Usporedba React.js i Vue.js radnih okvira za razvoj web aplikacija

Županović, Luka

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Preddiplomski sveučilišni studij računarstva

Završni rad

Usporedba React.js i Vue.js radnih okvira za razvoj web aplikacija

Rijeka, srpanj 2022.

Luka Županović

0069085842

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Preddiplomski sveučilišni studij računarstva

Završni rad

Usporedba React.js i Vue.js radnih okvira za razvoj web aplikacija

Mentor: doc.dr.sc. Marko Gulić

Rijeka, srpanj 2022.

Luka Županović

0069085842

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA ZAVRŠNE ISPITE

Rijeka, 7. ožujka 2022.

Zavod: **Zavod za računarstvo**
Predmet: **Razvoj web aplikacija**
Grana: **2.09.06 programsko inženjerstvo**

ZADATAK ZA ZAVRŠNI RAD

Pristupnik: **Luka Županović (0069085842)**
Studij: Preddiplomski sveučilišni studij računarstva

Zadatak: **Usporedba React.js i Vue.js radnih okvira za razvoj web aplikacija / The comparison of React.js and Vue.js frameworks for web application development**

Opis zadatka:

Treba napraviti detaljnu usporedbu React.js i Vue.js radnih okvira klijentske strane koji se koriste za razvoj web aplikacija. Usporedba će se izvršiti na način da se razviju dvije identične RESTful jednostranične aplikacije (Single Page Application, SPA) pomoću zadanih radnih okvira koje će u sebi sadržavati osnovne CRUD (create, read, update, delete) operacije i autentifikaciju. Treba analizirati mogućnosti, arhitekturu i načine razvoja web aplikacija pomoću ova dva radna okvira. Za razvoj poslužiteljskog dijela web aplikacije treba koristiti Laravel 8 radni okvir uz proizvoljno odabran sustav za upravljanje bazama podataka. Također, treba koristiti paket Laravel Sanctum za realizaciju autentifikacije s klijentskom stranom (JWT token).

Rad mora biti napisan prema Uputama za pisanje diplomskih / završnih radova koje su objavljene na mrežnim stranicama studija.

Zadatak uručen pristupniku: 21. ožujka 2022.

Mentor:

Doc. dr. sc. Marko Gulić

Predsjednik povjerenstva za završni ispit:

Prof. dr. sc. Kristijan Lenac

Izjava o samostalnoj izradi rada

Izjavljujem da sam samostalno izradio ovaj rad.

Rijeka, srpanj 2022.

_____

Luka Županović

Zahvala

Velike zahvale mentoru doc. dr. sc. Marku Guliću na svim smjernicama, pomoći i strpljenju pri izradi ovog završnog rada.

Sadržaj

Popis slika

Popis kodnih isječaka

1. Uvod

U današnjem modernom svijetu, gdje je uloga interneta nikada izraženija, neprestance raste potreba za što kvalitetnijim i efikasnijim razvojem web aplikacija kao i njihova složenost. Stoga je i odabir tehnologija za izradu istih poveći te konstantno evoluira. Neke od osnovnih značajki koje sve web aplikacije dijele jesu izvođenje upita na serveru, autentifikacija, visoka portabilnost te, naravno, mogućnost njihovom pristupu koristeći web preglednik. Kada govorimo o razvoju web aplikacija, ustaljena je podjela na dva aspekta njihove izrade: klijentski dio (engl. *frontend*) te poslužiteljski (engl. *backend*). Pod pojmom *frontend* se podrazumijeva onaj vidljivi dio web aplikacije kojime korisnik ostvaruje interakciju s aplikacijom. To uključuje različite komponente korisničkog sučelja (primjerice gumbovi, tražilice i sl.) kao i cjelokupni dizajn aplikacije. Zadaća je programera na serverskoj strani pobrinuti se da ono što korisniku nije vidljivo funkcionira na ispravan način, odnosno da je logika aplikacije valjajuća, a njezini podaci na ispravan način isporučeni. Zbog već spomenute velike potražnje za web aplikacijama kao i zbog naglaska na brzinu i efikasnost izrade posla u računalnoj industriji, programeri razvijaju radne okvire i za klijentsku i za serversku stranu koji omogućavaju da se pri izradi aplikacija ne gubi vrijeme na detalje niže razine. U ovome radu fokus će biti stavljen na klijentski dio razvoja web aplikacija, preciznije, čitatelj će biti detaljno upoznat s 2 popularna radna okvira klijentske strane: Vue-om i Reactom. Ova dva JavaScript radna okvira vrlo su čest izbor za izradu jednostraničnih aplikacija (engl. SPA - *single page application*) s čijim će konceptom također čitatelj biti upoznat. Osim same sintakse, bit će dan pregled značajki, arhitekture i mogućnosti koje React i Vue nude te navedene njihove sličnosti i razlike. Također, kako bi njihova usporedba bila što temeljitija, u svakom od ova dva radna okvira razvijena je i identična RESTful aplikacija za vođenje apartmana koja u sebi sadrži autentifikaciju kao i

