Univerzitet u Beogradu

Fakultet organizacionih nauka

Katedra za elektronsko poslovanje

ReactJS

Domaći zadatak 3

|  |  |  |
| --- | --- | --- |
| Ime | Prezime | Broj indeksa |
|  |  |  |
| Link ka Github-u |  | |

Sadržaj

[1 Korisnički zahtev 3](#_Toc117932679)

[2 Implementacija 3](#_Toc117932680)

[3 Korisničko uputstvo 3](#_Toc117932681)

# Korisnički zahtev

Ova web aplikacija pruža korisnicima mogućnost pregleda i upravljanja informacijama vezanim za univerzitete i ispite. Korisnici se mogu prijaviti na sistem, nakon čega dobijaju pristup detaljnim informacijama o različitim univerzitetima i ispitima. Aplikacija omogućava filtriranje univerziteta po državama, kao i pretragu ispita po predmetu i profesoru. Takođe, korisnici mogu pregledati informacije o svojim ispitima, uključujući ocene, ESP bodove i prosečne ocene. Postoji funkcionalnost za sortiranje ispita po ocenama i prikaz samo položenih ispita. Korisnici se mogu odjaviti iz sistema, a navigacija je pojednostavljena kroz intuitivan navigacioni bar.

# Implementacija

Komponenta Univerziteti koristi useState hook za kreiranje nekoliko stanja: univerziteti (niz univerziteta), trenutnaStranica (za praćenje trenutne stranice paginacije), univerzitetaPoStranici (broj univerziteta prikazanih po stranici), ukupnoStranica (ukupan broj stranica, izračunat na osnovu broja univerziteta i broja univerziteta po stranici), i trenutnaDrzava (za filtriranje univerziteta po državi).

Efekat useEffect se koristi za dohvatanje podataka o univerzitetima koristeći axios biblioteku. Ova funkcija se izvršava pri inicijalnom renderovanju komponente. Ukoliko dođe do greške prilikom dohvatanja podataka, greška se beleži u konzoli.

Komponenta takođe sadrži logiku za filtriranje univerziteta na osnovu izabrane države i paginaciju. Filtrirani niz univerziteta se zatim deli na stranice, a korisnik može promeniti trenutnu stranicu koristeći funkciju promeniStranicu.

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

import axios from 'axios';

import './Univerziteti.css';

import Kartica from './Kartica';

function Univerziteti() {

    const [univerziteti, setUniverziteti] = useState([]);

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

    const [univerzitetaPoStranici] = useState(9);

    const ukupnoStranica = Math.ceil(univerziteti.length / univerzitetaPoStranici);

    const [trenutnaDrzava, setTrenutnaDrzava] = useState('');

  useEffect(() => {

    const fetchUniverziteti = async () => {

      try {

        const response = await axios.get('http://universities.hipolabs.com/search');

        setUniverziteti(response.data);

      } catch (error) {

        console.error('Greška prilikom dohvatanja podataka o univerzitetima', error);

      }

    };

    fetchUniverziteti();

  }, []);

    const filtriraniUniverziteti = trenutnaDrzava

    ? univerziteti.filter(uni => uni.country === trenutnaDrzava)

    : univerziteti;

    const trenutniUniverziteti = filtriraniUniverziteti.slice(

      (trenutnaStranica - 1) \* univerzitetaPoStranici,

      trenutnaStranica \* univerzitetaPoStranici

  );

    const promeniStranicu = brojStranice => {

        setTrenutnaStranica(brojStranice);

    };

  return (

    <div style={{margin:"7%"}}>

      <div className="filteri">

                <button onClick={() => setTrenutnaDrzava('France')}>France</button>

                <button onClick={() => setTrenutnaDrzava('Chile')}>Chile</button>

                <button onClick={() => setTrenutnaDrzava('Ukraine')}>Ukraine</button>

                <button onClick={() => setTrenutnaDrzava('United Kingdom')}>United Kingdom</button>

