Razvoj web aplikacije upotrebom React razvojnog okvira

Paun, Gordan

Undergraduate thesis / Završni rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Tourism and Rural Development in Pozega / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet turizma i ruralnog razvoja u Požegi**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:277:194492

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-11-22**

Repository / Repozitorij:

FTRR Repository - Repository of Faculty Tourism and Rural Development Pozega

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET TURIZMA I RURALNOG RAZVOJA U POŽEGI

GORDAN PAUN, 0016105736

Razvoj web aplikacije upotrebom React razvojnog okvira

ZAVRŠNI RAD

Požega, 2024. godine.

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET TURIZMA I RURALNOG RAZVOJA U POŽEGI

PRIJEDIPLOMSKI STUDIJ ELEKTRONIČKO POSLOVANJE I PROGRAMSKO INŽENJERSTVO

Razvoj web aplikacije upotrebom React razvojnog okvira

ZAVRŠNI RAD

IZ KOLEGIJA Programiranje web aplikacija II

MENTOR: prof. dr. sc. Krešimir Nenadić

STUDENT: Gordan Paun

JMBAG studenta: 0016105736

Požega, 2024. godine

SAŽETAK

Aplikacija “VEG2GO“ izrađena je u svrhu demonstracije izrade web aplikacije korištenjem React razvojnog okvira. Cilj aplikacije je pokazati proces razvoja od početne instalacije alata, postavljanja projekta, baze podataka, do implementacije ključnih funkcionalnosti koje dinamički upravljaju sadržajem. React izgrađuje klijentsku stranu aplikacije, dok uz pomoć alata XAMPP i Apache poslužitelja, se izvršavaju PHP skripte koje se brinu za poslužiteljsku stranu aplikacije, odnosno čitanje, brisanje i pisanje iz baze podataka korisnika.

Aplikacija uključuje funkcionalnosti kao što su prijava i registracija korisnika, personalizirani prikaz sadržaja, mogućnost ažuriranja vlastitih podataka i administracijsko sučelje za upravljanje korisnicima. S pomoću `react-router-dom` paketa omogućena je navigacija kroz različite stranice aplikacije bez potrebe za njihovim ponovnim učitavanjem. Uz korištenje `react-bootstrap` paketa, aplikacije postiže responzivan i vizualno privlačan dizajn. Prikazan je način manipulacije i kontroliranja komponenti upotrebom kuka `useState` i `useEffect`.

“VEG2GO“ aplikacija služi kao praktičan primjer kako se uz React može izgraditi moderna, responzivna, funkcionalna web aplikacija koja uvelike zadovoljava standarde suvremenog web razvoja.

Ključne riječi: React razvojni okvir, XAMPP, Apache poslužitelj, MariaDB, hooks, dinamički sadržaj, react-router-dom, react-bootstrap

ABSTRACT

The “VEG2GO“ application was developed to demonstrate the creation of a web application using the React framework. The aim of the application is to showcase the development process, from the initial installation of tools, project setup, and database configuration, to the implementation of functionalities that dynamically manage content. React builds the client side of the application, while with the help of XAMPP and Apache server, PHP scripts are executed to handle the server side of the application, including reading, deleting, and writing data from the user database.

The application includes functionalities such as user login and registration, personalized content display, the ability to update personal data, and an administrative interface for user management. The `react-router-dom` package facilitates smooth navigation across different application pages, eliminating the need for full page reloads. With the use of the `react-bootstrap` package, the application achieves a responsive and visually appealing design. The application showcases the use of hooks, specifically `useState` and `useEffect`, for managing and controlling component state and side effects.

The “VEG2GO” application serves as a practical example of how React can be used to build a modern, responsive and functional web application that meets the standards of today’s web development.

Keywords: React framework, XAMPP, Apache server, MariaDB, hooks, dynamic content, react-router-dom, react-bootstrap

SADRŽAJ

1. UVOD

U današnjem digitalnom dobu, razvoj web aplikacija je ključan faktor za ostvarivanje interaktivnih korisničkih iskustava. Potrebe tržišta zahtijevaju skalabilna i učinkovita programska rješenja koja trebaju zadovoljiti sve kompleksnije zahtjeve korisnika. Samim time potaknut je razvoj tehnologija za izradu web aplikacija. Jedna od takvih modernih tehnologija je React, koji omogućava brzi razvoj dinamičkih i responzivnih korisničkih sučelja. U ovom završnom radu, postavlja se fokus na izradu web aplikacije pod nazivom "VEG2GO" korištenjem React razvojnog okvira. Aplikacija prikuplja i organizira informacije o veganskom načinu života s ciljem promicanja veganstva i pružanja korisnih resursa za one koji su zainteresirani za vegansku prehranu.

U sljedećem poglavlju pobliže se upoznaje React razvojni okvir, ali se daje i osvrt na ostale tehnologije za razvoj web aplikacija. Objašnjavaju se različiti pristupi korištenja određenih tehnologija te njihove prednosti i specifičnosti. Također, uspoređuju se s Reactom kako bi se istaknule ključne razlike, prednosti i mane u razvoju web aplikacija. Opisane su tehnologije koje se u praksi paralelno koriste uz React razvojni okvir, ali i one koje ga zamjenjuju.

U trećem poglavlju se opisuju funkcionalni zahtjevi web aplikacije "VEG2GO", odnosno što sve aplikacija treba omogućiti korisnicima. Dalje se detaljno opisuje način izrade aplikacije korištenjem Reacta. Opisuju se korišteni paketi i pobliže se objašnjava izrada pojedinih komponenti koje zajedno kombinirane čine potpunu aplikaciju. Za potrebe opisivanja mogućnosti Reacta, aplikacija koristi bazu podataka korisnika i demonstrira kako se sadržaj dinamički ažurira u skladu s korisničkim radnjama.

U četvrtom poglavlju prikazuje se rad same aplikacije "VEG2GO". Opisuju se načini korištenja aplikacije, korisničko sučelje te na koji način su implementirane različite funkcionalnosti. Poglavlje opisuje kretanje korisnika u aplikaciji i prikazuje dinamičko mijenjanje sadržaja uzrokovano korisničkim radnjama.

2. PREGLED TEHNOLOGIJA ZA IZRADU WEB APLIKACIJA

Razvoj web aplikacija u današnje doba obuhvaća široki spektar različitih tehnologija i alata koji omogućuju izgradnju responzivnih, složenih i dinamičkih korisničkih sučelja. Različite tehnologije imaju svoje posebnosti i prednosti, a izbor tehnologije ovisi o specifičnim zahtjevima projekta i složenosti web aplikacije. U nastavku je pregled popularnih i često korištenih rješenja za izradu web aplikacija.

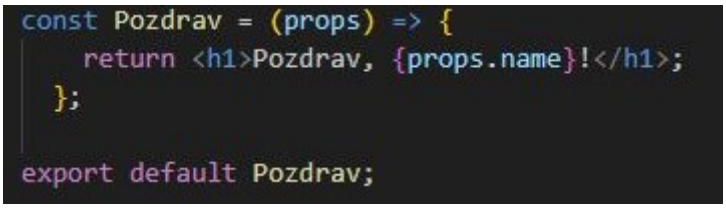
2.1. React

React je popularna JavaScript biblioteka koju je razvila tvrtka Facebook u 2013. godini. Koristi se prvenstveno za izradu korisničkih sučelja i posebno za izradu interaktivnih web aplikacija te se fokusira na brzinu, jednostavnost i skalabilnost. Za razliku od tradicionalnih razvojnih okvira, React se isključivo fokusira na „view" sloj u MVC (engl. *Model-View-Controller*) arhitekturi što ga čini jednostavnijim za učenje i primjenu jer je naglasak na izgradnji komponenata korisničkog sučelja bez potrebe za upravljanjem složenim logičkim strukturama (Mardan, 2017).

Deklarativni pristup programiranju je jedna od ključnih inovacija Reacta. Umjesto da se objektnim modelom dokumenta (DOM) direktno manipulira, u Reactu se korisničko sučelje definira za određeno stanje aplikacije. React koristi virtualni objektni model dokumenta kako bi pri promjeni stanja u aplikaciji prvo ažurirao virtualni DOM te ga usporedio s prethodnim stanjem i na najefikasniji način ažurirao stvarni DOM bez potrebe za ponovnim učitavanjem stranice. Samim time se povećava brzina aplikacije jer se smanjuje broj operacija potrebnih za manipulaciju DOM-om.

Komponente su osnovni građevni blokovi te svaka komponenta predstavlja zaseban dio korisničkog sučelja, kao na primjer gumb ili polje za unos, ali i cijela stranica. React omogućuje hijerarhijsku organizaciju komponenti olakšavajući izgradnju složenijih korisničkih sučelja iz manjih dijelova koji se mogu više puta koristiti u različitim dijelovima aplikacije. Komponente mogu upravljati vlastitim stanjem (engl. *stateful*) ili prikazivati podatke koje primaju putem svojih svojstava (engl. *stateless*).

JSX (engl. *JavaScript XML*) je inovacija Reacta koja omogućuje da se sintaksa HTML-a (engl. *HyperText Markup Language*) piše unutar JavaScript koda što pruža pojednostavljeni i jasniji razvoj komponenata jer se omogućuje intuitivnije povezivanje logike i prikaza u jednom jeziku (React, 2024, url).

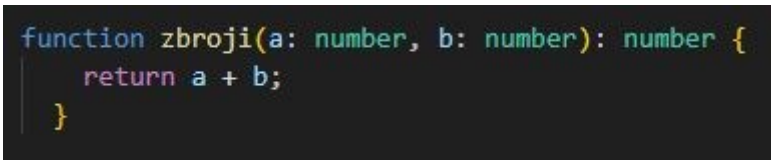


Slika 1 – JSX primjer

Slika 1 prikazuje primjer korištenja JSX-a za jednostavnu komponentu koja putem svojstava (props) prima podatke od svoje nadređene komponente, a u ovom primjeru su to podaci o korisniku, odnosno korisnikovo ime.