temeljne CRUD[1] operacije. Poslužiteljska je strana ove aplikacije izrađena korištenjem Laravela, najpoznatijeg PHP radnog okvira te će navedeni postupak također biti opisan u završnim poglavljima rada. Jedan od paketa koje Laravel nudi, Laravel Sanctum, korišten je za autentifikaciju korisnika [10]. Sanctum je odlično i jednostavno rješenje za autentifikaciju, posebice kada su u pitanju jednostranične aplikacije. Slanje zahtjeva poslužitelju vrši se pomoću axios paketa koji je ubačen kao zavisnost u React i Vue projekte. Također, za brži postupak dizajniranja aplikacije korišteni su popularni CSS radni okviri Tailwind i Bootstrap te je u ovome radu pružan opis nekih od njihovih mnogobrojnih značajki.

[1]Kreiranje, čitanje, osvježavanje i brisanje podataka u zadanoj bazi

2. Jednostranične aplikacije

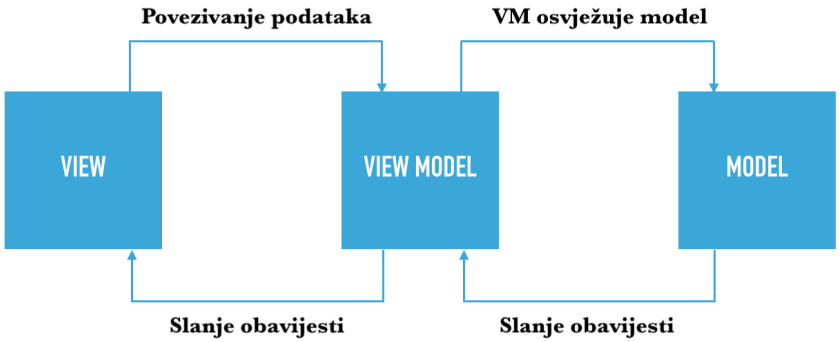
Oba radna okvira opisana u ovome radu spadaju među najpoznatije odabire kada je riječ o kreiranju jednostraničnih aplikacija. Jednostranična aplikacija ili SPA (*single page application*) je koncept kojim opisujemo web aplikacije kod kojih je web preglednik zadužen za preusmjeravanje različitih djelova aplikacije, a ne poslužitelj. Kod klasičnih web stranica, nakon što u preglednik biva unesena željena adresa, na poslužitelja se šalje zahtjev koji on potom obradi i pošalje korisniku nazad odgovarajući HTML dokument. Ukoliko korisnik primjerice stisne gumb na toj stranici ponovno se šalje potpuno novi zahtjev na server. Konstantno slanje zahtjeva nije optimalno rješenje iz razloga što može značajno usporiti web aplikaciju. Jednostranične aplikacije nude rješenje ovog problema. Svaki zahtjev za novim podacima ne uzrokuje ponovno učitavanje stranice, već samo izmjenu pojedinih djelova korisničkog sučelja. Web aplikacije kreirane na ovaj način imaju za cilj omogućiti korisnicima iskustvo kao da je u pitanju korištenje običnih *desktop* aplikacija. Nadalje, one zahtjevaju manje koda za kreiranje, imaju odlične alate za *debuggiranje* te im nije potrebna velika procesorska snaga. Neki od najpoznatijih primjera jednostraničnih aplikacije jesu Facebook, Amazon, Google Drive, Github... Zanimljivo je napomenuti kakve probleme loše performance web aplikacija mogu uzrokovati na primjerima dviju globalnih kompanija: Amazona i Googlea. Ukoliko bi Googleova tražilica generirala rezultat 0.5 sekundi sporije, njihov promet bi doživio pad od čak 20%, a jedna sekunda viška učitavanja proizvoda na stranici Amazona, godišnje bi ih koštala 1.6 milijardi dolara [12]. Međutim, valja napomenuti kako i jednostranične aplikacije imaju svoje nedostatke. Kao glavni nedostaci, mogu se izdvojiti duže trajanje inicijalnog učitavanja, ranjivost na *cross-site scripting* napade kao i kompromitirana optimizacija pretraživača iz razloga što je bitna metrika pretraživača pri rangiranju broj stranica web aplikacije. Postoje tehnike poput poslužiteljskog renderiranja koje ovaj zadnji problem ublažavaju.