                <button onClick={() => setTrenutnaDrzava('')}>Svi Univerziteti</button>

            </div>

     <div className="paginacija">

        <button onClick={() => promeniStranicu(1)} disabled={trenutnaStranica === 1}>Prva</button>

        <button onClick={() => promeniStranicu(trenutnaStranica - 1)} disabled={trenutnaStranica === 1}>Prethodna</button>

        <button onClick={() => promeniStranicu(trenutnaStranica + 1)} disabled={trenutnaStranica === ukupnoStranica}>Sledeća</button>

        <button onClick={() => promeniStranicu(ukupnoStranica)} disabled={trenutnaStranica === ukupnoStranica}>Poslednja</button>

      </div>

    <div className="univerziteti-container">

      {trenutniUniverziteti.map((uni,index) => (

          <Kartica key={index} uni={uni}></Kartica>

      ))}

       </div>

      <div className="paginacija">

        <button onClick={() => promeniStranicu(1)} disabled={trenutnaStranica === 1}>Prva</button>

        <button onClick={() => promeniStranicu(trenutnaStranica - 1)} disabled={trenutnaStranica === 1}>Prethodna</button>

        <button onClick={() => promeniStranicu(trenutnaStranica + 1)} disabled={trenutnaStranica === ukupnoStranica}>Sledeća</button>

        <button onClick={() => promeniStranicu(ukupnoStranica)} disabled={trenutnaStranica === ukupnoStranica}>Poslednja</button>

      </div>

      </div>

  );

}

export default Univerziteti;

Komponenta Login služi za realizaciju funkcionalnosti prijavljivanja korisnika. Komponenta koristi useState hook iz React-a za praćenje stanja emaila i lozinke, koje korisnik unosi. Takođe, koristi useNavigate hook iz React Router-a za navigaciju nakon uspešne prijave.

Kada korisnik pošalje formu za prijavu, funkcija handleLogin se aktivira. Ova funkcija sprečava standardno ponašanje forme (prevenira osvežavanje stranice), a zatim koristi axios biblioteku za slanje POST zahteva na definisani URL (u ovom slučaju, to je http://127.0.0.1:8000/api/login). Zahtev uključuje email, lozinku i tip korisnika ('student'), koji su neophodni za autentifikaciju u Laravel aplikaciji (kako je navedeno u komentaru).

Ukoliko je prijava uspešna, token pristupa i identifikacioni broj korisnika se čuvaju u sessionStorage, što omogućava očuvanje stanja sesije. Funkcija setToken se koristi za ažuriranje stanja tokena na nivou aplikacije. Nakon toga, korisnik se preusmerava na stranicu '/ispiti'.

U slučaju greške prilikom prijave, greška se beleži u konzoli.

import React, { useState } from 'react';

import axios from 'axios';

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

function Login({setToken}) {

  const [email, setEmail] = useState('');

  const [password, setPassword] = useState('');

  const navigate = useNavigate();

  const handleLogin = async (e) => {

    e.preventDefault();

    try {

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

        email,

        password,

        type:'student'  //ovo moramo da dodajemo jer u laravelu imamo razlicite uloge

      });

      sessionStorage.setItem('token', response.data.access\_token);

      sessionStorage.setItem('auth\_id', response.data.user.id);

      setToken(response.data.access\_token)

      navigate('/ispiti');

    } catch (error) {

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

    }

  };

  return (

    <div className="login-container">

      <img src="https://fon.bg.ac.rs/wp-content/uploads/2023/10/FON-Logo.svg" alt="Logo FONA" className="login-logo" />

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

        <input

          type="email"

          name="email"

          value={email}

          onChange={(e) => setEmail(e.target.value)}

          placeholder="Unesite email"

          required

        />

        <input

          type="password"

          name="password"

          value={password}

          onChange={(e) => setPassword(e.target.value)}

          placeholder="Unesite lozinku"

          required

        />

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

      </form>

    </div>

  );