2.2. Angular

Angular je razvojni okvir klijentskog dijela aplikacije kojeg je razvio Google 2010. godine. Namijenjen je razvoju dinamičkih i responzivnih web aplikacija. Danas Angular koristi TypeScript, što je prošireni JavaScript koji omogućava statičko tipiziranje, odnosno definiranje tipova podataka varijabli, parametara funkcija ili povratnih vrijednosti funkcija. TypeScript kod se prevodi u JavaScript kod prije nego što se izvrši, što ga čini kompatibilnim s bilo kojim JavaScript okruženjem (TypeScript 2024, url). Slika 2 prikazuje jednostavnu funkciju za zbrajanje dva broja u TypeScriptu, tip parametara funkcije kao i povratni tip su definirani, odnosno funkcija će javljati grešku ako parametar nije brojčanog tipa, što ne bi bio slučaj kod JavaScripta.



Slika 2 – TypeScript primjer

Osim TypeScripta, Angular također pruža funkcionalnosti kao što su dvostrano vezanje podataka, ugrađene module za HTTP (engl. *HyperText Transfer Protocol*) pozive, servise za upravljanje stanjem te alate za testiranje. Dvostrano vezanje podataka omogućuje automatsko ažuriranje sučelja prilikom promjene u modelu, što ubrzava razvoj aplikacije. U Angularu su aplikacije podijeljene na module, komponente i servise te se omogućuje bolje upravljanje složenijim aplikacijama. Moduli predstavljaju osnovne jedinice aplikacije koje povezuju dijelove aplikacije u funkcionalne grupe. Komponente su građevne jedinice korisničkog sučelja te svaka komponenta kontrolira određeni dio sučelja, kao i u Reactu. Servisi predstavljaju

logiku između različitih komponenti kao što je rukovanje podacima ili HTTP zahtjevi (Angular, 2024, url)

U usporedbi s Reactom, Angular je češći izbor za veće aplikacije koje zahtijevaju složeniju strukturu jer on dolazi s već unaprijed ugrađenim rješenjima za koje bi u Reactu bilo potrebno dodati određene biblioteke. Jedan od takvih primjera je ugrađeni sustav za rutiranje kojim se može navigirati između različitih pogleda unutar aplikacije. Sustav rutiranja upravlja učitavanjem komponenti na temelju URL-a (engl. *Uniform Resource Locator*), dok kod Reacta bi bilo potrebno dodati biblioteku za rutiranje. Angular je također složeniji i teži za naučiti od Reacta, no to ne umanjuje činjenicu da se React može koristiti za veća aplikacijska rješenja, što potvrđuje činjenica da su neke od najpoznatijih web aplikacija rađene u Reactu, kao što su Facebook, Airbnb i Instagram. Angularom su izrađene popularne aplikacije kao što su Gmail, PayPal, Upwork, Microsoft Office Online i dr.

2.3. Vue.js

Vue.js je progresivni JavaScript razvojni okvir kojeg je razvio Evan You. Napravljen je s ciljem pojednostavljivanja integracije s već postojećim projektima te je također prilagodljiv za izradu složenijih aplikacija. Veoma je popularan izbor za programere koji žele brz početak s minimalno prepreka, jer je jedna od ključnih karakteristika ovog razvojnog okvira njegova jednostavnost, ali i pristupačnost detaljne dokumentacije koja olakšava samo učenje i primjenu razvojnog okvira (Vue.js, 2024, url).

Vue, kao Angular i React, koristi virtualni DOM i dvostrano vezanje podataka, no s puno lakšom sintaksom. Nudi mogućnost kompozicije komponenti i reakciju na promjenu podataka što omogućava razvoj dinamičkih aplikacija. Vue omogućava lakšu integraciju s malim projektima te postupnu prilagodljivost, a može se koristiti kao biblioteka za određene dijelove aplikacije ili kao razvojni okvir za složenije aplikacije. Kao i kod Angulara, postoje već uključena rješenja koja bi se kod Reacta trebala ubaciti kao posebne biblioteke.

Nasuprot Angularu, Vue je jednostavniji za naučiti od Reacta te je samim time i popularan izbor kod projekata gdje je potrebna integracija ili brz razvoj interaktivnih korisničkih sučelja. Najpopularnije aplikacije koje koriste Vue.js su Alibaba, Xiaomi i Behance.

2.4. Laravel

Laravel je PHP (engl. *PHP: Hypertext* Preprocessor) razvojni okvir koji se temelji na već spomenutoj MVC arhitekturi. Razvijen je 2011. godine kako bi se pojednostavnio složeni razvoj web aplikacija koristeći veoma čitljivu sintaksu. Ovaj razvojni okvir nudi široku lepezu alata i funkcionalnosti kao što su sustavi za autentikaciju, autorizaciju i upravljanje sjednicama. Neki od najpopularnijih alata su Artisan za upravljanje određenim zadacima unutar same aplikacije i Blade templating engine za upravljanje pogledima aplikacije (Laravel, 2024, url).

Najčešća primjena Laravela je isključivo za razvoj poslužiteljskog dijela aplikacije razvoj aplikacija, posebice zbog lakoće rješavanja složene logike i pojednostavljenim upravljanjem podataka. Vrlo često se koristi u kombinaciji s Reactom. Laravel služi kao razvojni okvir poslužiteljskog dijela aplikacije, a React kao biblioteka za razvoj klijentskog dijela aplikacije, dakle Laravel upravlja bazama podataka, autentikacijom i logikom, dok React pospješuje dinamičko korisničko sučelje. Dakle, razlika između Laravela i Reacta se očituje u načinu na koji su dijelovi aplikacije izrađeni i gdje će se izvršavati, prvi je usmjeren na poslužiteljsku stranu dok je drugi na klijentsku stranu.

Laravel predstavlja veoma čest izbor za razvoj složenijih web aplikacija koje zahtijevaju visoku razinu prilagodljivosti i integracije s različitim bazama podataka i servisima.

2.5. Node.js

Node.js je okruženje za izvršavanje JavaScripta izvan preglednika, čime se omogućava razvoj aplikacija korištenjem JavaScripta na poslužiteljskoj strani. Posebno je popularan za izradu aplikacija u stvarnom vremenu, a zbog svog asinkronog modela ima sposobnosti upravljanja velikim brojem istovremenih veza s jako malim kašnjenjem te se često koristi za izradu igara ili aplikacija za razgovor. Omogućuje korištenje modula te se funkcionalnosti mogu bolje organizirati i ponovno koristiti(Node, 2024, url).

Node.js koristi Node Package Manager (`npm`), odnosno najveći upravitelj paketa za JavaScript ekosustav na svijetu, koji omogućava brzo korištenje različitih biblioteka i alata (npm, 2024, url). `npm` danas broji preko milijun dostupnih paketa, što značajno ubrzava razvoj aplikacija i uvođenje potrebnih funkcionalnosti. Slika 3 prikazuje jednostavan kod za pokretanje HTTP servera na portu 3000.

Slika 3 – Node.js pokretanje HTTP poslužitelja

Kao i Laravel, Node.js se izvršava na poslužiteljskoj strani dok se React izvršava na klijentskoj strani. Također se zajedno koriste za izradu složenijih web aplikacija. Nadalje i jedan i drugi koriste JavaScript jezik što omogućava da se koristi isti jezik kroz cijelu aplikaciju. Node.js omogućuje React aplikacijama prikazivanje na strani poslužitelja, što je poznato kao SSR (engl. *server-side rendering*). SSR-om se omogućuje generiranje HTML-a na samom poslužitelju prije nego se sadržaj pošalje korisniku, čime se osigurava brže učitavanje web aplikacije i bolje performanse. Samim time se poboljšava optimizacija za tražilice, jer se lakše indeksira sadržaj stranice koji je već generiran na poslužitelju.

Kombinacija Node.js-a i Reacta omogućuje razvoj visoko skalabilnih i efikasnih aplikacija. Node.js izgrađuje poslužiteljski dio za popularne web aplikacije kao što su Netflix, PayPal, Uber i LinkedIn.

3. POSTUPAK IZRADE APLIKACIJE

Za potrebe završnog rada izrađena je informativna web aplikacija pod nazivom "VEG2GO", koja pruža informacije o fiktivnoj mobilnoj aplikaciji namijenjenoj veganima. Web aplikacija razvijena je korištenjem React razvojnog okvira za klijentsku stranu i XAMPP-a za poslužiteljsku stranu aplikacije.

U nastavku će biti opisani funkcionalni zahtjevi koje aplikacija treba zadovoljiti, detaljan postupak izrade same aplikacije i ključne značajke React razvojnog okvira koje su korištene.

3.1. Funkcionalni zahtjevi aplikacije

Prije same izrade aplikacije, postavljaju se funkcionalni zahtjevi koji definiraju funkcionalnosti i ponašanje aplikacije kako bi se zadovoljio sam cilj nastanka, kao i potrebe korisnika. Funkcionalni zahtjevi opisuju što aplikacija treba raditi, a ne način na koji će se to izvesti.

Web aplikacija "VEG2GO" je informativnog karaktera, no u svrhu demonstracije rada Reacta, određuju se zahtjevi koji premašuju potrebe informativne web aplikacije. Prema tome, postavljaju se korisničke funkcionalnosti, koje u stvarnom svijetu nemaju smisla za web aplikaciju koja opisuje rad mobilne aplikacije. Potrebno je omogućiti korisnicima registraciju i prijavu sa svojim korisničkim imenom ili adresom e-pošte, također je potrebno provjeriti je li korisnik popunio sva polja obrasca ispravno. Nadalje, treba omogućiti korisniku pregled svog profila te izmjene svojih osobnih podataka. Svi posjetitelji aplikacije trebaju moći poslati e-poštu s određenim upitom ili problemom s kojim su se susreli izravno s web aplikacije. Potrebno je omogućiti postojanje korisnika s većim ovlastima od običnog korisnika, takav korisnik će moći pristupiti posebnoj stranici u aplikaciji gdje će moći pregledavati postojeće korisnike i dati im administratorske ovlasti. Na istoj stranici administrator može obrisati korisnike.

