Univerzitet u Beogradu  
Fakultet organizacionih nauka  
Katedra za elektronsko poslovanje

Web za studentsku službu

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 za studentsku službu. Potrebno je obezbediti sledeće uloge: student, profesor I neulogovani korisnik.

Korisnici koji nisu ulogovani dobijaju pristup detaljnim informacijama o različitim univerzitetima u svetu, uz mogućnost filtriranja univerziteta po državama. Za korisnike koji nisu ulogovani, dostupan je i kviz za testiranje znanja, što doprinosi interaktivnosti i angažovanju korisnika sa sistemom.

Studentima, aplikacija takođe omogućava da pregledaju ključne informacije o svojim ispitima, uključujući ocene, ESP bodove i prosečne ocene, uz napredne opcije sortiranja ispita po ocenama i filtriranja kako bi se prikazali samo položeni ispiti.

Profesori koji su prijavljeni na sistem mogu kreirati, brisati, menjati i dodavati predmete i ocene studentima. Studenti, s druge strane, imaju mogućnost da generišu PDF potvrde o svom statusu redovnog studenta.

# Opis sistema

## 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 predmeta

Slučaj korišćenja: Kreiranje predmeta

Akteri:

* Predavač
* Sistem (program)

Preduslov:

* Sistem je uključen, a predavač ima pristup funkcionalnostima za kreiranje novih predmeta.

Osnovni scenario:

1. Predavač bira opciju za kreiranje novog predmeta unutar sistema.
2. Sistem prikazuje formu za unos podataka o predmetu, uključujući naziv predmeta, opis, broj ESP bodova, i druge relevantne informacije.
3. Predavač unosi tražene informacije o predmetu u formu.
4. Predavač pregleda unete podatke i proverava da li su sve informacije ispravne i potpune.
5. Predavač potvrđuje unos klikom na dugme "Kreiraj predmet" ili slično.
6. Sistem proverava unete podatke na validnost i potpunost.
7. Ukoliko su svi podaci ispravni, sistem kreira novi predmet i registruje ga u bazi.
8. Sistem obaveštava predavača porukom: "Novi predmet je uspešno kreiran."

### Brisanje predmeta

Slučaj korišćenja: Brisanje predmeta

Akteri:

* Predavač (ili administrator, zavisno od ovlašćenja unutar sistema)
* Sistem (program)

Preduslov:

* Sistem je uključen, a predavač/administrator ima pristup funkcionalnostima za brisanje predmeta.
* Predmet koji treba obrisati postoji u sistemu.

Osnovni scenario:

1. Predavač/administrator pristupa delu sistema gde se upravlja predmetima (npr. lista predmeta, administrativni panel).
2. Predavač/administrator locira predmet koji želi da obriše, možda koristeći funkciju pretrage ili navigaciju kroz listu predmeta.
3. Predavač/administrator odabira opciju za brisanje predmeta. Ovo može biti dugme ili link pored naziva predmeta.
4. Sistem prikazuje dijalog za potvrdu brisanja, tražeći od predavača/administratora da potvrdi svoju nameru da obriše predmet, obično sa porukom poput "Da li ste sigurni da želite da obrišete ovaj predmet?".
5. Predavač/administrator potvrđuje nameru za brisanjem, klikom na "Da" ili sličnu opciju.
6. Sistem briše predmet iz baze podataka.
7. Sistem obaveštava predavača/administratora da je predmet uspešno obrisan, obično porukom poput "Predmet je uspešno obrisan."

## Opis arihtekture aplikacije

Naša aplikacija je razvijena koristeći moderni web razvojni stack koji kombinuje Laravel, React i Axios, omogućavajući brz i efikasan razvoj kompleksnih web aplikacija. Ovaj segment detaljno objašnjava kako se ove tehnologije međusobno dopunjuju u arhitekturi naše aplikacije.