}

export default Login;

Komponenta "Ispiti", služi za prikaz i upravljanje informacijama o ispitima. Koristi custom hook useIspiti za dohvatanje podataka o ispitima, a useState hook za upravljanje različitim stanjima kao što su ukupno ESP bodovi, prosečna ocena, pretraga, sortiranje i smer sortiranja. Komponenta omogućava sortiranje ispita po ocenama, pretraživanje ispita na osnovu naziva predmeta ili imena profesora, i prikazuje samo položene ispite određenog studenta (identifikovanog preko ID-a iz sessionStorage-a).

Dva useEffect hooka se koriste za manipulaciju podacima. Prvi se aktivira kada se promeni kriterijum sortiranja ili smer sortiranja, a drugi obrađuje ispite pri svakom ažuriranju liste ispita. U drugom useEffect hooku, filtriraju se položeni ispiti studenta sa ocenama većim od 5, izračunava se ukupan broj ESP bodova i prosečna ocena.

JSX deo komponente uključuje polje za pretragu, dugme za sortiranje, tabelu za prikaz ispita i sažetak sa ukupnim ESP bodovima i prosečnom ocenom. Tabela ispita koristi komponentu IspitRed za prikaz svakog ispita.

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

import axios from 'axios';

import IspitRed from './IspitRed';

import './Ispiti.css';

import useIspiti from './useIspiti';

function Ispiti() {

  const [ukupnoESP, setUkupnoESP] = useState(0);

  const [prosecnaOcena, setProsecnaOcena] = useState(0);

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

  const [sortirajPo, setSortirajPo] = useState(null);

  const [sortSmer, setSortSmer] = useState('asc');

  const [ispiti, loading, error,setIspiti] = useIspiti('http://127.0.0.1:8000/api/ispiti');

  const [polozeniIspiti, setPolozeniIspiti] = useState();

  const toggleSortSmer = () => {

    setSortSmer(prevSortSmer => (prevSortSmer === 'asc' ? 'desc' : 'asc'));

    setSortirajPo('ocena');

  };

  useEffect(() => {

    if (sortirajPo === 'ocena') {

      setIspiti(prevIspiti => [...prevIspiti].sort((a, b) => {

        return sortSmer === 'asc' ? a.ocena - b.ocena : b.ocena - a.ocena;

      }));

    }

  }, [sortirajPo, sortSmer]);

    useEffect(() => {

      const studentId = sessionStorage.getItem('auth\_id');

      const polozeniIspitiStudenta = ispiti.filter(ispit =>

        parseInt(ispit.student.id, 10) == parseInt(studentId) && ispit.ocena > 5

      );

      console.log(polozeniIspitiStudenta)

      const ispitiSaOcenomVecomOdPet = polozeniIspitiStudenta.filter(ispit => ispit.ocena > 5);

      const ukupnoESP = ispitiSaOcenomVecomOdPet.reduce((total, ispit) => total + ispit.predmet.esbp, 0);

      const prosecnaOcena = ispitiSaOcenomVecomOdPet.reduce((total, ispit) => total + ispit.ocena, 0) / ispitiSaOcenomVecomOdPet.length;

      setUkupnoESP(ukupnoESP);

      setProsecnaOcena(prosecnaOcena || 0);

      setPolozeniIspiti(polozeniIspitiStudenta)

    }, [ispiti]);

  const filtriraniIspiti = ispit => {

    return (

      ispit.predmet.naziv.toLowerCase().includes(pretraga.toLowerCase()) ||

      ispit.predmet.profesor.ime.toLowerCase().includes(pretraga.toLowerCase()) ||

      ispit.predmet.profesor.prezime.toLowerCase().includes(pretraga.toLowerCase())

    );