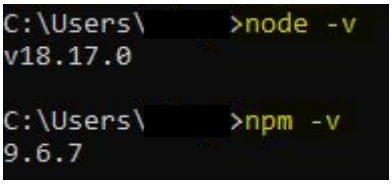
Potrebno je imati navigacijsku traku koja omogućava lako i brzo kretanje između različitih stranica aplikacije, također je potrebno u navigacijskoj traci jasno označiti na kojoj se stranici trenutno korisnik nalazi. Navigacijska traka treba sadržavati dugme za prijavu, ako korisnik nema korisnički račun, treba mu se omogućiti kreiranje računa. Nakon uspješne prijave korisnika, dugme za sadržaj treba imati njegovo ime i prezime. Sadržaj stranice treba sadržavati informacije o veganstvu, prednostima veganske prehrane i o samoj "VEG2GO" mobilnoj aplikaciji, odnosno o problemu kojeg mobilna aplikacija rješava, a s kojim se mnogi vegani susreću.

Sa sigurnosnog aspekta, trebaju se zaštititi korisnički podaci, trebaju biti sigurno pohranjeni i zaštićeni od neovlaštenog pristupa. Također se treba provoditi autorizacija pristupa, odnosno da korisnici s određenim ulogama mogu pristupiti određenom sadržaju aplikacije.

Navedeni funkcionalni zahtjevi određuju što aplikacija "VEG2GO" treba omogućiti svojim korisnicima, navedeni zahtjevi mogu biti prošireni prema specifičnim potrebama aplikacije. Navedeni zahtjevi su dovoljni kako bi se objasnio proces izrade web aplikacije React razvojnim okvirom.

3.2. Kreiranje React projekta

Prije inicijalne izrade React projekta, treba preuzeti Node.js sa službene stranice te slijediti upute za instalaciju. Sa Node.js se instalira i Node upravitelj paketa (`npm`) koji omogućuje instalaciju paketa i alata potrebnih za razvoj React aplikacija. Nakon instalacije može se provjeriti je li instalacija uspješna upisom naredbi sa slike 1 u komandnu liniju.



Slika 4 – Verzije Node.js-a i npm-a

Ako je verzija Node.js-a veća od 14.0.0 i npm-a veća od 5.6.0 onda je podržano kreiranje React projekta s naredbom `npx create-react-app veg2go-app`. Navedena naredba kreira React okruženje bez potrebe konfiguriranja istog od samog početka. U novu kreiranu mapu ulazi se s naredbom `cd veg2go-app` i pokreće se aplikacija na lokalnom poslužitelju s naredbom „npm start" te se omogućuje pregled aplikacije na adresi "http://localhost:3000/" u pregledniku.

Za izradu navedene aplikacije potrebni su dodatni paketi za stilizaciju i upravljanje rutama, točnije React Router DOM i Bootstrap. Navedene pakete instaliramo s naredbom `npm install react-router-dom react-bootstrap bootstrap`. Unutar projekta u zaglavlje `index.js` datoteke se uključuje Bootstrap naredbom `import 'bootstrap/dist/css/bootstrap.min.css'; `.

Bootstrap je CSS (engl. *Cascading Style Sheets*) razvojni okvir za pojednostavljeno stiliziranje web aplikacija i pruža unaprijed definirane komponente. Dovoljno je na element primijeniti odgovarajuću Bootstrap klasu kako bi taj element poprimio unaprijed definirane stilove (React Bootstrap, 2024, url). Pri izradi projekta Bootstrap se koristi isključivo za gumbove, obrasce i modalne prozore, ali i za responzivnost same aplikacije.

React Router DOM je vrlo popularna biblioteka čija je svrha upravljanje navigacijom unutar React aplikacije, omogućuje postojanje više stranica unutar aplikacije bez potrebe za

osvježavanjem stranice tijekom navigacije, jer koristi JavaScript za dinamičku izmjenu DOM-a. U poglavlju `App.js` detaljnije je objašnjeno korištenje React Router DOM-a.

3.3. Baza podataka

Prije daljnjeg razvoja aplikacije, za ostvarivanje funkcionalnih zahtjeva potrebno je izraditi bazu podataka u koju se spremaju podaci o korisnicima i njihove ovlasti. Za izradu baze podataka koristi se XAMPP, popularan razvojni alat koji omogućava postavljanje lokalnog web poslužitelja. Nakon instalacije sa službene stranice i pokretanje MySQL servisa te Apache poslužitelja, pristupa se alatu `phpMyAdmin` putem preglednika na adresi "http://localhost/phpmyadmin/". Navedeni alat omogućuje kreiranje nove baze podataka i tablica u bazi. Web aplikacija "VEG2GO" ima jednostavnu tablicu korisnika naziva "users" u bazi podataka.

	#	Name	Type
☐	1	id	int(11)
☐	2	firstname	text
☐	3	lastname	text
☐	4	email	text
☐	5	username	text
☐	6	userpassword	text
☐	7	userrole	text

Slika 5 – Struktura baze podataka

Struktura tablice (slika 5) sastoji se od sedam atributa od kojih šest popunjava sam korisnik, a stupac `id` je primarni ključ s automatskim inkrementom i univerzalan je za svakog korisnika. Od ostalih stupaca samo se `email` i `username` ne smiju ponavljati, dok ostali nemaju to svojstvo.

3.4. App.js

Komponenta `App.js` automatski se kreira prilikom izrade React aplikacije i u pravilu je definirana kao ulazna točka aplikacije ili glavna komponenta aplikacije. Sadrži osnovni izgled i logiku aplikacije te služi kao okvir unutar kojeg se izgrađuju sve ostale komponente. `App.js` se povezuje s DOM-om u datoteci `index.js` koja je također automatski generirana prilikom izrade projekta.

```
return (
  <BrowserRouter>
    <Routes>
      <Route path="/" element={<Main userdata={userData} handleuserdata={handleChangeUserData} />} >
        <Route index element={<Home userdata={userData}/>} />
        <Route path="userdashboard" element={<UserDashboard userdata={userData} handleuserdata={handleChangeUserData}/>} />
        <Route path="admindashboard" element={<AdminDashboard userdata={userData}/>} />
        <Route path="contact" element={<Contact userdata={userData}/>} />
        <Route path="thanks" element={<Thanks userdata={userData}/>} />
        <Route path="about" element={<About userdata={userData}/>} />
        <Route path="*" element={<ErrorPage />} />
      </Route>
    </Routes>
  </BrowserRouter>
);
```

Slika 6 – `App.js` upravljanje rutama

Komponenta `App.js` upravlja navigacijom i određuje kako će se ostale komponente prikazivati ovisno o URL-u. Koristi se `BrowserRouter`, `Routes` i `Route` (slika 6) komponente koje su iz paketa `react-router-dom`. Glavna ruta prikazuje komponentu `Main.js`, a unutar nje su ugniježđene ostale navigacijske rute. Također prosljeđuje se stanje `userData` i funkcija `handleChangeUserData` kao svojstva (engl. *props*) drugim komponentama te se mogu koristiti unutar drugih komponenti. Funkcija `handleChangeUserData` je identična funkciji stanja `setUserData`.

U `App.js`-u se definiraju različite kuke (engl. *hooks*) kako bi se moglo upravljati korisničkim podacima. Kuke u Reactu označavaju posebne funkcije koje omogućuju korištenje stanja i drugih značajki unutar funkcionalnih komponenata, a najpoznatije kuke su `useState` i `useEffect` (Javatpoint, 2024, url).

```
const [userData, setUserData] = useState({
  username: "",
  firstname: "",
  lastname: "",
  useremail: "",
  userrole: "user"
});
```

Slika 7 – `useState` kuka

Koristi se kuka `useState` kako bi se definirala varijabla stanja `userData` koja je u početku objekt s praznim stringovima i početnoj vrijednosti `user` za ulogu korisnika (slika 7).

Funkcija `setUserData` ažurira stanje varijable `userData` te se koristi isključivo kada se mijenjaju vrijednosti objekta.

```
useEffect(() => {
  const storedUserData = sessionStorage.getItem('userData');
  if (storedUserData) {
    setUserData(JSON.parse(storedUserData));
  }
}, []);

useEffect(() => {
  const handleBeforeUnload = (event) => {
    sessionStorage.setItem('userData', JSON.stringify(userData));
  };

  window.addEventListener('beforeunload', handleBeforeUnload);

  return () => {
    window.removeEventListener('beforeunload', handleBeforeUnload);
  };
}, [userData]);
```

Slika 8 – `useEffect` kuka

Također se koristi kuka `useEffect` koja označava funkciju za upravljanje efektima ili nuspojavama u komponenti. Kod prvog korištenja `useEffect` kuke (slika 8), drugi argument je ` [ ] ` što znači da će se funkcija izvršiti samo jednom i to pri prvoj izgradnji sučelja, navedeni `useEffect` provjerava postoje li korisnički podaci već u pohrani sjednice, i ako postoje onda se poziva funkcija koja mijenja varijablu stanja `userData` te popunjava objekt sa spremljenim korisničkim podacima.

Druga kuka `useEffect` se brine o spremanju objekta `userData` u pohranu sjednice svaki puta kada se stranica napušta ili osvježi. U navedenoj kuki potrebno je naglasiti dodavanje slušatelja događaja te njegovo istovremeno uklanjanje. Slušatelj događaja se dodaje prilikom montiranja (engl. *mounting*) komponente i uklanja se prilikom demontiranja komponente, zbog potencijalnih grešaka gdje će slušatelj događaja postojati iako je neka druga komponenta montirana. Montiranje u Reactu se odnosi na životni ciklus komponente kada se ona kreira i dodaje u DOM. Nasuprot tome, demontiranje je faza životnog ciklusa komponente kada se ona uklanja iz DOM-a jer više nije potrebna ili se treba zamijeniti drugom komponentom.

3.5. Main.js

Za razliku od `App` koja je korijenska komponenta, `Main` komponenta služi za raspored sadržaja na stranici te se brine o njezinoj strukturi. Navedena komponenta se sastoji od navigacijske trake i `Outlet` komponente koja je dio `react-router-dom` paketa.