2.1 Vue.js

2.1.1 Općenito o Vue.js-u

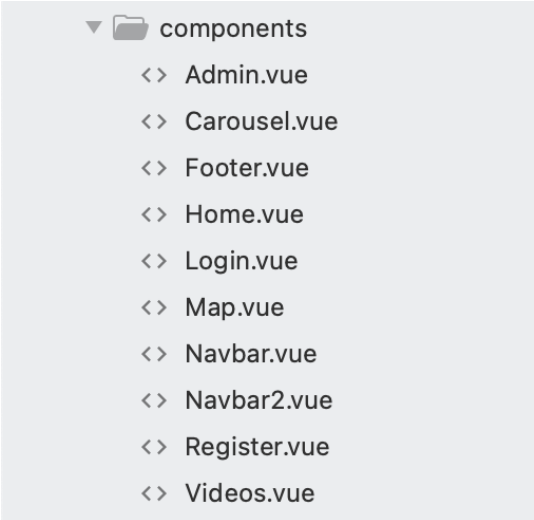
Kada je 2013. godine inženjer Googlea Evan You bio zaposlen na projektu u kojem se koristio Angular radni okvir, inspiriran njime došao je na ideju da razvije vlastiti radni okvir. Naime, značajke Angulara koje su mu se posebno svidjele jesu povezivanje podataka i činjenica da u Angularu programer nema potrebu izravno se petljati i manipulirati DOM-om (engl. *Document Object Model*)[1]. Međutim, unatoč navedenim prednostima Angular je ipak smatrao prekompleksnim za potrebe svog projekta te je, kako sam navodi, sa sobom nosio mnogobrojne koncepte zbog kojih u konačnici kod mora biti formiran na vrlo striktan način. S ciljem da stvori fleksibilniji radni okvir od Angulara kojim bi uz poznavanje HTML-a, CSS-a i Javascripta bilo relativno lako ovladati, You kreće razvijati Vue.js. Početkom 2014. godine prvi ga puta objavljuje javno na Githubu. Projektu nije dugo trebalo da naiđe na veliki interes zajednice koja ga i dalje nastavlja razvijati. Vue.js jedan je od najboljih primjera progresivnih radnih okvira [4]. On je u suštini fokusiran samo na prezentacijski sloj web aplikacije. Ne funkcionira po principu sve ili ništa, već se može koristiti i za kreiranje pojedinih elemenata korisničkog sučelja u postojećim web aplikacijama. Podržava Typescript te je u njemu i napisan. Prati *Model-Pogled-PogledModel* (engl. *Model-View-ViewModel*) arhitekturu čiji je sažet opis funkcioniranja prikazan na slici 2.1. MVVM je jedan od najčešće korištenih pristupa u programskom inženjerstvu kada je u pitanju arhitektura. *Model* predstavlja neki objekt koji sadrži podatke, *Pogled* označuje ono vidljivo korisniku, a *PogledModel* je ono što ih povezuje te je on zadužen za stvari poput, primjerice, dohvaćanja potrebnih podataka, kreiranja logike kad korisnik izvrši neku akciju i slično.

[1] višeplatformsko sučelje, neovisno o programskom jeziku koje omogućuje izmjenu elemenata dokumenta. Dokument je pritom prikazan kao stablo čiji su čvorovi njegovi elementi (veze između objekata nisu specificirane)



Slika 2.1 *MVVM arhitektura*

Vue.js predstavlja *PogledModel* dio ove arhitekture, odnosno zbog dvostrukog povezivanja podataka spona je između *Modela* i *Pogleda*. Velika prednost Vue.js-a jest njegova čitljivost. Kod je razložen u mnogo djelova: svaki dio čini jednu komponentu. Te komponente su u pravilu djelovi korisničkog sučelja. Ovakav način strukturiranja koda i njegovo razbijanje na male djelove jest arhitektura bazirana na komponentama (engl. *Component Based Architecture*). Tri glavne prednosti CBA-a su mogućnost ponovnog korištenja komponenti, dobra čitljivost te lakše održavanje koda i testiranje. Slika 2.2 prikazuje kreirane komponente u Vue aplikaciji za vođenje apartmana.



