tr key={reservation.id}>

              <td>{reservation.vreme}</td>

              <td>{reservation.korisnik.name}</td>

              <td>{reservation.usluga.naziv}</td>

            </tr>

          ))}

        </table>

      </div>

    </div>

  );

}

export default Mojerezervacije;

# Link ka github repozitorijumu

<https://github.com/elab-development/internet-tehnologije-projekat-salonlepote_2020_0087>

# Reference

* W3C. (s.d.). HTML i CSS specifikacije. https://www.w3.org/standards/webdesign/htmlcss
* MySQL. (s.d.). Zvanična web stranica MySQL baze podataka. https://www.mysql.com/
* <https://medium.com/@ITesic/uvod-u-react-ekosistem-8ccfad0a1030>
* React. (n.d.). Learn React [Web stranica]. <https://react.dev/learn>
* Laravel. (n.d.). Laravel 10.x Documentation [Web stranica]. https://laravel.com/docs/10.x