Potrebno je uključiti korištene komponente na vrhu datoteke kao što je prikazano na slici 9.

```
import { Link, Outlet, useLocation } from "react-router-dom";
import { Dropdown, DropdownButton } from 'react-bootstrap';
import ModalWindow from "./ModalWindow";
```

Slika 9 – Uključivanje komponenti u zaglavlju

Iz istog paketa se još koristi i `Link` komponenta koja omogućuje navigaciju unutar same aplikacije bez ponovnog učitavanja stranice, kao i `useLocation` kuku koja vraća objekt s informacijama o trenutnoj lokaciji za potrebe dodavanja klase linku u navigacijskog traci koji je trenutno otvoren. Prvo se postavlja funkcija koja provjerava odgovara li putanja trenutnoj lokaciji te korištenjem ternarnog operatora se vraća string koji će odrediti klasu, odnosno stil linka kojeg je korisnik otvorio.

```
const location = useLocation();

const isActive = (path) => {
    return location.pathname === path ? "current-menu-item" : "";
};
```

Slika 10 – Funkcija za provjeru trenutne lokacije

Funkcija `isActive` se zatim dodaje kao klasa svim linkovima u navigacijskog traci, a njezini parametri odgovaraju putanji na koju određeni linkovi vode. Prema tome, lijevi dio navigacijske trake izrađuje se na način prikazan na slici 11.

```
<div className="nav-left">
    <Link to="/" className="nav-logo-link"><img src={require("../img/veg2go.png")}/></Link>
    <Link to="/" className={isActive("/")}>Home</Link>
    <Link to="/app" className={isActive("/app")}>The app</Link>
    <Link to="/faq" className={isActive("/faq")}>FAQ</Link>
    <Link to="/about" className={isActive("/about")}>About us</Link>
    <Link to="/contact" className={isActive("/contact")}>Contact</Link>
</div>
```

Slika 11 – JSX dodavanje klase na element

Na desnoj strani navigacijske trake nalazi se dugme, čiji izgled ovisi o stanju varijable `userData`, odnosno ako korisnik nije prijavljen gumb će biti za prijavu, odnosno registraciju. Ako je korisnik prijavljen onda se koristi `DropDownButton` iz paketa `react-bootstrap` i tekst

gumba postaje korisnikovo ime i prezime. Klikom na gumb se otvara padajući izbornik gdje se korisniku omogućuje veza na vlastiti profil i mogućnost odjave (slika 12).

```
{
    (props.userdata.useremail == "") ?
    (
        <div className="d-flex" id="loginButtonDiv">
            <ModalWindow className="get-app-button" handleuserdata={props.handleuserdata}/>
        </div>
    ):(
        <DropdownButton id="dropdown-basic-button" title={props.userdata.firstname + ' ' + props.userdata.lastname}>
            <Dropdown.Item><Link to="/userdashboard">My profile</Link></Dropdown.Item>

            {props.userdata.userrole === "superadmin" && (
                <Dropdown.Item><Link to="/admindashboard">Admin dashboard</Link></Dropdown.Item>
            )}

            <Dropdown.Divider></Dropdown.Divider>
            <Dropdown.Item onClick={ handleLogout }>Logout</Dropdown.Item>
        </DropdownButton>
    )
}
```

Slika 12 – Logika za gumb u navigacijskoj traci

Funkcija `handleLogout` jednostavno koristi proslijeđenu funkciju iz `App` komponente putem svojstava props kako bi se postavile vrijednosti `userData` objekta na prazne stringove. Važno je primijetiti korištenje komponente `ModalWindow`, koja predstavlja modalni prozor za prijavu odnosno registraciju korisnika, također unutar `ModalWindow` komponente je sadržana logika za prijavu i registraciju kao i validacija obrazaca. `ModalWindow` koristi već pripremljene `Modal` komponente iz react-bootstrap paketa.

Polja za unos obrasca za prijavu i registraciju imaju sličnu strukturu, a prikazan je primjer polja za unos korisničkog imena na slici 13.

```
<Form.Floating className="mb-3">
  <Form.Control
    type="text"
    id="username"
    placeholder="Username"
    name='username'
    value={ userData.username }
    onChange={ (e) => handleChangeUserData(e) }/>
  <label htmlFor="username">Username</label>
</Form.Floating>
```

Slika 13 – Kontrola unosa u Reactu

Može se primijetiti da je vrijednost polja za unos kontrolirana varijablom stanja `userData` te se prikazuje trenutna vrijednost korisničkog imena. Nadalje, koristi se `onChange` upravitelj događaja koji poziva funkciju `handleChangeUserData` kad korisnik promjeni sadržaj polja za unos. U Reactu je ključno za sinkronizaciju logike i korisničkog sučelja da se korištenjem stanja kontroliraju komponente, samim time se olakšava validacija unosa i može se spriječiti neautorizirana izmjena. Same promjene stanja unutar komponente izazivaju

izgradnju komponente iznova koja ovisi o tome stanju te je sama aplikacija responzivnija na promjene korisnika.

Funkcija `handleChangeUserData` (slika 14), koja se poziva na svaku korisnikovu izmjenu unutar polja za unos, služi za ažuriranje stanja `userData` objekta. Kada se stanje ažurira automatski se mijenja vrijednost atributa `value` unutar polja za unos. Funkcija referencira polje za unos koje je uzrokovalo `onChange` događaj te uzima vrijednosti atributa `name` i `value`. Zatim se poziva funkcija `setUserData` kojoj se prosljeđuje parametar koji označava trenutnu vrijednost stanja `userData` te ga ažurira korištenjem spread operatora za kopiranje trenutnog stanja i korištenjem izračunatog imena svojstava kako bi se ažuriralo određeno svojstvo u `userData` objektu. U navedenom primjeru unosa korisničkog imena ažurira se svojstvo `username` te mu se dodjeljuje vrijednost koja odgovara vrijednosti atributa `value`.

```
const handleChangeUserData = (ev) => {
    const { name, value } = ev.target;
    setUserData((curentValue) => {
        return {
            ...curentValue,
            [name]: value
        };
    });
}
```

Slika 14 – Funkcija za ažuriranje korisnikovog unosa

U nastavku `ModalWindow` komponente se nalaze funkcije `handleRegister` i `handleLogin` koje se pozivaju klikom na gumb na kraju obrazaca. Služe za dohvaćanje i slanje podataka između baze podataka i web aplikacije. Navedene funkcije će biti pobliže opisane u PHP potpoglavlju.

Nakon navigacijske trake, postavljena je `Outlet` komponenta koja omogućava dinamičko ubacivanje sadržaja koji se temelji na trenutnoj putanji, odnosno omogućava prikaz ostalih komponenti koje su ugniježđene unutar `Main` komponente što je već prikazano u `App` komponenti u prethodnom potpoglavlju. `Main` komponenta se može zamisliti kao navigacijska traka i njezina logika na vrhu stranice te ovisno o trenutnoj putanji prikazuje se sadržaj različitih komponenti ispod navigacije uz pomoć `Outlet` komponente.

3.6. Contact.js

Komponenta `Contact` sadrži informativni sadržaj o mogućim načinima kontaktiranja u vezi s problemima vezanih uz mobilnu aplikaciju. Na dnu se nalazi obrazac za izravno slanje upita putem e-pošte. Na dugmetu za slanje obrasca se nalazi slušatelj događaja koji se montira pri izgradnji komponente i demontira kada se druga komponenta montira (slika 15).

```
useEffect(() => {
    const form = document.getElementById("contact-form");
    form.addEventListener("submit", validateForm);

    return () => {
        form.removeEventListener("submit", validateForm);
    };
}, []);
```

Slika 15 – `useEffect` kuka koja montira slušatelj događaja

Slušatelj događaja poziva funkciju `validateForm` koja obrađuje validaciju obrasca, provjerava jesu li su sva polja ispravno popunjena, je li format e-pošte ispravan te je li se korisnik složio sa zadanim uvjetima. Unos u polje e-pošte se provjerava regularnim izrazima. Funkcija prvotno zaustavlja uobičajeno ponašanje dugmeta, što je slanje obrasca određenoj skripti, te provjerava jesu li su unosi u obrascu ispravni. Ako funkcija provjeri da je unos ispravan, onda se obrazac šalje skripti i montira se komponenta `Thanks`. Navedena komponenta ispisuje tekst o uspješno poslanoj e-pošti i zahvaljuje se korisniku. U slučaju da obrazac nije ispravno popunjen, ispod obrasca se ispisuje pogreška o netočnom unosu.

Za stvarno slanje e-pošte putem obrasca potreban je SMTP (engl. *Simple Mail Transfer Protocol*) poslužitelj koji je zadužen za slanje i primanje e-pošte. Web aplikacija "VEG2GO" služi kao primjer izrade web aplikacije u React razvojnom okviru te za potrebe iste nije omogućen SMTP poslužitelj, ali će PHP skripta koja bi uz postojanje SMTP servera slala poštu biti objašnjenja u nastavku.

3.7. Korisnička nadzorna ploča

Postoje dvije različite komponente nadzorne ploče, to su `UserDashboard` i `AdminDashboard`. Prva je dostupna svim korisnicima kada se prijave, komponenta se montira klikom na element u padajućoj listi koja se otvara klikom na dugme u navigacijskoj traci. `AdminDashboard` komponenta je vidljiva i pristupačna samo korisnicima s ulogom "superadmin" te joj se također pristupa iz padajućeg izbornika na gumbu u navigacijskoj traci.

