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 omogućava koriscnima da se lako prijave ili registruju, gde mogu da unesu svoje podatke poput imena, prezimena, email adrese i lozinke, a sve to sa ciljem kreiranja personalizovanog korisničkog iskustva. Jednom kada su prijavljeni, korisnici imaju pristup detaljnim informacijama o firmama, uključujući njihove osnovne podatke kao što su naziv, adresa, vlasnik i PIB. Što se tiče finansijskog segmenta, aplikacija pruža uvid u aktuelne cene akcija raznih kompanija, pri čemu korisnici mogu da pretražuju i sortiraju informacije prema različitim kriterijumima, uključujući i naziv firme ili simbol akcija.

# Implementacija

Komponenta LoginComponent omogućava korisnicima da unesu svoje email adrese i lozinke kroz dva polja formulara. Kad korisnik popuni ova polja i potvrdi unos pritiskom na dugme za prijavu, informacije se šalju na server koristeći axios, koji je zadužen za komunikaciju s backendom. U slučaju uspešne prijave, aplikacija čuva dobijeni token u sesijskoj memoriji pregledača i preusmerava korisnika na stranicu sa firmama, signalizirajući da je prijava uspešna. Ukoliko dođe do greške tokom procesa prijave, korisnik će biti obavešten o neuspehu. Pored mogućnosti prijave, postoji i link koji korisnike usmerava na stranicu za registraciju, ukoliko još uvek nemaju kreiran nalog.

import React, { useState } from 'react';

import axios from 'axios';

import './LoginComponent.css';

import InputField from './InputField';

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

const LoginComponent = ({setToken}) => {

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

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

  let navigate = useNavigate();

  const handleSubmit = async (event) => {

    event.preventDefault();

    try {

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

        email,

        password

      });

      sessionStorage.setItem('token', response.data.token);

      setToken(response.data.token);

      navigate('/firme');

      console.log('Login successful:', response.data);

    } catch (error) {

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

    }

  };

  return (

    <div className="login-container">

      <div className="login-form">

        <h2>Member Login</h2>

        <form onSubmit={handleSubmit}>

          <InputField

            type="email"

            name="email"

            placeholder="Email"

            value={email}

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

          />

          <InputField

            type="password"

            name="password"

            placeholder="Password"

            value={password}

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

          />

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

        </form>

        <div className="login-footer">

          {/\* <a href="#forgot">Forgot Username / Password?</a> \*/}

          <a href="/registration">Create your Account →</a>

        </div>

      </div>

    </div>

  );

};

export default LoginComponent;

Komponenta RegisterComponent namenjena je za registraciju novih korisnika, gde se od njih traži da unesu svoje osnovne informacije kao što su ime, prezime, email adresa i lozinka. Sve ovo se odvija unutar forme koja je stilizovana da izgleda privlačno i intuitivno za korisnika. Kada se forma popuni i potvrdi, podaci se šalju na server pomoću axios-a, koji služi za asinhronu komunikaciju sa backend-om. U slučaju uspešne registracije, korisnik dobija povratnu informaciju o tome i automatski se preusmerava na stranicu za prijavu, gde može da koristi svoje novo kreirane akreditive za pristup sistemu. Ukoliko proces registracije ne prođe kako treba, korisnik će biti obavešten o grešci. Takođe, na dnu forme nalazi se link koji vodi na stranicu za prijavu, namenjen korisnicima koji već imaju nalog i žele da se prijave umesto da se registruju.

import React, { useState } from 'react';

import axios from 'axios';

import './LoginComponent.css';

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

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

  <div className="input-group">

    <input

      type={type}

      name={name}

      id={name}

      placeholder={placeholder}

      value={value}

      onChange={onChange}

      required

    />

  </div>

);

const RegistrationComponent = () => {

  const [ime, setIme] = useState('');

  const [prezime, setPrezime] = useState('');

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

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

  let navigate = useNavigate();

  const handleSubmit = async (event) => {

    event.preventDefault();

    try {

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

        ime,

        prezime,

        email,

        password,

        uloga: 'korisnik'

      });

      console.log('Registration successful:', response.data);

      navigate('/login');

    } catch (error) {

      console.error('Registration failed:', error);

    }

  };

  return (

    <div className="login-container">

      <div className="login-form">

        <h2>Member Registration</h2>

        <form onSubmit={handleSubmit}>

          <InputField

            type="text"

            name="ime"

            placeholder="Ime"

            value={ime}

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

          />

          <InputField

            type="text"

            name="prezime"

            placeholder="Prezime"

            value={prezime}

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

          />

          <InputField

            type="email"

            name="email"

            placeholder="Email"

            value={email}

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

          />

          <InputField

            type="password"

            name="password"

            placeholder="Password"

            value={password}

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

          />

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

        </form>

        <div className="login-footer">

          <a href="/login">Already have an account? Log in →</a>

        </div>

      </div>

    </div>

  );