Laravel je odabran kao backend okvir zbog svoje visoke performanse, sigurnosti i lakoće korišćenja. Služi kao temelj naše aplikacije, upravljajući bazom podataka, autentifikacijom korisnika, logikom aplikacije i API endpointima. Laravelova arhitektura MVC (Model-View-Controller) omogućava jasnu separaciju logike i prezentacije, čime se olakšava održavanje koda i unapređenje aplikacije. API resursi su kreirani koristeći Laravel, što omogućava aplikaciji da komunicira sa frontendom putem JSON odgovora, pružajući efikasnu razmenu podataka.

Za izgradnju korisničkog interfejsa koristimo React, JavaScript biblioteku za izgradnju korisničkih interfejsa, koja nam omogućava da kreiramo dinamične i interaktivne web stranice. React se koristi za izgradnju komponenata koje definišu elemente i logiku korisničkog interfejsa, omogućavajući brzu i efikasnu izradu single-page aplikacija (SPA). Korišćenjem Reacta, aplikacija može da ažurira i prikazuje pravilne komponente na korisničkom interfejsu u realnom vremenu bez potrebe za ponovnim učitavanjem stranice, što znatno poboljšava korisničko iskustvo.

Za komunikaciju između React frontend-a i Laravel backend-a koristimo Axios, popularnu JavaScript biblioteku za izvršavanje HTTP zahteva. Axios nam omogućava da jednostavno upravljamo zahtevima i odgovorima između klijenta i servera, uključujući konfigurisanje zahteva, prenos tokena za autentifikaciju i obradu odgovora servera. Zahvaljujući Axiosu, naša aplikacija može efikasno da dohvati podatke, šalje podatke na server i implementira kompleksne funkcionalnosti poput CRUD operacija (Create, Read, Update, Delete) na siguran i pouzdan način.

## Opis procesa slučajeva korišćenja

### Dijagram sekvenci – prijava korisnika

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

Opis je automatski generisan

Slika 1 ds - prijava korisnika

### Dijagram sekvenci – kreiranje predmeta

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

Opis je automatski generisan

Slika 2 ds -kreiranje predmeta

### Dijagram sekvenci – brisanje predmeta

Slika na kojoj se nalazi tekst, rukopis, Lepljiva beleška, Font

Opis je automatski generisan

Slika 3 ds -brisanje predmeta

## Model podataka

Model "User" je opšti model za korisnike sistema, koji može uključivati administratore, studente, profesore i druge korisnike. Ovaj model sadrži podatke za autentifikaciju i autorizaciju, kao što su korisnička imena, lozinke i uloge.

Model "Student" drži informacije o studentima, kao što su njihova imena, brojevi indeksa, i možda kursevi koje pohađaju.

Model "Profesor" sadrži informacije o profesorima, kao što su njihova imena, titule, i predmeti koje predaju.

Model "Predmet" se odnosi na kurseve ili predmete u obrazovnom sistemu. Ovaj model sadrži informacije kao što su naziv predmeta, opis, i možda informacije o profesoru koji predaje.

Model "Ispit" uključuje osnovne informacije o ispitima, kao što su ID studenta, ID predmeta, datum, ocena i opisna ocena. Postoje veze ka modelima "Student" i "Predmet", što ukazuje na to da svaki ispit pripada određenom studentu i predmetu.