  };

  console.log(ispiti)

  return (

    <div className="ispiti-container">

         <div className="controls">

        <input

          type="text"

          className="search-input"

          placeholder="Pretraži predmete ili profesore"

          value={pretraga}

          onChange={(e) => setPretraga(e.target.value)}

        />

        <button onClick={toggleSortSmer}>

          Sortiraj po oceni ({sortSmer === 'asc' ? 'rastuće' : 'opadajuće'})

        </button>

      </div>

      <table className="ispiti-table">

        <thead>

          <tr>

            <th>Datum</th>

            <th>Predmet</th>

            <th>Br. esp</th>

            <th>Profesor</th>

            <th>Ocena</th>

            <th>Opisna ocena</th>

          </tr>

        </thead>

        <tbody>

            { polozeniIspiti && polozeniIspiti.filter(filtriraniIspiti).map((ispit) => (

                <IspitRed key={ispit.id} ispit={ispit} />

            ))}

        </tbody>

      </table>

      <div className="ispiti-summary">

          <p>Ukupan broj ESP bodova: {ukupnoESP}</p>

          <p>Prosečna ocena: {prosecnaOcena}</p>

        </div>

    </div>

  );

}

export default Ispiti;

Komponenta Navbar je dizajnirana da služi kao navigacioni bar za web aplikaciju. Ova komponenta prima dva prop-a: token i setToken. token se koristi za proveru da li je korisnik trenutno prijavljen, dok se setToken koristi za ažuriranje stanja tokena, posebno prilikom odjave korisnika.

Komponenta koristi useNavigate hook iz React Router-a za programsku navigaciju između stranica. Postoji funkcija handleLogout koja se aktivira kada korisnik klikne na opciju za odjavu. Ova funkcija prvo uklanja token iz sessionStorage-a i postavlja stanje tokena na null, a zatim šalje zahtev za odjavu koristeći axios. Nakon uspešnog odgovora servera na zahtev za odjavu, korisnik je preusmeren na početnu stranicu. U slučaju greške, ova se beleži u konzoli.

import React from 'react';

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

import axios from 'axios';

import './Navbar.css';

function Navbar({token,setToken}) {

  const navigate = useNavigate();

  const handleLogout = async () => {

    setToken(null);

    sessionStorage.removeItem('token');

    try {

      await axios.post('http://127.0.0.1:8000/api/logout', {}, {

        headers: {

          'Authorization': `Bearer ${token}`

        }

      });

      navigate('/');

    } catch (error) {

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

    }

  };

  return (

    <nav className="navbar">

      <img src="https://fon.bg.ac.rs/wp-content/uploads/2023/10/FON-Logo.svg" alt="Logo FONA" className="navbar-logo" />

      <ul className="navbar-nav">

      {token==null ?

        <>

            <li className="nav-item" onClick={() => navigate('/')}>

              Pocetna

            </li>

            <li className="nav-item" onClick={() => navigate('/univerziteti')}>

              Univerziteti

            </li>

            <li className="nav-item" onClick={() => navigate('/login')}>

              Login

            </li>

          </>

        : <>

          <li className="nav-item" onClick={() => navigate('/')}>

            Pocetna

          </li>

          <li className="nav-item" onClick={handleLogout}>

              Odjava

            </li>

      </>

      }

      </ul>

    </nav>

  );

}

export default Navbar;

# Korisničko uputstvo

Kada se aplikacija otvori, otvara se početna stranica kao na slici ispod

Slika na kojoj se nalazi tekst, snimak ekrana, Font

Opis je automatski generisan

Korisnici mogu da se uloguju kao studenti popunjavanjem forme kao na slici ispod

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

Opis je automatski generisan

Odabirom opcije univerziteti iz navigacionog menija, korisicni imaju mogućnost pregleda svih univerziteta u svetu

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

Opis je automatski generisan

Stranica na kojoj korisnici mogu da vide ispite koje su polozili do sada, ukupan broj esp poena, kao i prosecnu ocenu

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

Opis je automatski generisan