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

Potrebno je kreirati aplikaciju za organizaciju i upravljanje dokumentima na jednostavan i efikasan način. Korisnici imaju različite funkcionalnosti u zavisnosti da li su prijavljeni ili ne.

Neulogovani korisnici mogu da pristupe samo osnovnim funkcionalnostima aplikacije. Na početnoj stranici, mogu da vide linkove ka "Login" i "Register" stranicama kako bi se prijavili ili registrovali. Takođe, mogu pregledati dokumente dostupne javno, ali nemaju pristup dodatnim opcijama kao što su uređivanje ili preuzimanje dokumenata.

Prijavljeni korisnici imaju pristup punom spektru funkcionalnosti aplikacije. Nakon prijavljivanja, korisnici mogu da pristupe raznim sekcijama aplikacije, uključujući "IT dokumente" za specifične tehničke informacije, "Upload" za dodavanje novih dokumenata, i "Dokumenti" za pregled, pretragu, uređivanje i preuzimanje postojećih dokumenata. Korisnici mogu da pregledaju listu dokumenata sa osnovnim informacijama kao što su naslov, autor i kategorija, kao I da pretraže dokumente po kategoriji i oznakama kako bi suzili rezultate pretrage. Prijavljeni korisnici mogu da uređuju postojeće dokumente, unoseći nove informacije ili izmene u postojeće, kao I da preuzmu dokumente sa servera i sačuvaju ih na svojim računarima.Tabela sa dokumentima je paginirana, omogućavajući korisnicima da se kreću kroz više dokumenata sa jednostavnim navigacijama.

# Implementacija

Komponenta Register je odgovorna za proces registracije korisnika u web aplikaciji. Ova komponenta sadrži formu sa poljima za unos osnovnih informacija, kao što su ime, email adresa, lozinka, datum rođenja i biografija korisnika.

Kada korisnik popuni ova polja sa svojim podacima i pritisne dugme "Register", komponenta aktivira funkciju handleSubmit. Ova funkcija ima ključnu ulogu u procesu registracije jer šalje Axios POST zahtev prema određenom API endpointu (u ovom slučaju, 'http://127.0.0.1:8000/api/register') kako bi se korisnički podaci poslali na server u svrhu registracije.

Funkcija handleChange se poziva svaki put kada korisnik unese ili izmeni neku od vrednosti u poljima forme. Korišcenjem handleChange, komponenta prati promene unosa i ažurira formData stanje kako bi reflektovalo trenutne vrednosti unosa.

import React, { useState } from 'react';

import './Register.css';

import InputField from './InputField';

import axios from 'axios';

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

const Register = () => {

  const [formData, setFormData] = useState({

    name: 'ana',

    email: 'anaa@gmail.com',

    password: 'anaana1234',

    password\_confirmation: 'anaana1234',

    date\_of\_birth: '2000-01-01',

    bio: 'ana bio'

  });

  const navigate = useNavigate();

  const handleChange = (e) => {

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

  };

  const handleSubmit = async (e) => {

    e.preventDefault();

    try {

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

      console.log(response.data);

      navigate('/login');

    } catch (error) {

      console.error(error);

    }

  };

  return (

    <div className="register-container">

      <div className="register-card">

        <h2>Register</h2>

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

          <InputField

            label="Name"

            type="text"

            id="name"

            name="name"

            value={formData.name}

            onChange={handleChange}

            required

          />

          <InputField

            label="Email"

            type="email"

            id="email"

            name="email"

            value={formData.email}

            onChange={handleChange}

            required

          />

          <InputField

            label="Password"

            type="password"

            id="password"

            name="password"

            value={formData.password}

            onChange={handleChange}

            required

          />

          <InputField

            label="Confirm Password"

            type="password"

            id="confirmPassword"

            name="password\_confirmation"

            value={formData.confirmPassword}

            onChange={handleChange}

            required

          />

          <InputField

            label="Date of Birth"

            type="date"

            id="dateOfBirth"

            name="date\_of\_birth"

            value={formData.dateOfBirth}

            onChange={handleChange}

            required

          />

          <InputField

            label="Bio"

            type="textarea"

            id="bio"

            name="bio"

            value={formData.bio}

            onChange={handleChange}

            required

          />

          <button type="submit" className="register-button">Register</button>

        </form>

      </div>

    </div>

  );

};

export default Register;

Komponenta Login je sluzi za prijavu korisnika u aplikaciju. Ova komponenta prikazuje formu sa dva polja za unos - jedno za unos email adrese i drugo za unos lozinke.