};

export default RegistrationComponent;

komponenta StockPrices, koja služi za prikaz cene akcija različitih kompanija, omogućavajući korisnicima da lako pretražuju i pregledaju informacije o ceni akcija poznatih firmi poput Apple-a, Google-a i Microsoft-a. Korisnici mogu uneti ime firme ili simbol akcije u polje za pretragu kako bi brzo pronašli željene informacije. Podaci se dinamički dohvaćaju sa servera korišćenjem axios-a, biblioteke namenjene za asinhronu HTTP komunikaciju. Pored funkcionalnosti pretrage, implementirana je i paginacija, što znači da se rezultati pretrage prikazuju na više stranica radi bolje preglednosti.

import React from 'react';

const StockRow = ({ ticker, name, price, exchange, updated }) => {

  return (

    <tr className="firma-row">

      <td>{ticker}</td>

      <td>{name}</td>

      <td>${price}</td>

      <td>{exchange}</td>

      <td>{new Date(updated \* 1000).toLocaleString()}</td>

    </tr>

  );

};

export default StockRow;

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

import axios from 'axios';

import StockRow from './StockRow';

import ReactPaginate from 'react-paginate';

const StockPrices = () => {

  const [stockPrices, setStockPrices] = useState([]);

  const apiKey = 'irvNlWP5UMn7uBeOhIwIsey9rehSVNoPOKj1v9Ez';

  const companies = [

    { symbol: 'AAPL', name: 'Apple Inc.' },

    { symbol: 'GOOGL', name: 'Alphabet Inc.' },

    { symbol: 'MSFT', name: 'Microsoft Corporation' },

    { symbol: 'AMZN', name: 'Amazon.com Inc.' },

    { symbol: 'FB', name: 'Meta Platforms, Inc.' },

    { symbol: 'TSLA', name: 'Tesla, Inc.' },

    { symbol: 'JPM', name: 'JPMorgan Chase & Co.' },

    { symbol: 'V', name: 'Visa Inc.' },

    { symbol: 'JNJ', name: 'Johnson & Johnson' },

    { symbol: 'NFLX', name: 'Netflix, Inc.' },

    { symbol: 'BAC', name: 'Bank of America Corporation' },

    { symbol: 'INTC', name: 'Intel Corporation' },

    { symbol: 'CSCO', name: 'Cisco Systems, Inc.' },

    { symbol: 'VZ', name: 'Verizon Communications Inc.' },

    { symbol: 'ADBE', name: 'Adobe Inc.' },

    { symbol: 'KO', name: 'The Coca-Cola Company' },

    { symbol: 'PG', name: 'Procter & Gamble Company' },

  ];

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

    const itemsPerPage =5;

    const [searchQuery, setSearchQuery] = useState('');

  useEffect(() => {

    companies.forEach(company => {

      const { symbol } = company;

      axios.get(`https://api.api-ninjas.com/v1/stockprice?ticker=${symbol}`, {

        headers: { 'X-Api-Key': apiKey }

      })

      .then(response => {

        setStockPrices(prices => [...prices, response.data]);

      })

      .catch(error => console.error(`Error fetching stock price for ${symbol}:`, error));

    });

  }, []);

  //za pretragu

  const filteredStockPrices = stockPrices.filter(stock =>

    (stock.name && stock.ticker) &&

    (

      stock.name.toLowerCase().includes(searchQuery.toLowerCase()) ||

      stock.ticker.toLowerCase().includes(searchQuery.toLowerCase())

    )

  );

  //za paginaciju

  const startIndex = currentPage \* itemsPerPage;

  const endIndex = startIndex + itemsPerPage;

  const displayedStockPrices = filteredStockPrices.slice(startIndex, endIndex);

  return (

    <div className="firme-container">

            <input

            type="text"

            placeholder="Search by ticker or name"

            value={searchQuery}

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

            className="search-input"

        />

        <ReactPaginate

            pageCount={Math.ceil(displayedStockPrices.length / itemsPerPage)}

            pageRangeDisplayed={5} // Broj prikazanih stranica u paginaciji

            marginPagesDisplayed={2} // Broj stranica koje će biti prikazane pre i posle trenutne stranice

            previousLabel={'Previous'}

            nextLabel={'Next'}

            breakLabel={'...'}

            onPageChange={({ selected }) => setCurrentPage(selected)}

            containerClassName={'pagination'}

            activeClassName={'active'}

            />

      <table className="firme-table">

        <thead>

          <tr>

            <th>Ticker</th>

            <th>Name</th>

            <th>Price</th>

            <th>Exchange</th>

            <th>Last Updated</th>

          </tr>

        </thead>

        <tbody>

          {displayedStockPrices.map((stock, index) => (

           <StockRow

                key={index}

                ticker={stock.ticker}

                name={stock.name}

                price={stock.price}

                exchange={stock.exchange}

                updated={stock.updated}

         />

          ))}

        </tbody>

      </table>

    </div>

  );

