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 koja predstavlja platformu za komunikaciju i interakciju među korisnicima. Glavne funkcionalnosti aplikacije uključuju mogućnost registracije i prijave korisnika. Nakon prijave, korisnici mogu da uređuju svoj korisnički profil, uključujući promenu lozinke i pregledanje informacija o sebi.

Jedna od ključnih funkcionalnosti je mogućnost pristupanja i pregledanja različitih chat soba. Korisnici mogu pretraživati dostupne chat sobe, pridruživati se njima i razmenjivati poruke u realnom vremenu unutar svake sobe. Takođe, aplikacija omogućava pretragu poruka po ključnim rečima.

Aplikacija takođe ima dodatnu funkcionalnost prikaza "Fun Fact"-a, koji se periodično osvežava i pruža korisnicima zanimljive činjenice.

# Implementacija

Komponenta AuthForm se koristi za implementaciju funkcionalnosti prijave i registracije korisnika u aplikaciji. Komponenta se koristi za interakciju sa korisničkim podacima kao što su ime, email, šifra, slika profila, datum rođenja, biografija i lokacija.

Komponenta koristi React-ov Hook "useState" za praćenje trenutnog stanja prijave ili registracije (označeno varijablom "isLogin") i podataka unetih u formi (spremljenih u "formData"). Takođe se koristi "useNavigate" iz "react-router-dom" za preusmeravanje korisnika na određenu stranicu nakon uspešne prijave ili registracije.

Kod takođe uključuje funkcije za obradu promena unosa (inputa) i selektovanja slike profila. Za prijavu, šalje se HTTP zahtev prema serveru sa unetim emailom i šifrom. Za registraciju, koristi se "FormData" objekat koji omogućava slanje različitih tipova podataka, uključujući i sliku profila.

import React, { useState } from 'react';

import axios from 'axios';

import './AuthForm.css';

import InputField from './InputField';

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

const AuthForm = ({setToken}) => {

  let navigate= useNavigate();

  const [isLogin, setIsLogin] = useState(true);

  const [formData, setFormData] = useState({

    name: '',

    email: '',

    password: '',

    confirmPassword: '',

    date\_of\_birth: '',

    bio: '',

    location: '',

  });

  const handleInputChange = (e) => {

    const { name, value } = e.target;

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

  };

  const handleFileChange = (e) => {

    setFormData({ ...formData, profile\_image: e.target.files[0] });

  };

  const handleSubmit = async (e) => {

    e.preventDefault();

    if (isLogin) {

      try {

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

          email: formData.email,

          password: formData.password,

        });

        console.log(response.data);

        localStorage.setItem("auth\_token",response.data.token);

        localStorage.setItem("auth\_id",response.data.user.id);

        setToken(response.data.token)

        navigate('/chatrooms')

      } catch (error) {

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

        alert(error.response.data.message.toString());

      }

    } else {

      try {

        const registerFormData = new FormData();

        Object.keys(formData).forEach((key) => {

          if (key !== 'confirmPassword') {

            registerFormData.append(key, formData[key]);

          }

        });

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

          headers: {

            'Content-Type': 'multipart/form-data',

          },

        });

        console.log(response.data);

      } catch (error) {

        console.error('Registration error:', error.response.data);

        alert(error.response.data.message.toString());

      }

    }

  };

  return (

    <div className="auth-form">

      <h2>{isLogin ? 'Login' : 'Register'}</h2>

      <form onSubmit={handleSubmit}>

        {!isLogin && (

          <>

            <InputField

              type="text"

              name="name"

              placeholder="Name"

              value={formData.name}

              onChange={handleInputChange}

            />

            <InputField

              type="date"

              name="date\_of\_birth"

              placeholder="Date of Birth"

              value={formData.date\_of\_birth}

              onChange={handleInputChange}

            />

            <InputField

              type="text"

              name="bio"

              placeholder="Bio"

              value={formData.bio}

              onChange={handleInputChange}

            />

            <InputField

              type="text"

              name="location"

              placeholder="Location"

              value={formData.location}

              onChange={handleInputChange}

            />

            <InputField

              type="file"

              name="profile\_image"

              onChange={handleFileChange}

            />

          </>

        )}

        <InputField

          type="email"

          name="email"

          placeholder="Email"

          value={formData.email}

          onChange={handleInputChange}

        />

        <InputField

          type="password"

          name="password"

          placeholder="Password"

          value={formData.password}

          onChange={handleInputChange}

        />

        {!isLogin && (

          <InputField

            type="password"

            name="confirmPassword"

            placeholder="Confirm Password"

            value={formData.confirmPassword}

            onChange={handleInputChange}

          />

        )}

        <button type="submit">{isLogin ? 'Login' : 'Create Account'}</button>

      </form>

      <button onClick={() => setIsLogin(!isLogin)}>

        {isLogin ? "Don't have an account? Register" : 'Have an account? Login'}

      </button>

    </div>

  );