Kada korisnik unese svoje podatke i pritisne dugme "Login", komponenta aktivira funkciju handleSubmit. Ova funkcija šalje Axios POST zahtjev prema određenom API endpointu (u ovom slučaju, 'http://127.0.0.1:8000/api/login') kako bi korisnički podaci bili poslani na server radi provere identiteta. Ako su korisnički podaci tačni i uspešno autentifikovani na serveru, server će odgovoriti sa tokenom koji se koristi za autentifikaciju korisnika u budućim zahtevima prema zaštićenim resursima. Takođe, korisnički ID se takođe čuva u sesiji radi dalje identifikacije.

U slučaju neuspešnog prijavljivanja, greške će biti uhvaćene i prikazane kao poruke u konzoli.

import React, { useState } from 'react';

import './Login.css';

import { FaUser } from "react-icons/fa";

import InputField from './InputField';

import axios from 'axios';

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

const Login = ({setToken}) => {

  let navigate=useNavigate();

  const [formData, setFormData] = useState({

    email: 'ana@gmail.com',

    password: 'anaana1234',

  });

  const handleChange = (e) => {

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

  };

  const handleSubmit = async (e) => {

    e.preventDefault();

    try {

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

      const { token } = response.data;

      console.log(response.data);

      setToken(token)

      sessionStorage.setItem('token', token);

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

      console.log('Login successful');

      navigate('/docs');

    } catch (error) {

      console.error(error);

    }

  };

  return (

    <div className="login-container">

      <div className="login-card">

        <div className="login-icon">

          <FaUser />

        </div>

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

          <InputField

            label="Email Address"

            type="email"

            id="email"

            name="email"

            required

            value={formData.email}

            onChange={handleChange}

          />

          <InputField

            label="Password"

            type="password"

            id="password"

            name="password"

            required

            value={formData.password}

            onChange={handleChange}

          />

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

        </form>

        <div className="login-footer">

          <a href="#signup">Signup</a>

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

        </div>

      </div>

    </div>

  );

};

export default Login;

Ove tri komponente (ispod) predstavljaju korisnički definisane kuke (custom hooks) u React aplikaciji. Kuke su funkcije koje olakšavaju ponovno korišćenje i organizaciju koda za zajedničke funkcionalnosti. Na primer useCategories omogućava aplikaciji da dohvati i koristi informacije o kategorijama iz API-ja.

Koristi useState kako bi čuvala listu kategorija i useEffect kako bi izvršila HTTP GET zahtev ka određenom API endpointu za kategorije.

Kada se podaci uspešno dohvate, setuje ih u stanje categories koje se može koristiti u komponentama koje koriste ovu kuku.

import { useState, useEffect } from 'react';

import axios from 'axios';

const useTags = () => {

    const [tags, setTags] = useState([]);

    useEffect(() => {

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

            .then(response => {

                setTags(response.data);

            })

            .catch(error => {

                console.error('Error fetching tags:', error);

            });

    }, []);

    return {tags,setTags};

};

export default useTags;

import { useState, useEffect } from 'react';

import axios from 'axios';

const useDocuments = () => {

    const [documents, setDocuments] = useState([]);

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

    useEffect(() => {

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

            .then(response => {

                setDocuments(response.data);

            })

            .catch(error => {

                console.error('There was an error fetching the documents:', error);

                setError(error);

            });

    }, []);

    return { documents,setDocuments, error };

};

export default useDocuments;

import { useState, useEffect } from 'react';

import axios from 'axios';

const useCategories = () => {

    const [categories, setCategories] = useState([]);

    useEffect(() => {

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

            .then(response => {

                setCategories(response.data);

            })

            .catch(error => {

                console.error('Error fetching categories:', error);

            });

    }, []);

    return {categories,setCategories};

};

export default useCategories;

DocumentsTable komponenta ima više funkcionalnosti, uključujući prikaz, pretragu, uređivanje, preuzimanje i brisanje dokumenata. Koristi se za upravljanje dokumentima unutar aplikacije.

Komponenta prvo koristi tri custom kuke (useDocuments, useCategories i useTags) kako bi dohvatila podatke o dokumentima, kategorijama i oznakama sa servera.

U tabeli se prikazuju podaci o dokumentima, kao što su ID, naslov, sadržaj, autor, kategorija, oznake, status i akcije (preuzimanje, brisanje i uređivanje). Tabela je paginirana, što znači da se prikazuje određeni broj dokumenata po stranici, a korisnici mogu da se kreću kroz stranice koristeći paginaciju.

Korisnicima je omogućeno filtriranje dokumenata po kategoriji i oznakama, što olakšava pronalaženje određenih dokumenata.

Takođe, komponenta omogućava uređivanje dokumenata putem modala. Kada se izabere dokument za uređivanje, otvara se modal sa formom za izmenu podataka o dokumentu. Nakon što se izmene podaci i sačuvaju, dokument se ažurira na serveru i prikazuje se ažuriran prikaz u tabeli.