`UserDashboard` komponenta sadrži korisničke podatke te posjeduje logiku za izmjenu vlastitih korisničkih podataka. Sadrži jednostavnu logiku za provjeru je li korisnik prijavljen,

ako nije onda se preusmjerava na početnu stranicu. Provjerava se da li postoji e-pošta korisnika koji je proslijeđen putem svojstava props (slika 16). U slučaju da e-pošta u objektu `userData` je prazan string, znači da korisnik nije prijavljen i ne bi trebao vidjeti komponentu `UserDashboard`.

```
const navigate = useNavigate();
useEffect(() => {
    if(props.userdata.useremail == "") {
        navigate("/");
    }
});
```

Slika 16 – Prosljeđivanje neprijavljenog korisnika

Klikom na gumb "Edit" omogućava se izmjena postojećih podataka. Postoje tri funkcije u komponenti, a to su `handleEditClick`, `handleEditClose` i `handleSaveClick`. Sadržaj informacija korisnika ovisi o korisnikovoj akciji, naime koristi se ternarni operator kako bi provjerio stanje varijable `editing` te na temelju njezine istinitosti, sadržaj komponente postaje obrazac za izmjenu podataka s gumbovima za spremanje i zatvaranje obrazca. Ako navedena varijabla predstavlja neistinost, onda se prikazuju samo informacije o korisniku i gumb za uređivanje.

```
const [editing, setEditing] = useState(false);
const [editedData, setEditedData] = useState({ ...props.userdata });

const handleEditClick = () => {
    setEditing(true);
};
const handleEditClose = () => {
    setEditing(false);
};
```

Slika 17 – Promjene `editing` stanja

Funkcija `handleSaveClick` šalje unesene podatke skripti koja mijenja korisnikove podatke u bazi podataka, funkcija nakon uspješnog izvršenja postavlja varijablu `editing` na laž kako bi se obrazac za uređivanje zatvorio. Naravno, polja za unos obrasca se kontroliraju po istom principu kao i polja obrasca za prijavu i registraciju, jedina razlika je u tome što se u slučaju uređivanja podataka ne mijenjaju trenutni korisnički podaci u realnom vremenu, već samo klikom na gumb za spremanje. Svaka izmjena na polju za unos u obrazac poziva funkciju `handleChange` (slika 18), ali navedena funkcija mijenja novi objekt `editedData`.

```
const handleChange = (e) => {
    const { name, value } = e.target;
    setEditedData({ ...editedData, [name]: value });
};
```

Slika 18 – Funkcija za kontrolu unosa u Reactu

`AdminDashboard` komponenta je dostupna samo korisnicima koji imaju ulogu "superadmin" te ako nisu prijavljeni ili ako nemaju navedenu ulogu prosljeđuje se korisnika na početnu stranicu, odnosno sprječava se neovlašteni pristup isto kao i u komponenti `UserDashboard`. Komponenta `AdminDashboard` prikazuje popis svih korisnika i administratoru omogućava brisanje i promjenu ovlasti korisnika. Dakle potrebne su funkcije `handleUserRoleChange` i `handleUserDelete`. Pri montiranju same komponente dohvaćaju se podaci o korisnicima iz baze podataka. Funkcija za izmjenu uloge korisnika `handleUserRoleChange` prosljeđuje skripti `admin.php` korisničko ime i novu ulogu navedenog korisnika, kada skripta pošalje odgovor da je promjena uspješna funkcija ažurira `users` stanje u koje je spremila dohvaćene korisnike iz baze.

```
const handleUserRoleChange = (username, newRole) => {
    fetch("http://localhost/veg2go/admin.php", {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({
            action: 'updateUserRole',
            username: username,
            newRole: newRole
        }),
    })
    .then(response => response.json())
    .then(data => {
        if (data.success) {
            const updatedUsers = users.map(user =>
                user.username === username ? { ...user, userrole: newRole } : user
            );
            setUsers(updatedUsers);
        } else {
            console.error('Failed to update user role:', data.error);
        }
    })
    .catch(error => console.error('Error updating user role:', error));
};
```

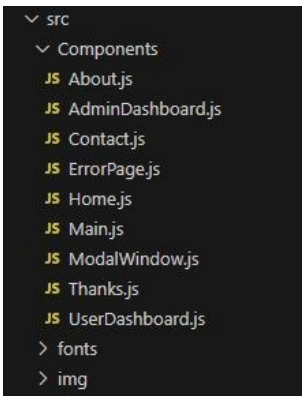
Slika 19 – Funkcija za promjenu uloge korisnika

Logika funkcije `handleUserDelete` je identična, jedina razlika je u tome što kao parametar funkcije je potrebno samo korisničko ime, kada se korisnika uspješno izbriše iz baze podataka stiže odgovor od skripte te se `users` stanje ažurira tako da se filtriraju svi korisnici

osim korisnika čije je korisničko ime proslijeđeno kao parametar. Kada se dogodi promjena u listi korisnika, automatski se promjene očituju u sadržaju aplikacije bez potrebe za osvježavanjem.

3.8. Ostale komponente

Osim već objašnjenih komponenti, dodaju se komponente koje u sebi nemaju specifičnu logiku već služe kao informativni sadržaj web aplikacije.



Slika 20 – Prikaz korištenih komponenti

`Home` komponenta se montira kada se korisnik nalazi na početnoj lokaciji web aplikacije te vraća informativni sadržaj vezan uz vegansku prehranu i način života. Komponenta je podređena komponenti `Main` koja služi kao okvir. Na mjestu `Outlet` komponente unutar `Main` komponente se ispisuje sadržaj `Home` komponente kada je montirana.

Prema istom principu je izgrađena i `About` komponenta te se montira kada korisnik posjeti "About us" vezu u navigaciji. Definirana ruta za `ErrorPage` u `App` komponenti je bilo koja putanja koja nije povezana s niti jednom komponentom. `ErrorPage` služi kako bi se korisnika obavijestilo da stranica na kojoj se nalazi ne postoji.

Komponenta `Thanks` se montira kada se obrazac unutar `Contact` komponente uspješno pošalje. Sve navedene komponente imaju istu logiku kojom su izgrađene i ne sadrže nikakve funkcije. Vrijedno ih je spomenuti jer one upotpunjuju logiku aplikacije te samu navigaciju korisnika. `Thanks` komponenta je najkraći primjer strukture navedenih komponenti i prikazana je na slici 21.

Slika 21 – `Thanks` komponenta

U općem smislu, komponente u Reactu su osnovni građevni blokovi aplikacije te njihovim korištenjem se izgrađuje korisničko sučelje. Postoje dvije vrste komponenti: funkcionalne i klasne. U verziji 16.8. Reacta uvedene se kuke koje omogućavaju funkcionalnim komponentama da imaju sve mogućnosti kao i klasne komponente. Zbog toga se u modernom razvoju aplikacija većinom koriste funkcionalne komponente. Primjer funkcionalne komponente je `Thanks` komponenta (slika 21). Takve komponente su jednostavne JavaScript funkcije koje vraćaju JSX i obično koriste kuke, poput `useState` i `useEffect`, za upravljanje stanjem i efektima. Komponente se većinom sastoje od JSX-a, unutarnjeg stanja komponente, svojstava (props) i životnog ciklusa. Životni ciklus komponente obuhvaća faze montiranja, ažuriranja i demontiranja komponente (Sidelnikov, 2016).

Komponente u Reactu obično se nalaze unutar ".js" ili ".jsx" datoteka. Naredba `export` se koristi za označavanje funkcija, klasa, varijabli ili komponenata koje se želi učiniti dostupnima drugim datotekama. S pomoću `import` naredbe i točne putanje uvoze se funkcionalnosti i komponente u druge datoteke i komponente unutar aplikacije. Ovo omogućava modularnost i organizaciju koda, čime se olakšava njegovo upravljanje i ponovno korištenje.

3.9. PHP

PHP skripte su dizajnirane za rad na strani poslužitelja, a ne klijenta, stoga se za izvršavanje PHP skripti koristi Apache web poslužitelj iz paketa XAMPP. Iz istog paketa koristi se i MariaDB baza podataka za potrebe spremanja podataka o korisnicima. Važno je pojasniti način rada skripti kako bi u potpunosti bila objašnjena izrada web aplikacije.

Skripta `connect.php` je jednostavna skripta koja sadrži podatke za spajanje na bazu podataka te se poziva u svim ostalim skriptama koje trebaju spajanje na bazu podataka.

Skripta za prijavu korisnika od funkcije `fetch` metodom prima podatke o korisničkom imenu ili e-pošti i zaporci. Zaporka se radi sigurnosti hashira MD5 kriptografskom funkcijom

te se kriptira u 32 heksadecimalna znaka kako ni oni koji imaju izravan pristup bazi podataka neće znati lozinku korisnika. Važno je napomenuti da će za svaki ulaz MD5 (engl. *Message Digest Algorithm 5*) funkcija proizvesti isti hash. Nakon toga se postavlja SQL upit bazi podataka koji traži postojanje korisnika s primljenim podacima u bazi podataka. Ako korisnik postoji, skripta u odgovor vraća podatke o korisniku što je vidljivo na slici 22.

```php
$username = $phpData["username"];
$userpass = md5($phpData["pwd"]);
$stmtUN = $conn->prepare("SELECT firstname, lastname, email, userrole, username
FROM users WHERE username = :un AND userpassword = :up");
$stmtUN->execute([":un" => $username, ":up" => $userpass]);
$numRowsUN = $stmtUN->rowCount();
if($numRowsUN > 0) {
    $rowUN = $stmtUN->fetchAll(PDO::FETCH_ASSOC);
    echo json_encode(["response" => $rowUN], true);
}
```

Slika 22 – PHP logika za prijavu korisnika

Skripta za registraciju radi na temelju slične logike, razlika je u broju podataka koji skripta prima, odnosno prima kompletne podatke o korisniku. Također skripta, prije nego zapiše novog korisnika u bazu podataka, treba provjeriti postoji li već korisnik s istim korisničkim imenom ili e-poštom. U slučaju da korisnik ne postoji, novi korisnik i svi njegovi primljeni podaci se upisuju u bazu podataka.

Skripta za ažuriranje korisničkih podataka, koja se poziva iz `UserDashboard` komponente, prima sve nove unesene podatke u obrascu za ažuriranje korisničkih podataka, ali i trenutno korisničko ime i njegovu e-poštu. Skripta nadalje ažurira korisnika u bazi podataka koji odgovara korisničkom imenu i e-pošti korisnika koji je podnio zahtjev za ažuriranjem svojih podataka. Nakon što su podaci ažurirani, skripta kao odgovor funkciji u `UserDashboard` komponenti vraća nove ažurirane korisničke podatke.