};

export default AuthForm;

UserProfile" se koristi za prikaz i uređivanje korisničkog profila. Korisnicima se omogućava pregled informacija o profilu, poput imena, email adrese, datuma rođenja, lokacije i biografije, kao i pregled chat soba koje korisnik poseduje. Takođe, korisnicima je omogućeno uređivanje informacija o profilu i promena lozinke.

Komponenta koristi različite React Hooks kao što su "useState" za praćenje stanja, "useEffect" za asinhrono dohvaćanje podataka o chat sobama i "useNavigate" za upravljanje rutama. Takođe, koristi se "axios" za komunikaciju sa serverom i "react-icons/fa" za prikaz ikona.

Kada korisnik klikne na dugme "Uredi Profil", komponenta omogućava uređivanje informacija o profilu, uključujući ime, email, datum rođenja, biografiju i lokaciju. Nakon uređivanja, korisnik može sačuvati promene klikom na dugme "Ažuriraj Profil".

Takođe, komponenta omogućava promenu lozinke kroz formu "Promena Lozinke", gde korisnik unosi trenutnu lozinku, novu lozinku i potvrdu nove lozinke.

Ukoliko korisnik poseduje chat sobe, komponenta prikazuje listu tih soba i omogućava korisniku da otvori određenu sobu klikom na dugme "Otvori".

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

import axios from 'axios';

import './UserProfile.css';

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

import { FaBirthdayCake, FaEnvelope, FaMapMarkerAlt, FaUser, FaUserEdit } from 'react-icons/fa';

import useUserData from '../korDefKuke/useUserData';

const UserProfile = () => {

  const auth\_token = localStorage.getItem('auth\_token');

  const { userData, loading, error, setUserData } = useUserData(auth\_token);

  const [newPasswordData, setNewPasswordData] = useState({

    current\_password: '',

    new\_password: '',

    new\_password\_confirmation: '',

  });

  const [chatRooms, setChatRooms] = useState([]);

  const navigate = useNavigate();

  const [isEditing, setIsEditing] = useState(false);

  const handleUserInputChange = (e) => {

    const { name, value } = e.target;

    setUserData({ ...userData, [name]: value });

  };

  const handlePasswordChange = (e) => {

    const { name, value } = e.target;

    setNewPasswordData({ ...newPasswordData, [name]: value });

  };

  const handleProfileUpdate = async (e) => {

    e.preventDefault();

    try {

      const response = await axios.put('http://127.0.0.1:8000/api/user/update', userData, {

        headers: {

          Authorization: `Bearer ${localStorage.getItem('auth\_token')}`,

        },

      });

      console.log('Profile updated:', response.data);

      setIsEditing(false);

    } catch (error) {

      console.error('Profile update error:', error.response.data);

    }

  };

  const handleChangePassword = async (e) => {

    e.preventDefault();

    try {

      const response = await axios.post('http://127.0.0.1:8000/api/change-password', newPasswordData, {

        headers: {

          Authorization: `Bearer ${localStorage.getItem('auth\_token')}`,

        },

      });

      console.log('Password changed:', response.data);

      setNewPasswordData({

        current\_password: '',

        new\_password: '',

        new\_password\_confirmation: '',

      });

    } catch (error) {

      console.error('Change password error:', error.response.data);

    }

  };

  useEffect(() => {

    const fetchChatRooms = async () => {

      try {

        const response = await axios.get('http://127.0.0.1:8000/api/my-chat-rooms', {

          headers: {

            Authorization: `Bearer ${auth\_token}`,

          },

        });

        setChatRooms(response.data);

      } catch (error) {

        console.error('Error fetching chat rooms:', error.response.data);

      }

    };

    fetchChatRooms();

  }, [auth\_token]);

  if (loading) {

    return <div>Loading...</div>;

  }

  if (error) {

    return <div>Error: {error}</div>;

  }

  const handleOpenRoom = (roomId) => {

    navigate(`/chatrooms/${roomId}`);