Osim uređivanja, komponenta omogućava i preuzimanje dokumenata. Kada korisnik klikne na dugme za preuzimanje, dokument se preuzima sa servera i spašava na računar korisnika.

Takođe, komponenta omogućava i brisanje dokumenata. Kada korisnik klikne na dugme za brisanje, dokument se briše sa servera i uklanja iz tabele.

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

import axios from 'axios';

import './DocumentsTable.css';

import DocumentTableRow from './DocumentTableRow';

import useDocuments from '../customHooks/useDocuments';

import useCategories from '../customHooks/useCategories';

import useTags from '../customHooks/useTags';

import DocumentEditModal from './update/DocumentEditModal';

const DocumentsTable = () => {

    const { documents,setDocuments, error } = useDocuments();

    const [isEditModalOpen, setIsEditModalOpen] = useState(false);

    const [currentDocument, setCurrentDocument] = useState(null);

    const [filterByTags, setFilterByTags] = useState([]);

    const [filterByCategory, setFilterByCategory] = useState('');

    const [filteredDocuments, setFilteredDocuments] = useState([]);

    const {categories,setCategories} = useCategories();

    const {tags,setTags} = useTags();

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

    const documentsPerPage = 3;

    const handleEdit = (document) => {

        setCurrentDocument(document);

        setIsEditModalOpen(true);

    };

    const handleSaveEdit = (documentId, editFormData) => {

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

        const config = {

            headers: {

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

                'Content-Type': 'application/json' // Ako šaljete JSON

            }

        };

        const updatedData = {

            ...editFormData,

            tags: editFormData.tags.map(Number)

        };

        console.log(updatedData.tags)

        axios.put(`http://127.0.0.1:8000/api/documents/${documentId}`, updatedData, config)

            .then(response => {

                // Ažuriranje prikaza dokumenata

                setDocuments(documents.map(doc => doc.id === documentId ? {...doc, ...response.data} : doc));

                alert('Document updated successfully');

                setIsEditModalOpen(false); // Zatvaranje modala

            })

            .catch(error => {

                console.error('Error updating document:', error);

            });

    };

    // const [documents, setDocuments] = useState([]);

    // useEffect(() => {

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

    //         .then(response => {

    //             setDocuments(response.data);

    //         })

    //         .catch(error => {

    //             console.error('There was an error fetching the documents:', error);

    //         });

    // }, []);

    const handleDownload = (documentId, fileName) => {

        axios({

            url: `http://127.0.0.1:8000/api/documents/${documentId}/download`,

            method: 'GET',

            responseType: 'blob', // Važno za preuzimanje fajlova  \*/////Opcija responseType: 'blob' u Axios zahtevu označava da želite da odgovor koji primate od servera bude predstavljen kao "Blob" objekt u JavaScriptu. "Blob" (Binary Large Object) je tip objekta koji se koristi za predstavljanje binarnih podataka, kao što su slike, audio datoteke, ili u ovom slučaju, datoteke koje preuzimate sa servera.

        })

        .then((response) => {

            const blob = new Blob([response.data], { type: response.headers['content-type'] });  //Ova linija koda stvara objekt Blob (Binary Large Object) koji predstavlja binarni podatak. response.data pretpostavlja se da sadrži binarni podatak koji je došao kao odgovor na HTTP zahtjev. response.headers['content-type'] koristi se za postavljanje tipa sadržaja (MIME tip) za taj objekt Blob, na osnovu informacija iz zaglavlja odgovora.

            const url = window.URL.createObjectURL(blob);  //Ova linija koda koristi window.URL.createObjectURL metodu kako bi se stvorila URL adresa koja se odnosi na Blob objekt stvoren u prethodnoj liniji. Ova URL adresa će biti privremeno dostupna i može se koristiti za preuzimanje sadržaja.

            const link = document.createElement('a'); //Ova linija koda stvara novi HTML element <a> (hipertekstualna veza) koji će se koristiti za kreiranje linka za preuzimanje datoteke.

            link.href = url;  //Ova linija koda postavlja href atribut HTML elementa <a> na prethodno stvorenu URL adresu, čime se veza povezuje s Blob objektom i omogućava preuzimanje sadržaja.

            link.setAttribute('download', fileName);  //Ova linija koda postavlja atribut download na HTML elementu <a> i dodjeljuje mu vrijednost fileName. To omogućava korisnicima da preuzmu sadržaj sa zadanim imenom datoteke kada kliknu na vezu.

            document.body.appendChild(link); //Ova linija koda dodaje HTML element <a> u telo (body) HTML dokumenta, čime se omogućava njegovo prikazivanje na stranici.

            link.click(); //pokrece preuzimanje fajla

            link.parentNode.removeChild(link); //Na kraju, ova linija koda uklanja HTML element <a> iz dokumenta kako bi se oslobodili resursi i očistila stranica od tog elementa nakon preuzimanja.