Već spomenuta skripta za slanje elektroničke pošte za koju je potreban SMTP poslužitelj prima podatke iz obrasca koji se nalazi u `Contact` komponenti te ih priprema za slanje korištenjem `mail` funkcije u PHP-u (slika 23).

```php
if(isset($_POST['mail'])){
    $to = "sfxgoc@gmail.com";
    $from = $_POST['mail'];
    $name = $_POST['name'];
    $subject = $_POST["subject"];
    $message = $name . " wrote the following:" . "\n\n" . $subject . "\n\n" . $_POST['message'];
    $headers = "From:" . $from;
    mail($to,$subject,$message,$headers);
    header('../thanks');
    }
```

Slika 23 – PHP za slanje elektroničke pošte

Skripta `admin.php` omogućuje administratoru brisanje korisnika i mijenjanje njihovih uloga iz `AdminDashboard` komponente. Definiraju se tri funkcije unutar skripte, za dohvaćanje svih korisnika, za ažuriranje uloge korisnika i za brisanje korisnika (slika 24).

```php
include "connect.php";
1 reference
function getUsers($conn) {
    $stmt = $conn->prepare("SELECT * FROM users");
    $stmt->execute();
    return $stmt->fetchAll(PDO::FETCH_ASSOC);
}
1 reference
function updateUserRole($conn, $username, $newRole) {
    $stmt = $conn->prepare("UPDATE users SET userrole = ? WHERE username = ?");
    $stmt->bindParam(1, $newRole);
    $stmt->bindParam(2, $username);
    return $stmt->execute();
}
1 reference
function deleteUser($conn, $username) {
    $stmt = $conn->prepare("DELETE FROM users WHERE username = ?");
    $stmt->bindParam(1, $username);
    return $stmt->execute();
}
```

Slika 24 – PHP za administratorsko upravljanje korisnicima

Funkcija `getUsers` se poziva prilikom montiranja komponente `AdminDashboard` kako bi se prikazali podaci o svim korisnicima unutar same komponente.

Na temelju akcije koja je poslana `fetch` metodom iz funkcije prikazane na slici 19 unutar `AdminDashboard` komponente, skripta prikazana na slici 25 izvršava ažuriranje uloge korisnika ili brisanje korisnika iz baze podataka.

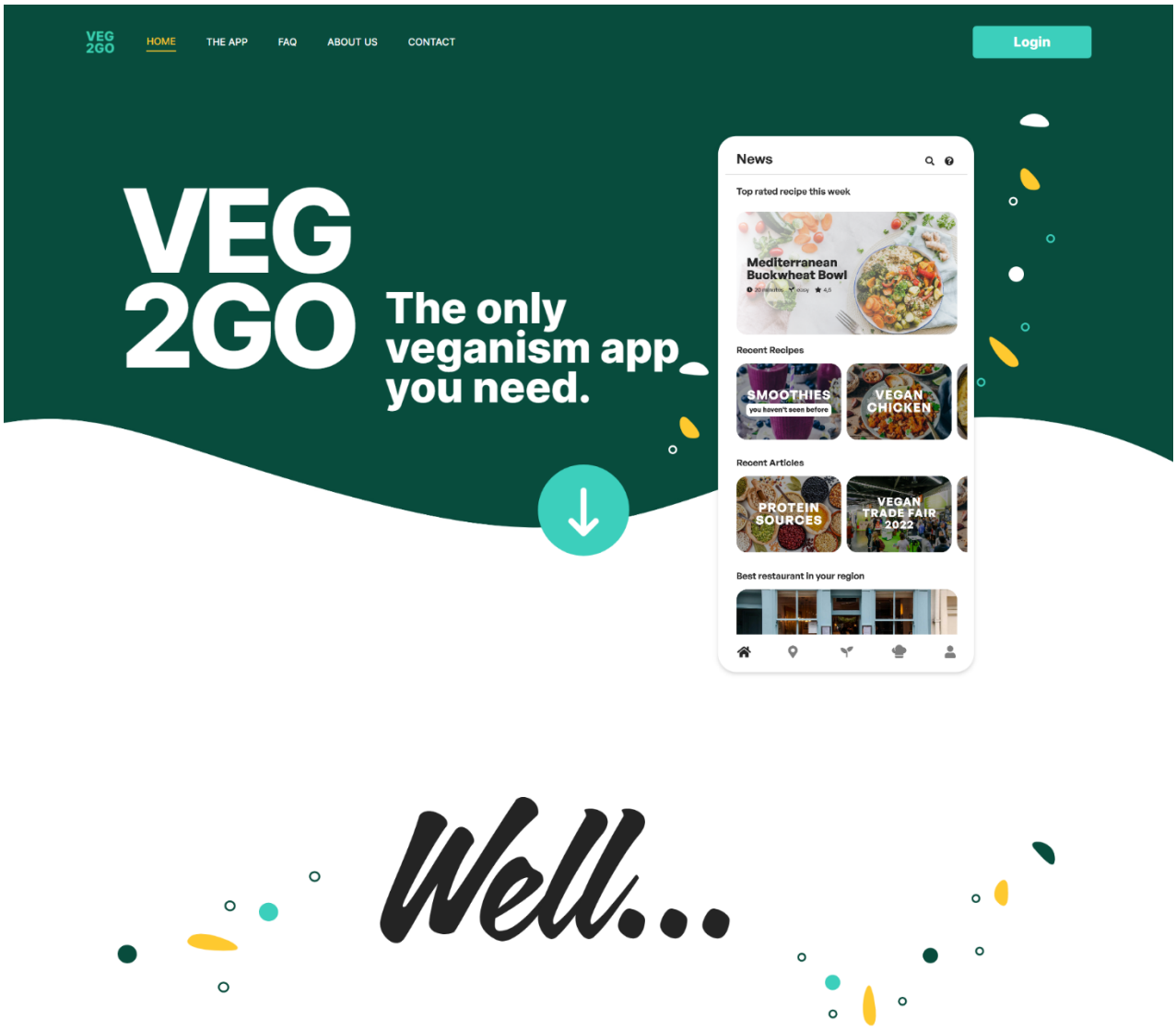
```php
if (isset($postData["action"])) {
    if ($postData["action"] === "updateUserRole") {
        if (isset($postData["username"], $postData["newRole"])) {
            if (updateUserRole($conn, $postData["username"], $postData["newRole"])) {
                echo json_encode(["success" => true]);
            } else {
                echo json_encode(["error" => "Failed to update user role"]);
            }
        }
    } elseif ($postData["action"] === "deleteUser") {
        if (isset($postData["username"])) {
            if (deleteUser($conn, $postData["username"])) {
                echo json_encode(["success" => true]);
            } else {
                echo json_encode(["error" => "Failed to delete user"]);
            }
        }
    }
}
```

Slika 25 – PHP pozivanje različitih funkcija na temelju akcije

Kao odgovor funkciji unutar `AdminDashboard` komponente dostavlja se potvrda o uspješnom ažuriranju ili brisanju korisnika, na temelju tog odgovora funkcije unutar komponente ažuriranju varijablu stanja `users` s novim informacijama o korisnicima te nema potrebe ponovno dohvaćati podatke o korisnicima iz baze podataka.

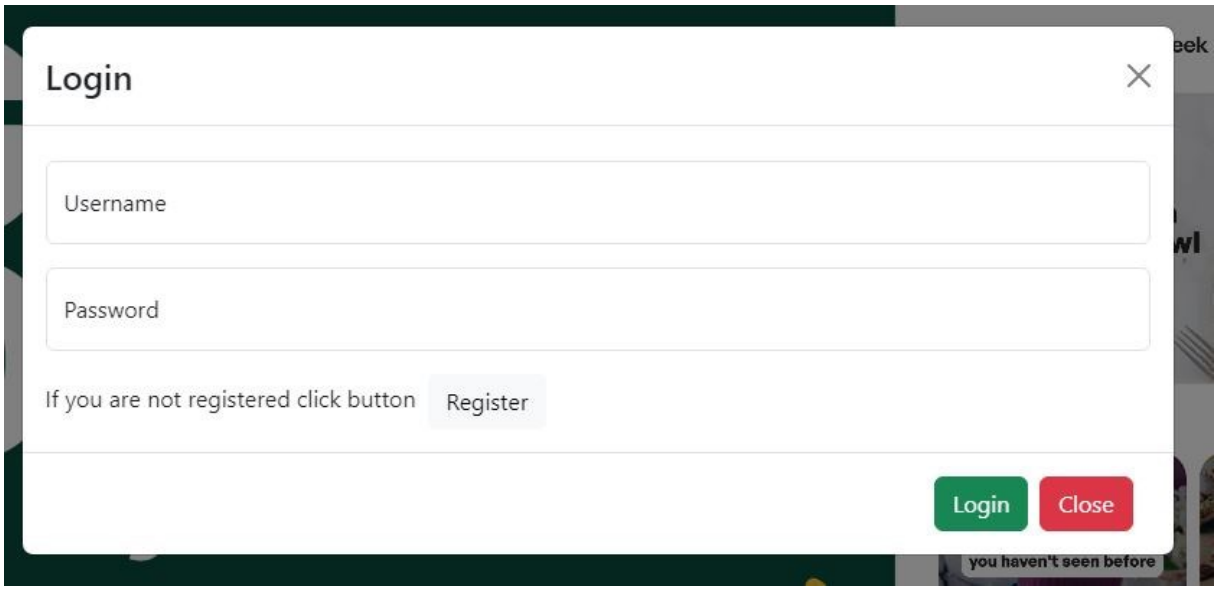
4. PRIKAZ RADA APLIKACIJE

Prilikom pokretanja web aplikacije “VEG2GO“ (slika 26), početna stranica je informativnog karaktera te korisnici koji nisu prijavljeni mogu pregledati čitav sadržaj aplikacije osim sučelja za profil korisnika i administratorskog sučelja.