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

Opis je automatski generisan

Slika 4 dijagram klasa

## Specifikacija REST API-ja

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

|  |  |
| --- | --- |
| Opis funkcije | Prikaz svih studenata |
| HTTP metoda | GET |
| URL | /api/studenti |
| URL parametri | nema |
| HTTP body parametri | nema |
| Format HTTP body parametara | JSON |
| Izlazni parametri | {      "data": [          {              "id": 1,              "ime": "Jamal",              "prezime": "Klocko",              "email": "nicolas.imelda@example.com",              "jmbg": "76817481353",              "adresa": "4113 Giovanny Drive Apt. 693\nTorpberg, AL 08097",              "kontakt": "+1-443-374-1399",              "godina": 2020,              "broj": 440,              "prosek": 8.5,              "esbp": 62          },          {              "id": 2,              "ime": "Marie",              "prezime": "Zulauf",              "email": "allan68@example.org",              "jmbg": "66595001342",              "adresa": "5843 Maya Lock Apt. 855\nKatarinaville, VA 51025-6434",              "kontakt": "501-834-0473",              "godina": 2019,              "broj": 580,              "prosek": 9.01,              "esbp": 138          },  …  **}** |
| Format izlaznih parametara | Application/json |
|  |  |
| Opis funkcije | Kreiranje studenta |
| HTTP metoda | POST |
| URL | /api/studenti |
| URL parametri | ?ime=nevena&prezime=Dukic&email=nevena@gmail.com&password=nevenadukic&jmbg=7412589631236&adresa=adresa1&kontakt=0624584154&godina=2020&broj=0426&prosek=8.75&esbp=200 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "data": {          "id": 21,          "ime": "nevena",          "prezime": "Dukic",          "email": "nevena@gmail.com",          "jmbg": "7412589631236",          "adresa": "adresa1",          "kontakt": "0624584154",          "godina": "2020",          "broj": "0426",          "prosek": "8.75",          "esbp": "200"      }  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Ažuriranje studenta |
| HTTP metoda | PUT |
| URL | /api/studenti |
| URL parametri | /21?ime=nevena&prezime=Dukic&email=nevena@gmail.com&password=nevenadukic&jmbg=7412589631236&adresa=adresa1&kontakt=0624584154&godina=2020&broj=0426&prosek=8.75&esbp=220 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "data": {          "id": 21,          "ime": "nevena",          "prezime": "Dukic",          "email": "nevena@gmail.com",          "jmbg": "7412589631236",          "adresa": "adresa1",          "kontakt": "0624584154",          "godina": "2020",          "broj": "0426",          "prosek": "8.75",          "esbp": "220"      }  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Brisanje studenta |
| HTTP metoda | DELETE |
| URL | /api/studenti |
| URL parametri | 12 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "message": "Student successfully deleted"  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Prikaz svih predmeta |
| HTTP metoda | GET |
| URL | /api/predmeti |
| URL parametri | nema |
| HTTP body parametri | nema |
| Format HTTP body parametara | JSON |
| Izlazni parametri | {      "data": [          {              "id": 1,              "naziv": "voluptatibus",              "esbp": 5,              "semestar": 3,              "profesor": {                  "id": 2,                  "ime": "Andre",                  "prezime": "Witting",                  "email": "cary24@example.org",                  "titula": "docent",                  "katedra": "deserunt",                  "kabinet": "829a",                  "konsultacije": "Tuesday 15:06"              },              "tip": "obavezan"          },          {              "id": 2,              "naziv": "quasi",              "esbp": 7,              "semestar": 1,              "profesor": {                  "id": 5,                  "ime": "Hardy",                  "prezime": "Cassin",                  "email": "goodwin.virgie@example.com",                  "titula": "red. prof.",                  "katedra": "nisi",                  "kabinet": "014a",                  "konsultacije": "Tuesday 12:15"              },              "tip": "obavezan"          }, |
| Format izlaznih parametara | Application/json |
|  |  |
| Opis funkcije | Kreiranje predmeta |
| HTTP metoda | POST |
| URL | /api/predmeti |
| URL parametri | ? naziv=P1&esbp=5&semestar=1&profesor\_id=1&tip=obavezan |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "data": {          "id": 11,          "naziv": "P1",          "esbp": "5",          "semestar": "1",          "profesor": {              "id": 1,              "ime": "Nathaniel",              "prezime": "Mante",              "email": "draynor@example.com",              "titula": "asistent",              "katedra": "exercitationem",              "kabinet": "713a",              "konsultacije": "Friday 19:41"          },          "tip": "obavezan"      }  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Ažuriranje predmeta |
| HTTP metoda | PUT |
| URL | /api/ predmeti |
| URL parametri | 11?naziv=Programiranje1&esbp=5&semestar=1&profesor\_id=1&tip=obavezan |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "data": {          "id": 11,          "naziv": "Programiranje1",          "esbp": "5",          "semestar": "1",          "profesor": {              "id": 1,              "ime": "Nathaniel",              "prezime": "Mante",              "email": "draynor@example.com",              "titula": "asistent",              "katedra": "exercitationem",              "kabinet": "713a",              "konsultacije": "Friday 19:41"          },          "tip": "obavezan"      }  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Brisanje predmeta |
| HTTP metoda | DELETE |
| URL | /api/ predmeti |
| URL parametri | 11 |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "message": "Predmet successfully deleted"  } |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Prijava korisnika |
| HTTP metoda | POST |
| URL | /api/login |
| URL parametri | ?type=student&email=nicolas.imelda@example.com&password=password |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | {      "access\_token": "5|8J7Y7QNNQEpdHEBUdFyr3MwTtwdftNY8YzKku0jg",      "token\_type": "Bearer",      "user": {          "id": 1,          "ime": "Jamal",          "prezime": "Klocko",          "email": "nicolas.imelda@example.com",          "jmbg": "76817481353",          "adresa": "4113 Giovanny Drive Apt. 693\nTorpberg, AL 08097",          "kontakt": "+1-443-374-1399",          "godina": 2020,          "broj": 440,          "prosek": 8.5,          "esbp": 62,          "created\_at": "2024-01-09T21:03:48.000000Z",          "updated\_at": "2024-01-09T21:03:48.000000Z"      }  } |
| 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": "Logged out successfully"  } |
| Format izlaznih parametara | application/json |