  };

return (

  <div className="user-profile-container">

  <div className="user-info-display">

      {/\* <img

      src={userData.profile\_image ? `http://127.0.0.1:8000/storage/${userData.profile\_image}` : <FaUser />}

      alt="Profile"

      className="profile-image"

    /> \*/}

    <div  className="profile-image">

      {userData.profile\_image ? null : <FaUser />}

    </div>

        <h2><FaUser /> {userData.name}</h2>

        <p><FaEnvelope /> {userData.email}</p>

        <p><FaBirthdayCake /> {userData.date\_of\_birth}</p>

        <p><FaMapMarkerAlt /> {userData.location}</p>

        <p>{userData.bio}</p>

        <button onClick={() => setIsEditing(true)} className="edit-button">

          <FaUserEdit /> Uredi Profil

        </button>

      </div>

      <div className="user-chat-rooms">

        <h2>Moje Chat Sobe</h2>

        <ul>

          {chatRooms.map((room) => (

            <li key={room.id}>

              {room.name}

              <button onClick={() => handleOpenRoom(room.id)}>Otvori</button>

            </li>

          ))}

        </ul>

      </div>

    {isEditing ? ( // Prikazivanje forme za uređivanje samo ako je isEditing true

      <>

        <h2>Profil</h2>

        <form onSubmit={handleProfileUpdate} className="profile-form">

          <input

            type="text"

            name="name"

            placeholder="Name"

            value={userData.name}

            onChange={handleUserInputChange}

          />

          <input

            type="email"

            name="email"

            placeholder="Email"

            value={userData.email}

            onChange={handleUserInputChange}

          />

          <input

            type="date"

            name="date\_of\_birth"

            placeholder="Date of Birth"

            value={userData.date\_of\_birth}

            onChange={handleUserInputChange}

          />

          <input

            type="text"

            name="bio"

            placeholder="Bio"

            value={userData.bio}

            onChange={handleUserInputChange}

          />

          <input

            type="text"

            name="location"

            placeholder="Location"

            value={userData.location}

            onChange={handleUserInputChange}

          />

          <button type="submit">Ažuriraj Profil</button>

          <button onClick={() => setIsEditing(false)} className="close-button">

            Zatvori

          </button>

        </form>

      </>

    ) : null}

      {/\* <h2>Promena Lozinke</h2>

      <form onSubmit={handleChangePassword}>

        <input

          type="password"

          name="current\_password"

          placeholder="Trenutna Lozinka"

          value={newPasswordData.current\_password}

          onChange={handlePasswordChange}

        />

        <input

          type="password"

          name="new\_password"

          placeholder="Nova Lozinka"

          value={newPasswordData.new\_password}

          onChange={handlePasswordChange}

        />

        <input

          type="password"

          name="new\_password\_confirmation"

          placeholder="Potvrda Nove Lozinke"

          value={newPassword}

            onChange={handlePasswordChange}

            />

            <button type="submit">Promeni Lozinku</button>

          </form>

         \*/}

         </div>

      );

    };

    export default UserProfile;

"ChatRooms" se koristi za prikazivanje dostupnih chat soba i omogućava korisnicima da pretražuju sobe, pridruže se njima i pregledaju ih. Komponenta koristi različite React Hook-ove kao što su "useState" za praćenje stanja, "useEffect" za asinhrono dohvaćanje podataka o sobama, i "useCallback" za optimizaciju performansi prilikom poziva funkcije za dohvaćanje soba.

Korisnicima je omogućeno da unesu pojam za pretragu chat soba, pri čemu se pri svakoj promeni termina automatski ažurira lista prikazanih soba. Takođe, komponenta podržava paginaciju, gde se prikazuju samo određeni broj soba po stranici, a korisnik može da prelazi na sledeće i prethodne stranice.

Sobe se prikazuju u tabeli sa informacijama kao što su ime, da li su privatne, maksimalni broj učesnika i opis. Korisnik može da se pridruži sobi klikom na dugme "Pridruži se".

Komponenta takođe obrađuje greške koje mogu da se pojave prilikom komunikacije sa serverom i prikazuje odgovarajuće poruke o greškama ili poruku o učitavanju dok se podaci dohvataju.