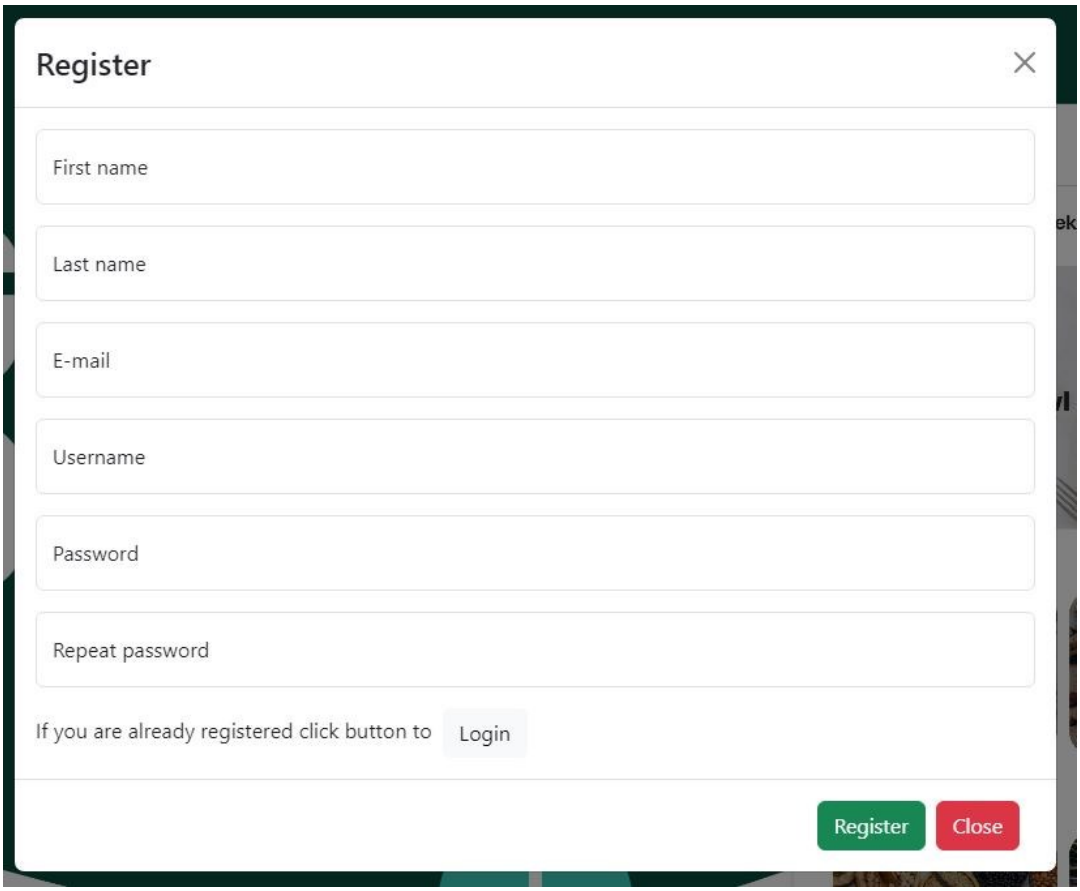
Slika 26 – ”Home” stranica aplikacije “VEG2GO”

Navigacijska traka sadrži logo aplikacije koji ujedno vodi na početnu stranicu, kao i pritisak na “Home“. Na desnoj strani navigacijske trake je gumb za prijavu korisnika, pritiskom na gumb otvara se modalni prozor koji zahtjeva unošenje korisničkog imena ili e-pošte i zaporke (slika 27).



Slika 27 – Modalni prozor za prijavu korisnika

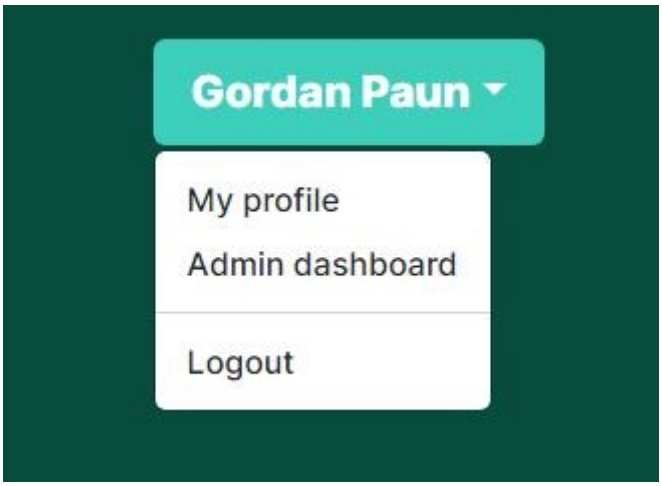
Ako korisnik nema izrađen račun, poziva ga se da klikne na gumb „Register“, zatim se otvara novi modalni prozor za registraciju korisnika (slika 28).



Slika 28 – Modalni prozor za registraciju korisnika

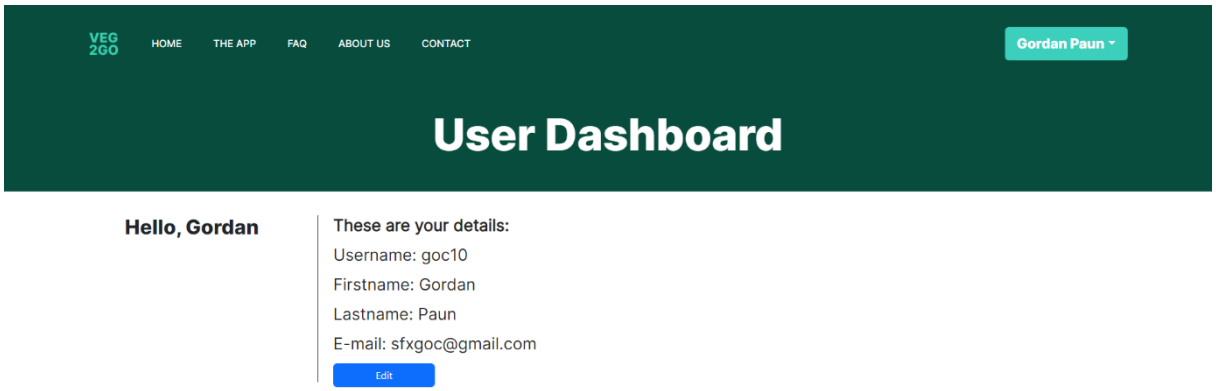
Točnim ispunjavanjem obrasca korisnik može odraditi registraciju klikom na gumb “Register“. Zatim se može prijaviti sa svojim korisničkim podacima.

Nakon uspješne prijave gumb “Login“ u navigacijskog traci postaje padajući gumb s imenom i prezimenom prijavljenog korisnika (slika 29).



Slika 29 – Padajući izbornik na dugmetu prijavljenog korisnika

U otvorenom padajućem izborniku obični korisnik može vidjeti samo svoj profil i odjavu. Ako je korisnik administrator uz navedeno može pristupiti i administratorskoj nadzornoj ploči. Prilikom odabira “My profile“ otvara se profil korisnika koji prikazuje njegove osobne podatke i gumb za mogućnost izmjene svojih podataka (slika 30).

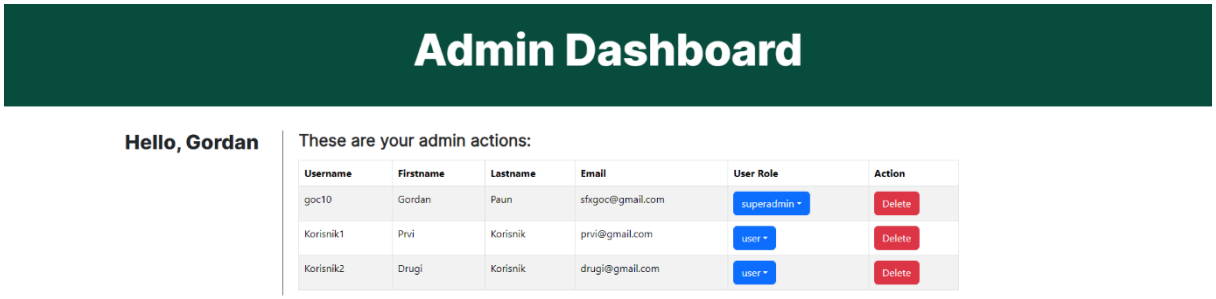


Slika 30 – Korisnički profil

Prilikom klika na gumb “Edit“ mijenja se sučelje (slika 31) i prostor s informacijama o korisnikovim podacima postaje obrazac koji je unaprijed popunjen te omogućava korisniku da promjeni jednu ili više stavku uz uvjet da novo izabrano korisničko ime ili elektronička pošta ne postoji u bazi podataka.

Slika 31 – Obrazac za ažuriranje korisničkih podataka

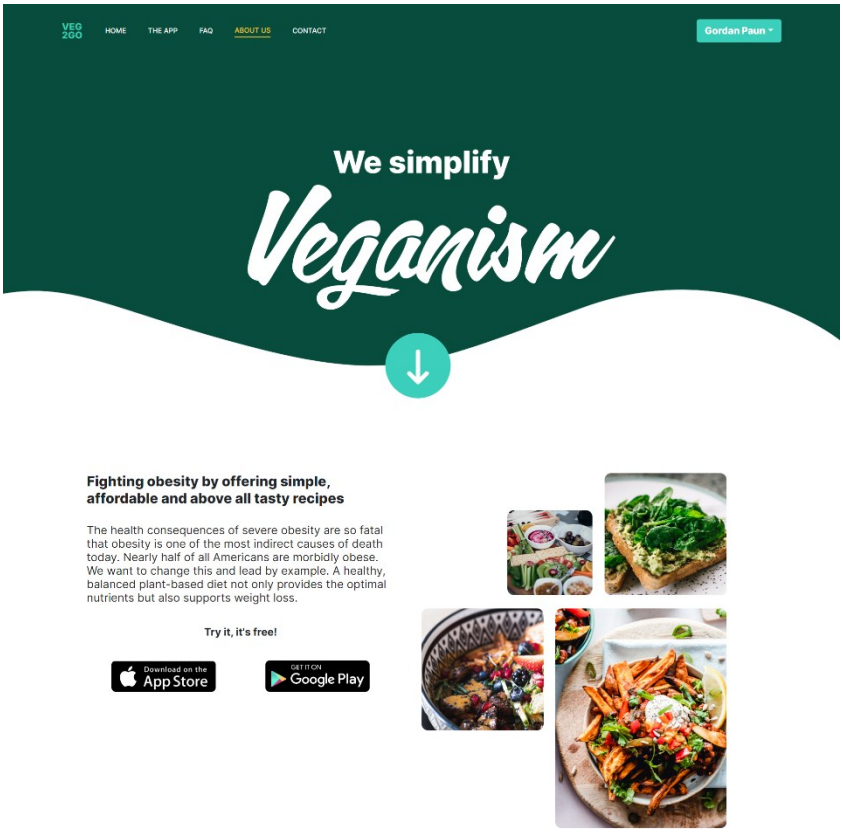
Ako korisnik posjeduje administratorske ovlasti, moguće je posjetiti administratorsku nadzornu ploču. U nadzornoj ploči mogu se pregledati podaci o svim registriranim korisnicima te se iste može obrisati ili im izmijeniti uloga (slika 32).