# Opis tehnologija korišćenih u aplikaciji

U razvoju naše web aplikacije koristili smo niz savremenih tehnologija koje nam omogućavaju da izgradimo efikasnu, sigurnu i skalabilnu platformu. 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 PHP web aplikacijski okvir koji je odabran zbog svoje elegantne sintakse, robustnih funkcionalnosti i snažne zajednice. Laravel nudi bogat set alata i biblioteka koje olakšavaju razvoj, uključujući Eloquent ORM za rad sa bazom podataka, middleware za filtriranje HTTP zahteva, i Blade templating engine za efikasniju obradu pogleda. Laravel takođe pruža ugrađenu podršku za autentifikaciju, autorizaciju, rutiranje i sesije, čime se ubrzava razvoj sigurnih i visoko performantnih web aplikacija.

### React

React je deklarativna, efikasna i fleksibilna 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.

# 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

Slika 5 početna

Korisnici koji nisu ulogovani na system imaju mogućnost testiranja svog znanja preko kratkog kviza.

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

Opis je automatski generisan

Slika 6 kviz

Takođe, 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

Slika 7 univerziteti

Korisnici mogu da se uloguju na sistem. Prilikom prijave se bira tip korisnika – profesor ili student.

Slika na kojoj se nalazi tekst, snimak ekrana, Font, Operativni sistem

Opis je automatski generisan

Slika 8 login forma

Kada se uloguje korisnik koji je student, otvara mu se stranica koja izgleda kao na slici ispod.

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

Opis je automatski generisan

Slika 9 prikaz studentske stranice

Na toj stranici korisnici mogu da vide svoje položene ispite, prosek I broj ostvarenih esp poena. Klikom na dugme Kreiraj PDF potvrdu preuzima se pdf fajl koji predstavlja potrvrdu o redovnom studiranju.

Slika na kojoj se nalazi tekst, snimak ekrana, Font

Opis je automatski generisan

Slika 10 potvrda o redovnom studiranju

Kada se na system prijavi professor njemu se prikazuju svi predmeti koje on predaje, kaon a slici ispod.