};

export default StockPrices;

Kod ispod je namenjen za prikazivanje liste firmi sa osnovnim informacijama poput naziva, adrese, imena vlasnika i PIB-a. Koristi se prilagođena kuka useFirme koja omogućava dohvatanje podataka o firmama sa servera, uzimajući u obzir autorizaciju putem tokena koji se prenosi kao header u zahtevima. Korisnicima je omogućeno da pretražuju firme kroz dinamički unos, gde se lista automatski ažurira u skladu sa terminom pretrage. Dodatno, postoji opcija za sortiranje firmi po nazivu u rastućem ili opadajućem redosledu, čime se poboljšava preglednost i omogućava korisnicima lakše snalaženje. Prikaz svake firme je realizovan kroz posebnu komponentu Firma, koja formatira prikaz informacija u tabeli. U slučaju da se podaci još uvek učitavaju ili dođe do greške, prikazuju se odgovarajuće poruke korisnicima, čime se poboljšava korisničko iskustvo i daje povratna informacija o statusu zahteva.

import React from 'react';

import './Firma.css';

const Firma = ({ firma }) => {

  return (

    <tr className="firma-row">

      {/\* <td><img src={firma.logo} alt={`Logo za ${firma.naziv}`} /></td> \*/}

      <td>{firma.naziv}</td>

      <td>{firma.adresa}</td>

      <td>{firma.vlasnik.ime} {firma.vlasnik.prezime}</td>

      <td>{firma.PIB}</td>

    </tr>

  );

};

export default Firma;

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

import axios from 'axios';

import Firma from './Firma';

import './Firma.css';

import useFirme from '../kuke/useFirme';

const Firme = () => {

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

  const [sortOrder, setSortOrder] = useState("asc");

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

  const { firme, loading, error } = useFirme('http://127.0.0.1:8000/api/firme', token);

  const handleSearch = (value) => {

    setSearchTerm(value.toLowerCase());

  };

  const toggleSortOrder = () => {

    setSortOrder(sortOrder === "asc" ? "desc" : "asc");

  };

  const sortedFirme = [...firme].sort((a, b) => {

    const firmaA = a.naziv.toLowerCase();

    const firmaB = b.naziv.toLowerCase();

    if (firmaA < firmaB) {

      return sortOrder === "asc" ? -1 : 1;

    }

    if (firmaA > firmaB) {

      return sortOrder === "asc" ? 1 : -1;

    }

    return 0;

  });

  const filteredFirme = sortedFirme.filter((firma) => {

    const { naziv, PIB, vlasnik, adresa } = firma;

    const vlasnikImePrezime = `${vlasnik.ime} ${vlasnik.prezime}`.toLowerCase();

    return (

      naziv.toLowerCase().includes(searchTerm) ||

      PIB.toLowerCase().includes(searchTerm) ||

      adresa.toLowerCase().includes(searchTerm) ||

      vlasnikImePrezime.includes(searchTerm)

    );

  });

  return (

    <div className="firme-container">

      {loading && <p>Učitavanje...</p>}

      {error && <p>Došlo je do greške prilikom učitavanja.</p>}

      {!loading && !error && (

        <>

          <h1>Firme</h1>

          <div className="search-container">

            <input

              type="text"

              placeholder="Pretraži firmu..."

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

            />

          </div>

          <div className="sort-container">

            <button onClick={toggleSortOrder}>

              Sortiraj po nazivu: {sortOrder === "asc" ? "Rastuće" : "Opadajuće"}

            </button>

          </div>

          <table className="firme-table">

            <thead>

              <tr>

                <th>Naziv</th>

                <th>Adresa</th>

                <th>Vlasnik</th>

                <th>PIB</th>

              </tr>

            </thead>

            <tbody>

              {filteredFirme.map((firma) => (

                <Firma key={firma.id} firma={firma} />

              ))}

            </tbody>

          </table>

        </>

      )}

    </div>

  );

};

export default Firme;

# Korisničko uputstvo

Pocetna stranica izgleda kao na slici ispod

Slika na kojoj se nalazi tekst, snimak ekrana, Font

Opis je automatski generisan

Stranica za login

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

Opis je automatski generisan

Stranica za registraciju

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

Opis je automatski generisan

Stranica na kojoj korisnici mogu da vide cene akcija za popularne firme. Na ovoj stranici je moguce pretrazivati firme, a izvrsena je i paginacija

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

Opis je automatski generisan

Stranica na kojoj su ucitane sve firme iz baze podataka. Na ovoj stranici je moguce pretrazivati firme, kao i sortirati ih po nazivu.

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

Opis je automatski generisan