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

import axios from 'axios';

import './ChatRooms.css';

import ReactPaginate from 'react-paginate';

const ChatRooms = () => {

  const authToken = localStorage.getItem('auth\_token');

  const userId = localStorage.getItem('auth\_id');

  const [chatRooms, setChatRooms] = useState([]);

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

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

  const [searchTerm, setSearchTerm] = useState('');

  const [currentPage, setCurrentPage] = useState(0);

  const perPage = 5;

  const handlePageClick = (event) => {

    setCurrentPage(event.selected);

  };

  const handleSearchChange = (e) => {

    setSearchTerm(e.target.value);

    setCurrentPage(0);

  };

    const filteredChatRooms = chatRooms.filter(

      room =>

        room.name.toLowerCase().includes(searchTerm.toLowerCase()) ||

        room.description.toLowerCase().includes(searchTerm.toLowerCase())

    );

  const fetchChatRooms = useCallback(async () => {

    setLoading(true);

    try {

      const response = await axios.get('http://127.0.0.1:8000/api/chat-rooms', {

        headers: {

          Authorization: `Bearer ${authToken}`,

        },

      });

      setChatRooms(response.data);

    } catch (err) {

      setError(err);

    } finally {

      setLoading(false);

    }

  }, [authToken]);

  useEffect(() => {

    fetchChatRooms();

  }, [fetchChatRooms]);

  const handleJoinRoom = async (chatRoomId) => {

    try {

      await axios.post(

        'http://127.0.0.1:8000/api/chat-room-users',

        {

          user\_id: userId,

          chat\_room\_id: chatRoomId,

          joined\_at: new Date().toISOString(),

          role: 'member',

        },

        {

          headers: {

            Authorization: `Bearer ${authToken}`,

          },

        }

      );

      fetchChatRooms();

    } catch (error) {

      console.error('Error joining room:', error.response.data);

    }

  };

  if (loading) return <div>Loading...</div>;

  if (error) return <div>Error: {error.message}</div>;

  const displayedChatRooms = filteredChatRooms.slice(currentPage \* perPage, (currentPage + 1) \* perPage);

  return (

    <div className="chat-rooms-container">

      <h1>Available chat rooms</h1>

      <input

        type="text"

        placeholder="Pretraga chat soba"

        value={searchTerm}

        onChange={handleSearchChange}

        className="search-input"

      />

      <table className="chat-rooms-table">

        <thead>

          <tr>

            <th>Name</th>

            <th>Private</th>

            <th>Max Participants</th>

            <th>Description</th>

            <th>Action</th>

          </tr>

        </thead>

        <tbody>

         {displayedChatRooms.map((room) => (

            <tr key={room.id}>

              <td>{room.name}</td>

              <td>{room.is\_private ? 'Yes' : 'No'}</td>

              <td>{room.max\_participants}</td>

              <td>{room.description || 'N/A'}</td>

              <td>

                <button onClick={() => handleJoinRoom(room.id)} className="join-button">

                  Pridruži se

                </button>

              </td>

            </tr>

          ))}

        </tbody>

      </table>

      <ReactPaginate

        previousLabel={'Prethodna'}

        nextLabel={'Sledeća'}

        breakLabel={'...'}

        pageCount={Math.ceil(chatRooms.length / perPage)}

        marginPagesDisplayed={2}

        pageRangeDisplayed={5}

        onPageChange={handlePageClick}

        containerClassName={'pagination'}

        activeClassName={'active'}

      />

    </div>

  );

};

export default ChatRooms;

ChatRoom" se koristi za prikazivanje i upravljanje chat sobom u aplikaciji. Korisnicima se omogućava slanje poruka u chat sobi, pretraga poruka po ključnim rečima i prikazivanje poslednjih pet poruka. Takođe, komponenta prikazuje "Fun Fact" koji se periodično osvežava i menja.

Komponenta koristi različite React Hook-ove kao što su "useState" za praćenje stanja, "useEffect" za asinhrono dohvatanje poruka iz chat sobe i periodično osvežavanje, i "useParams" za dobijanje identifikatora chat sobe iz URL adrese.

Kada korisnik unese poruku i klikne na dugme "Pošalji", poruka se šalje na server putem HTTP zahteva, a zatim se dodaje u listu prikazanih poruka. Takođe, korisnici mogu da pretražuju poruke po ključnim rečima pomoću polja za pretragu.

"Fun Fact" se dohvata sa spoljnog API-ja i prikazuje se u okviru komponente.