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

Opis je automatski generisan

Slika 11 prikaz stranice za profesora

Klikom na dugme dodaj novi predmet, korisnik ima mogućnost kreiranja predmeta.

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

Opis je automatski generisan

Slika 12 kreiranje predmeta

Klikom na dugme Detalji, profesoru se prikazuje statistika u vezi odabranog predmeta.

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

Opis je automatski generisan

Slika 13 statistika predmeta

Klikom na dugme unesi ocene, professor može da unese ocenu odabranom student.

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

Opis je automatski generisan

Slika 14 unos ocene

# Prikaz delova koda

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;

Komponenta Profesor u React aplikaciji predstavlja složenu strukturu koja omogućava upravljanje predmetima i unos ocena studenata. Na početku, korišćenjem useState hook-a, definisani su različiti delovi stanja koji čuvaju informacije o predmetima, formama, modalima i studentima, kao i indikatore za učitavanje i selektovane elemente. useEffect hook je korišćen za inicijalno dohvatanje podataka o predmetima i studentima koristeći axios za asinhronu komunikaciju sa serverom. Ovi podaci se dohvaćaju odmah nakon što se komponenta montira, što osigurava da su informacije dostupne odmah nakon učitavanja komponente.

Za navigaciju unutar aplikacije koristi se useNavigate hook iz React Router-a, omogućavajući preusmeravanje na detalje predmeta ili druge stranice unutar aplikacije. Funkcionalnosti za brisanje, uređivanje i dodavanje novih predmeta realizovane su kroz asinhrone funkcije koje komuniciraju sa serverom, koristeći token za autorizaciju dobijen iz sesije.

Komponenta takođe omogućava upravljanje formama za dodavanje ili ažuriranje predmeta, kao i unos ocena za studente. Forme su dizajnirane tako da se pojavljuju kao modalni prozori, čime se korisniku pruža interaktivni i pregledni način za unos podataka. Promene unutar formi rukovane su kroz handleChange funkcije, koje ažuriraju stanje formi sa unetim vrednostima.

Kada je reč o unosu ocena, omogućeno je otvaranje posebnog modalnog prozora za svaki predmet, gde profesor može izabrati studenta i uneti ocene. Ovaj proces takođe uključuje asinhronu komunikaciju sa serverom za upis ocena u bazu podataka. U slučaju uspešnog unosa, modal se zatvara i stanje formi se resetuje.

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

import axios from 'axios';

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

function Profesor() {

  const [predmeti, setPredmeti] = useState([]);

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

  const [formData, setFormData] = useState({

    naziv: '',

    esbp: '',

    semestar: '',

    tip: 'obavezan',

  });

  const [isFormOpen, setIsFormOpen] = useState(false);

  const [selectedPredmetId, setSelectedPredmetId] = useState(null);

  const [isGradeModalOpen, setIsGradeModalOpen] = useState(false);

  const [students, setStudents] = useState([]);

  const [selectedPredmetForGrade, setSelectedPredmetForGrade] = useState(null); // Updated state

  const [gradeFormData, setGradeFormData] = useState({

    student\_id: '',

    predmet\_id: '',

    datum: '',

    ocena: '',

    opisnaOcena: '',

  });

  useEffect(() => {

    const fetchPredmeti = async () => {

      try {

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

        const headers = {

          'Authorization': `Bearer ${token}`,

        };

        const response = await axios.get('http://127.0.0.1:8000/api/predmeti/profesor', { headers });

        setPredmeti(response.data.data);

        setLoading(false);

      } catch (error) {

        console.error('Greška prilikom dobijanja predmeta:', error);

        setLoading(false);

      }

    };

    fetchPredmeti();

  }, []);

  const navigate = useNavigate();

  const handleDetaljiClick = (predmetId) => {

    navigate(`/profesor/predmet/${predmetId}`);