Slika 32 – Administratorska nadzorna ploča

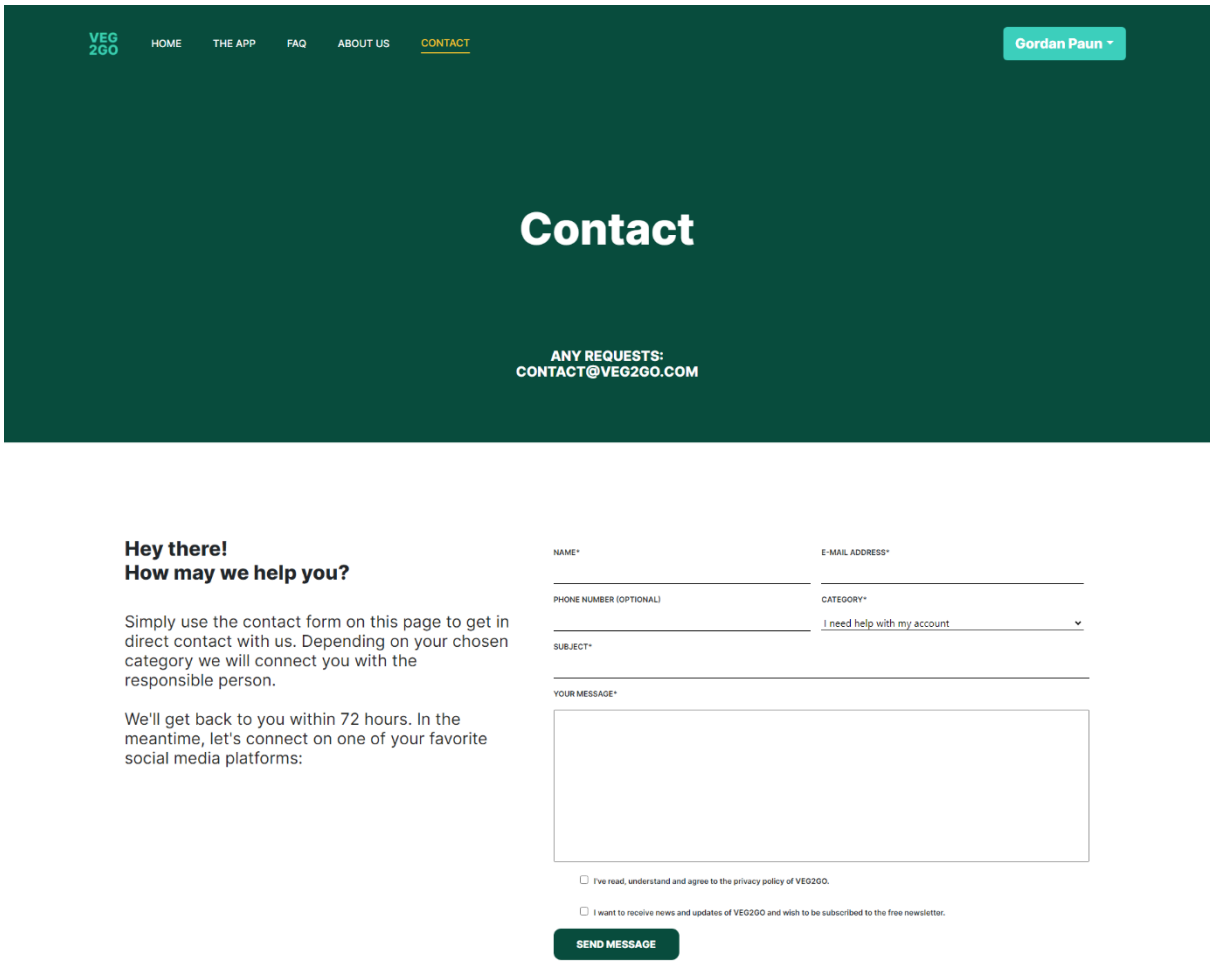
Klikom na ulogu korisnika otvara se padajući izbornik gdje se izabire uloga "superadmin" ili "user".

Stranica "About us" (slika 33) je informativnog karaktera te se ne razlikuje u sadržaju kod različitih korisnika. Isto vrijedi za stranice "The App" i "FAQ". Stranice "The App" i "FAQ" nisu izrađene za potrebe demonstracije izrade aplikacije u React razvojnom okviru.



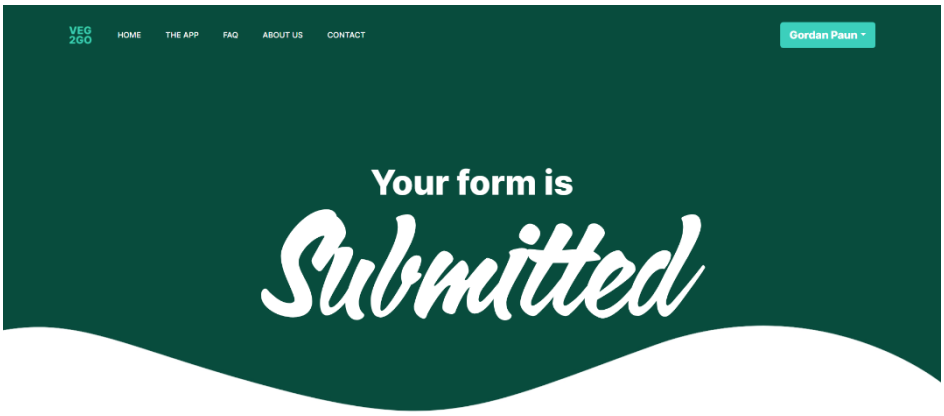
Slika 33 – "About us" stranica

Stranica “Contact“ (slika 34) posjeduje kontakt informacije kao i obrazac za izravno slanje e-pošte.



Slika 34 – “Contact” stranica

Nakon uspješne ispune obrazca, korisnika se preusmjerava na novu stranicu koja potvrđuje da je e-pošta uspješno poslana (slika 35).



Slika 35 – Stranica potvrde poslane elektroničke pošte

5. ZAKLJUČAK

Zadatak ovog završnog rada je izraditi web aplikaciju korištenjem React razvojnog okvira. Izrađena aplikacija “VEG2GO“ predstavlja informativni sadržaj za istoimenu fiktivnu mobilnu aplikaciju, koja će možda biti realizirana u budućnosti. Aplikacija omogućuje prijavu i registraciju korisnika te omogućuje dinamičko mijenjanje sadržaja uzrokovano korisnikovim radnjama. Time je prikazana moć Reactovog virtualnog DOM-a, u stvarnom DOM-u se samo zamjenjuju komponente koje se razlikuju od komponenti virtualnog DOM-a te se izbjegava potreba za osvježavanjem stranice. Drugim riječima, izgrađuju se samo komponente koje su promijenjene, a ne cijela stranica.

Pri izradi aplikacije koristio se i alat XAMPP koji omogućava pokretanje Apache poslužitelja kako bi se PHP skripte mogle izvršiti. XAMPP također omogućuje korištenje MariaDB baze podataka u koju se spremaju podaci o korisnicima. Radi korisnikove sigurnosti lozinka se hashira MD5 algoritmom i kao takva se sprema u bazu podataka. Korisnici se međusobno razlikuju u bazi podataka prema korisničkom imenu i e-pošti, omogućeno je da se korisnik može prijaviti s oba podatka.

Aplikacija se može proširiti tako da korisnici imaju određene mogućnosti koje zaista zahtijevaju njihovu prijavu, kao na primjer postojanje bloga, gdje bi prijavljeni korisnici mogli objavljivati recepte ili informativni sadržaj te također komentirati jedni drugima objave. Za potrebe demonstriranja izrade web aplikacije upotrebom Reacta, postoji logika personaliziranog sadržaja i mogućnosti korisnika različitih uloga te se daljnji razvoj aplikacije može nadograditi i omogućiti veću interakciju između samih korisnika.

Uspješnom implementacijom svih funkcionalnosti i zahtjeva koji su postavljeni, možemo zaključiti da je React snažan alat za izradu interaktivnih web aplikacija te ima mogućnost pružanja vrlo dobrog korisničkog iskustva.

LITERATURA

1. A. Mardan (2017), *React Quickly: Painless web apps with React, JSX, Redux, and GraphQL,* New York: Manning
2. Angular. URL: https://angular.dev/ [pristup: 22.08.2024.]
3. Apache Friends. URL: https://www.apachefriends.org/ [pristup: 24.08.2024.]
4. Create React App. URL: https://create-react-app.dev/ [pristup: 24.08.2024.]
5. G. Sidelnikov (2016), *React.js Book: Learning React JavaScript Library From Scratch*, River Tigris LLC
6. Javatpoint. URL: https://www.javatpoint.com/react-hooks [pristup: 24.08.2024.]
7. Laravel. URL: https://laravel.com/ [pristup: 22.08.2024.]
8. Node.js. URL: https://nodejs.org/en [pristup: 22.08.2024.]
9. npm. URL: https://www.npmjs.com/ [pristup: 22.08.2024.]
10. php. URL: https://www.php.net/ [pristup: 22.08.2024.]
11. React. URL: https://react.dev/ [pristup: 22.08.2024.]
12. React Bootstrap. URL: https://react-bootstrap.github.io/ [pristup: 24.08.2024.]
13. TypeScript. URL: https://www.typescriptlang.org/docs/ [pristup: 22.08.2024.]
14. Vue.js. URL: https://vuejs.org/ [pristup: 22.08.2024.]

POPIS SLIKA

IZJAVA O AUTORSTVU RADA

Ja, **Gordan Paun**, pod punom moralnom, materijalnom i kaznenom odgovornošću, izjavljujem da sam isključivi autor završnog/diplomskog rada pod naslovom: **Razvoj web aplikacije upotrebom React razvojnog okvira** te da u navedenom radu nisu na nedozvoljen način korišteni dijelovi tuđih radova.

U Požegi, 11. rujan 2024.

Potpis studenta