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

import axios from 'axios';

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

import './ChatRoom.css';

import Message from './Message';

const ChatRoom = () => {

  const [message, setMessage] = useState('');

  const [messages, setMessages] = useState([]);

  const [searchTerm, setSearchTerm] = useState('');

  const [funFact, setFunFact] = useState(''); // Dodajemo state za fun fact

  const userId = localStorage.getItem('auth\_id');

  const authToken = localStorage.getItem('auth\_token');

  const { id: chatRoomId } = useParams();

  const fetchChatRoomMessages = async () => {

    try {

      const response = await axios.get(

        `http://127.0.0.1:8000/api/chat-rooms/messages/${chatRoomId}`,

        {

          headers: {

            Authorization: `Bearer ${authToken}`,

          },

        }

      );

      setMessages(response.data.data);

    } catch (error) {

      console.error('Greška pri dobijanju poruka:', error);

    }

  };

  const fetchFunFact = async () => {

    const apiKey = '7xiJG3ZG/DVXBFQcpnUANw==DCKsOuWEdluVhptV';

    const apiUrl = 'https://api.api-ninjas.com/v1/facts?limit=1';

    try {

      const response = await axios.get(apiUrl, {

        headers: {

          'X-Api-Key': apiKey

        }

      });

      if (response.data) {

        setFunFact(response.data[0].fact);

      }

    } catch (error) {

      console.error('Error fetching Fun Fact:', error);

    }

  };

  useEffect(() => {

    fetchChatRoomMessages();

    // Postavljanje intervala za periodično osvežavanje svakih 5 sekundi

    const intervalId = setInterval(fetchChatRoomMessages, 5000);

    // Očisti interval pri povlačenju komponente

    return () => clearInterval(intervalId);

  }, [chatRoomId]);

  useEffect(() => {

    fetchFunFact(); // Pozivamo funkciju za dohvatanje fun fact-a

  }, []); // Prazan niz znači da će se pozvati samo jednom prilikom montiranja komponente

  const handleSubmit = async (e) => {

    e.preventDefault();

    try {

      const response = await axios.post(

        'http://127.0.0.1:8000/api/messages',

        {

          body: message,

          user\_id: userId,

          chat\_room\_id: chatRoomId,

        },

        {

          headers: {

            Authorization: `Bearer ${authToken}`,

          },

        }

      );

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

      setMessages([...messages, response.data]);

      setMessage('');

    } catch (error) {

      console.error('Greška pri kreiranju poruke:', error);

    }

  };

  const handleSearchChange = (e) => {

    setSearchTerm(e.target.value);

  };

  const filteredMessages = messages.filter((msg) => {

    const userMatch = msg.user.name.toLowerCase().includes(searchTerm.toLowerCase());

    const bodyMatch = msg.body.toLowerCase().includes(searchTerm.toLowerCase());

    return userMatch || bodyMatch;

  });

  return (

    <div className="chat-room-container">

      <h1>Chat Room</h1>

      <div className="fun-fact">

        <p>{funFact}</p>

      </div>

      <div className="search-input-container">

        <label>Pretraga:</label>

        <input

          type="text"

          value={searchTerm}

          onChange={handleSearchChange}

          className="search-input"

        />

      </div>

      <table className="chat-room-table">

        <thead>

          <tr>

            <th>ID</th>

            <th>Poruka</th>

            <th>Korisnik</th>

          </tr>

        </thead>

        <tbody>

          {filteredMessages.slice(-5).map((msg) => (

            <Message

              key={msg.id}

              id={msg.id}

              body={msg.body}

              userName={msg.user.name}

            />

          ))}

        </tbody>

      </table>

      <br />

      <form className="chat-room-form" onSubmit={handleSubmit}>

        <div>

          <label>Poruka:</label>

          <input

            type="text"

            value={message}

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

            className="chat-room-input"

          />

        </div>

        <button type="submit" className="chat-room-submit-button">

          Pošalji

        </button>

      </form>

    </div>

  );

};

export default ChatRoom;

# Korisničko uputstvo

Pocetna stranica

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

Opis je automatski generisan

Stranica na kojoj korisnici mogu da se uloguju ili registruju

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

Opis je automatski generisan

Slika na kojoj se nalazi tekst, snimak ekrana, softver, Ikona na računaru

Opis je automatski generisan

Stranica na kojoj ulogovani korisnici mogu da vide podatke o svom profilu

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

Opis je automatski generisan

Klikom na dugme Uredi profil korisniku se otvara forma za azuriranje podataka

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

Opis je automatski generisan

Klikom na dugme otvori u Moje Chat Sobe sekciji korisniku se otvara komponenta preko koje mu je omoguceno da cetuje sa drugim korsinicima

Slika na kojoj se nalazi tekst, snimak ekrana, softver, Ikona na računaru

Opis je automatski generisan

Stranica na kojoj korisnici mogu da se pridruze chat sobama

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

Opis je automatski generisan