  };

  const handleDeleteClick = async (predmetId) => {

    try {

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

      const headers = {

        'Authorization': `Bearer ${token}`,

      };

      await axios.delete(`http://127.0.0.1:8000/api/predmeti/${predmetId}`, { headers });

      setPredmeti(predmeti.filter((predmet) => predmet.id !== predmetId));

    } catch (error) {

      console.error('Greška prilikom brisanja predmeta:', error);

    }

  };

  const handleEditClick = (predmetId) => {

    const selectedPredmet = predmeti.find((predmet) => predmet.id === predmetId);

    setFormData({

      naziv: selectedPredmet.naziv,

      esbp: selectedPredmet.esbp,

      semestar: selectedPredmet.semestar,

      tip: selectedPredmet.tip,

    });

    setSelectedPredmetId(predmetId);

    setIsFormOpen(true);

  };

  const handleCancelEdit = () => {

    setFormData({

      naziv: '',

      esbp: '',

      semestar: '',

      tip: 'obavezan',

    });

    setSelectedPredmetId(null);

    setIsFormOpen(false);

  };

  const handleUpdate = async () => {

    try {

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

      const headers = {

        'Authorization': `Bearer ${token}`,

      };

      await axios.put(`http://127.0.0.1:8000/api/predmeti/${selectedPredmetId}`, formData, { headers });

      const updatedPredmeti = predmeti.map((predmet) => {

        if (predmet.id === selectedPredmetId) {

          return {

            ...predmet,

            ...formData,

          };

        }

        return predmet;

      });

      setPredmeti(updatedPredmeti);

      setFormData({

        naziv: '',

        esbp: '',

        semestar: '',

        tip: 'obavezan',

      });

      setSelectedPredmetId(null);

      setIsFormOpen(false);

    } catch (error) {

      console.error('Greška prilikom ažuriranja predmeta:', error);

    }

  };

  const handleChange = (e) => {

    const { name, value } = e.target;

    setFormData({

      ...formData,

      [name]: value,

    });

  };

  const handleSubmit = async (e) => {

    e.preventDefault();

    try {

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

      const headers = {

        'Authorization': `Bearer ${token}`,

      };

      const response = await axios.post('http://127.0.0.1:8000/api/predmeti', formData, { headers });

      setPredmeti([...predmeti, response.data.data]);

      setFormData({

        naziv: '',

        esbp: '',

        semestar: '',

        tip: 'obavezan',

      });

      setIsFormOpen(false);

    } catch (error) {

      console.error('Greška prilikom dodavanja predmeta:', error);

    }

  };

  const fetchStudents = async () => {

    try {

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

      const headers = {

        'Authorization': `Bearer ${token}`,

      };

      const response = await axios.get('http://127.0.0.1:8000/api/studenti', { headers });

      setStudents(response.data.data);

    } catch (error) {

      console.error('Greška prilikom dobijanja studenata:', error);

    }

  };

  useEffect(() => {

    fetchStudents();

  }, []);

  const openGradeModal = (predmetId) => {

    setSelectedPredmetForGrade(predmetId);

    setIsGradeModalOpen(true);

  };

  const handleSubmitGrade = async (e) => {

    e.preventDefault();

    try {

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

      const headers = {

        'Authorization': `Bearer ${token}`,

      };

      gradeFormData.predmet\_id=selectedPredmetForGrade;

      console.log(selectedPredmetForGrade)

      const response = await axios.post('http://127.0.0.1:8000/api/ispiti', gradeFormData, { headers });

      console.log(response)

      // Handle successful submission (you can update UI or close the modal here)

      setGradeFormData({

        student\_id: '',

        predmet\_id: '',

        datum: '',

        ocena: '',

        opisnaOcena: '',

      });

      setSelectedPredmetId('');

      setIsGradeModalOpen(false);

    } catch (error) {

      console.error('Greška prilikom unosa ocene:', error);