        })

        .catch((error) => {

            console.error('There was an error downloading the file:', error);

        });

    };

    const handleDelete = (documentId) => {

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

        const config = {

            headers: {

                'Authorization': `Bearer ${token}`

            }

        };

        axios.delete(`http://127.0.0.1:8000/api/documents/${documentId}`, config)

            .then(() => {

                setDocuments(documents.filter(document => document.id !== documentId));

            })

            .catch((error) => {

                console.error('Došlo je do greške prilikom brisanja dokumenta:', error);

            });

    };

    useEffect(() => {

        const filterDocuments = () => {

            let updatedFilteredDocuments = [...documents];

            if (filterByCategory) {

                updatedFilteredDocuments = updatedFilteredDocuments.filter(document => document.category\_id.toString() === filterByCategory);

            }

            if (filterByTags.length > 0) {

                updatedFilteredDocuments = updatedFilteredDocuments.filter(document =>

                    filterByTags.every(tag => document.tags.map(tag => tag.toString()).includes(tag))

                );

            }

            setFilteredDocuments(updatedFilteredDocuments);

        };

        filterDocuments();

    }, [documents, filterByTags, filterByCategory]);   // Ovde primenjujemo filtere svaki put kada se promeni stanje `documents`, `filterByTags`, ili `filterByCategory`

    if (error) {

        return <div>Došlo je do greške: {error.message}</div>;

    }

    const indexOfLastDocument = currentPage \* documentsPerPage;

    const indexOfFirstDocument = indexOfLastDocument - documentsPerPage;

    const currentDocuments = filteredDocuments.slice(indexOfFirstDocument, indexOfLastDocument);

    const paginate = (pageNumber) => setCurrentPage(pageNumber);

    const pageNumbers = [];

    for (let i = 1; i <= Math.ceil(filteredDocuments.length / documentsPerPage); i++) {

      pageNumbers.push(i);

    }

    const renderPageNumbers = pageNumbers.map(number => {

        return (

          <li key={number} className='page-item'>

            <button onClick={() => paginate(number)}   className='page-link'>

              {number}

            </button>

          </li>

        );

      });

    return (

        <div className="documents-container">

            <div className="filter-container">

                <div className="input-group">

                    <label htmlFor="filter-category">Filter by Category</label>

                    <select id="filter-category" value={filterByCategory} onChange={(e) => setFilterByCategory(e.target.value)}>

                    <option value="">All Categories</option>

                    {categories.map((category) => (

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

                        {category.name}

                        </option>

                    ))}

                    </select>

                </div>

                <div className="input-group">

                    <label htmlFor="filter-tags">Filter by Tags</label>

                    <select id="filter-tags" multiple value={filterByTags} onChange={(e) => {

                    const options = e.target.options;

                    const value = [];

                    for (let i = 0, l = options.length; i < l; i++) {

                        if (options[i].selected) {

                        value.push(options[i].value);

                        }

                    }

                    setFilterByTags(value);

                    }}>

                    {tags.map((tag) => (

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

                        {tag.name}

                        </option>

                    ))}

                    </select>

                </div>

                </div>

            <table className="documents-table">

                <thead>

                    <tr>

                        <th>ID</th>

                        <th>Title</th>

                        <th>Content</th>

                        <th>Author ID</th>

                        <th>Category ID</th>

                        <th>Tags</th>

                        <th>Public</th>

                        <th>Download</th>

                        <th>Delete</th>

                        <th>Edit</th>

                    </tr>

                </thead>

                <tbody>

                    {currentDocuments.map((document) => (

                        <DocumentTableRow

                        key={document.id}

                        document={document}

                        onDelete={handleDelete}

                        onDownload={handleDownload}

                        onEdit={handleEdit}

                        />

                    ))}

                    </tbody>

            </table>

            <div>

            <ul className='pagination'>

                {renderPageNumbers}

            </ul>

            </div>

            {isEditModalOpen && currentDocument && (

                <DocumentEditModal

                    document={currentDocument}

                    categories={categories}

                    tags={tags}

                    onSave={handleSaveEdit}

                    onClose={() => setIsEditModalOpen(false)}

                />

            )}

        </div>

    );

};

export default DocumentsTable;

# Korisničko uputstvo

Pocetna stranica

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

Opis je automatski generisan

Stranice za login i registraciju

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

Opis je automatski generisan

Slika na kojoj se nalazi tekst, snimak ekrana, dizajn

Opis je automatski generisan

Stranica za prikaz tabele sa dokumentima

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

Opis je automatski generisan

Stranica za unos novog dokumenta

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

Opis je automatski generisan

Forma za editovanje dokumenata

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

Opis je automatski generisan

Stranica za prikaz dokumenata sa spoljnog apija

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

Opis je automatski generisan