    }

  };

  const handleGradeFormChange = (e) => {

    const { name, value } = e.target;

    setGradeFormData({

      ...gradeFormData,

      [name]: value,

    });

  };

  if (loading) {

    return <p>Učitavanje...</p>;

  }

  return (

    <div className="predmeti-container">

      <button onClick={() => setIsFormOpen(true)}>Dodaj novi predmet</button>

      {isFormOpen && (

        <div className="modal">

          <form onSubmit={handleSubmit}>

            <div>

              <label>Naziv:</label>

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

            </div>

            <div>

              <label>ESBP:</label>

              <input type="number" name="esbp" value={formData.esbp} onChange={handleChange} required />

            </div>

            <div>

              <label>Semestar:</label>

              <input type="number" name="semestar" value={formData.semestar} onChange={handleChange} required />

            </div>

            <div>

              <label>Tip:</label>

              <select name="tip" value={formData.tip} onChange={handleChange} required>

                <option value="obavezan">Obavezan</option>

                <option value="izborni">Izborni</option>

              </select>

            </div>

            {selectedPredmetId ? (

              <div>

                <button type="button" onClick={handleUpdate}>Ažuriraj predmet</button>

                <button type="button" onClick={handleCancelEdit}>Otkaži</button>

              </div>

            ) : (

              <button type="submit">Dodaj predmet</button>

            )}

            <button type="button" onClick={() => setIsFormOpen(false)}>Zatvori</button> {/\* Dugme za zatvaranje modala \*/}

          </form>

        </div>

      )}

 {isGradeModalOpen && (

        <div className="modal">

          <form onSubmit={handleSubmitGrade}>

            <div>

              <label>Student:</label>

              <select

                name="student\_id"

                value={gradeFormData.student\_id}

                onChange={handleGradeFormChange}

                required

              >

                <option value="">Izaberite studenta</option>

                {students.map((student) => (

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

                    {student.ime} {student.prezime}

                  </option>

                ))}

              </select>

            </div>

            <div>

              <label>Ocena:</label>

              <input

                type="number"

                name="ocena"

                value={gradeFormData.ocena}

                onChange={handleGradeFormChange}

                required

              />

            </div>

            <div>

              <label>Datum:</label>

              <input

                type="date"

                name="datum"

                value={gradeFormData.datum}

                onChange={handleGradeFormChange}

                required

              />

            </div>

            <div>

              <label>Opisna ocena:</label>

              <input

                type="text"

                name="opisnaOcena"

                value={gradeFormData.opisnaOcena}

                onChange={handleGradeFormChange}

              />

            </div>

            <button type="submit">Unesi ocenu</button>

            <button type="button" onClick={() => setIsGradeModalOpen(false)}>

              Zatvori

            </button>

          </form>

        </div>

      )}

      <h1>Moji predmeti:</h1>

      <table className="predmeti-table">

        <thead>

          <tr>

            <th>ID</th>

            <th>Naziv</th>

            <th>Akcije</th>

            <th>Obriši</th>

            <th>Unesi ocene</th>

          </tr>

        </thead>

        <tbody>

          {predmeti.map((predmet) => (

            <tr key={predmet.id}>

              <td>{predmet.id}</td>

              <td>{predmet.naziv}</td>

              <td>

                <button onClick={() => handleDetaljiClick(predmet.id)}>Detalji</button>

                <button onClick={() => handleEditClick(predmet.id)}>Uredi</button>

              </td>

              <td>

                <button onClick={() => handleDeleteClick(predmet.id)}>Obriši</button>

              </td>

              <td>

                <button onClick={() => openGradeModal(predmet.id)}>Unesi ocene</button> {/\* Button to open the grade modal for this predmet \*/}

              </td>

            </tr>

          ))}

        </tbody>

      </table>

    </div>

  );

}

export default Profesor;

Ovaj React komponent nazvana "Kviz" je implementacija jednostavnog kviza koji koristi React Hooks, uključujući useState i useEffect, za upravljanje stanjem aplikacije. Na početku, komponenta inicijalizuje nekoliko stanja: questions za čuvanje pitanja kviza, currentIndex za praćenje trenutnog pitanja na koje korisnik odgovara, score za zabeležavanje broja tačnih odgovora, i showScore za kontrolu prikaza rezultata kviza.

Kada se komponenta montira (useEffect hook sa praznim nizom zavisnosti osigurava da se efekat pokrene samo jednom, pri prvom renderovanju), izvršava se asinhrona funkcija fetchQuestions. Ova funkcija koristi fetch API za dobavljanje pitanja iz javno dostupnog API-ja. Preuzeta pitanja se transformišu tako da se korektan odgovor meša sa netačnim odgovorima, čime se osigurava nasumično raspoređivanje opcija odgovora. U slučaju greške prilikom dohvatanja pitanja, zabeležiće se odgovarajuće poruke o grešci.

Kada korisnik odabere odgovor, funkcija handleAnswer se poziva sa odabranim odgovorom kao argumentom. Funkcija proverava da li je odabrani odgovor tačan upoređujući ga sa korektnim odgovorom trenutnog pitanja. Ukoliko je odgovor tačan, rezultat se povećava. Nakon toga, proverava se da li postoje još pitanja za odgovor; ako da, prikazuje se sledeće pitanje, a ako ne, prikazuje se rezultat kviza.

Komponenta vizuelno prikazuje kviz kroz JSX. Ako je kviz završen (showScore je true), prikazuje se rezultat korisnika. U suprotnom, prikazuje se trenutno pitanje sa opcijama odgovora. Svako pitanje i odgovori se dinamički generišu na osnovu trenutnog stanja questions i currentIndex.

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

const Kviz = () => {

  const [questions, setQuestions] = useState([]);

  const [currentIndex, setCurrentIndex] = useState(0);

  const [score, setScore] = useState(0);

  const [showScore, setShowScore] = useState(false);

  useEffect(() => {

    const fetchQuestions = async () => {

      try {

        const res = await fetch('https://opentdb.com/api.php?amount=5&category=18&difficulty=medium&type=multiple');

        const data = await res.json();

        if (data.results && Array.isArray(data.results)) {

          setQuestions(data.results.map((q) => ({

            ...q,

            answers: [q.correct\_answer, ...q.incorrect\_answers].sort(() => Math.random() - 0.5),

          })));

        } else {

          // Ovde možete postaviti neku vrstu greške ili upozorenja ako nema rezultata

          console.error('No results found');

        }

      } catch (error) {

        // Rukovanje greškama ako dođe do problema sa mrežom ili API-jem

        console.error('Failed to fetch questions', error);

      }

    };

    fetchQuestions();

  }, []);

  const handleAnswer = (answer) => {

    if (answer === questions[currentIndex].correct\_answer) {

      setScore(score + 1);

    }

    const nextQuestion = currentIndex + 1;

    if (nextQuestion < questions.length) {

      setCurrentIndex(nextQuestion);

    } else {

      setShowScore(true);

    }

  };

  return (

    <div className="quiz-container">

      {showScore ? (

        <div className="score-section">

          Score: {score} out of {questions.length}

        </div>

      ) : (

        <>

          {questions.length > 0 && (

            <div className="question-section">

              <div className="question-count">

                <span>Question {currentIndex + 1}</span>/{questions.length}

              </div>

              <div className="question-text">{questions[currentIndex].question}</div>

              <div className="answer-section">

                {questions[currentIndex].answers.map((answer, index) => (

                  <button key={index} onClick={() => handleAnswer(answer)}>

                    {answer}

                  </button>

                ))}

              </div>

            </div>

          )}

        </>

      )}

    </div>

  );

};

export default Kviz;

# Link ka github repozitorijumu

<https://github.com/elab-development/internet-tehnologije-projekat-studentskasluzba_2020_0426>

# 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