Slika 2.2 *Primjer korištenja komponenti u Vue.js-u u izrađenoj aplikaciji za apartmane*

2.1.2 Sintaktičke osnove Vue.js-a

Početna točka svake Vue.js aplikacije jest Vue instanca čiji je primjer dan u isječku koda 2.1. Ona može primati različite parametre i metode, a njezina je uloga da kontrolira cjelokupnu aplikaciju ili samo njezine pojedine djelove. Parametar *el* može biti CSS selektor kao u kodnom isječku 2.1 ili HTML element(npr. *document.getElementById('app')*)

```
import Router from './routes.js'

new Vue({
  el: '#app',
  render: h => h(App),
  router: Router,
  data: {
        vueString: 'Pisanje zavrsnog rada',
        link: 'http://www.riteh.uniri.hr'
    }
})
```

Kodni isječak 2.1 *Kreiranje Vue instance*

Nakon kreiranja Vue instance, slijedi primjer njezinog korištenja unutar komponente (isječak koda 2.2). Navedeni isječak demonstrira korištenje nekih od najbitnijih sintaktičkih elemenata, a u nastavku je detaljnije objašnjen.

```
<template>
<div class="app">
<h2> {{ vueString }} </h2>
<a v-bind:href="link"> Riteh </a>
<button v-on:click="vueString = 'Mijenjanje vrijednosti varijable'">
Click me
</button>
</div>
</template>

<script>
</script>

```

```
14 <style>
15 </style>
```

Kodni isječak 2.2 *Primjer jednostavne Vue komponente*

U 2. liniji koda kodnog isječka 2.2 * * element biva povezan s kreiranom Vue instancom. Na ovaj način omogućen je pristup svim varijablama i metodama koje su definirane u odgovarajućoj Vue instanci unutar ove div sekcije. Direktiva *v-bind* koja se nalazi u 4. liniji služi za povezivanje podataka s nekim atributom (u ovom slučaju *href*). Slijedom toga, odgovarajući link vodi na službenu stranicu Tehničkog fakulteta u Rijeci (definirano u isječku 2.1). Jedan od najbitnijih aspekata izrade web aplikacija jesu događaji (engl. *events*). Naredba koja se koristi za slušanje DOM događaja poput primjerice klikova jest *v-on* nakon koje slijedi tip događaja koji se želi promatrati. U gornjem isječku koda klikom na gumb mijenja se vrijednost *vueString* varijable. Ova vrijednost također biva promijenjena i na samom prikazu stranice (3. linija koda). Ono što je velikoj većini komponenti zajedničko jest činjenica da ih možemo podjeliti na tri djela:

- Predložak (*<template>*) - ovdje se slažu HTML elementi na željeni način
- Skripta (*<script>*) - definiranje potrebnih varijabli i metoda
- Dio za stiliziranje (*<style>*) - uređivanje komponente CSS-om

Ovakav način podjele izvrstan je za čitljivost. Za potrebe ovog primjera dostatno je korištenje predloška te bi navedeni kod identično funkcionirao i brisanjem skripte i djela za stilizirnje. Od ostalih direktiva koje nisu u ovom kodnom isječku navedene, ključno je još spomenuti *v-if* i *v-for*. Prva se koristi sama ili u kombinaciji s *v-else* i *v-else-if* ukoliko programer želi da se pojedini djelovi korisničkog sučelja prikažu u ovisnosti o nekom uvjetu, a druga za renderiranje elemenata određeni broj puta. U velikoj većini Vue.js aplikacija, preusmjeravanje se vrši pomoću *VueRoutera*, službenog preusmjerivača ovog radnog okvira. Dva načina njegovog dodavanja u projekt su uz *npm* i CDN:

- Naredba u terminalu: *npm install vue-router*

- *<script src="https://unpkg.com/vue-router@4"></script>*

Primjer korištenja *VueRoutera* u *routes.js* datoteci aplikacije izrađene u sklopu ovoga rada vidljiv je na slici 2.3.

```
import Vue from 'vue'
import VueRouter from 'vue-router'
import Login from './components/Login.vue'
import Register from './components/Register.vue'
import Home from './components/Home.vue'
import Carousel from './components/Carousel.vue'
import Videos from './components/Videos.vue'
import Maps from './components/Map.vue'
Vue.use(VueRouter)

const router = new VueRouter({
    routes: [
        {
            path: "/login",
            component: Login
        },
        {
            path: "/register",
            component: Register
        },
        {
            path: "/",
            component: Home
        },
        {
            path: "/contact",
            component: Maps
        }
    ]
})
export default router
```

Slika 2.3 *Definiranje putanja i odgovarajućih komponenti*

Nakon pridruživanja komponenti odgovarajućim putanjama, potrebno je prikaz na odgovarajućem mjestu renderirati. To se postiže ubacivanjem *<router-view>* na željeno mjesto. Korištenje *<router-link to="/imePutanje">* omogućuje renderiranje *<a> tag-a* s navedenim *href* atributom. *VueRouter* također nudi neke naprednije značajke poput ugniježdenih ruta, modularne konfiguracije itd.

2.2 React.js

2.2.1 Općenito o React.js-u

React.js je nedvojbeno jedna od najpopularnijih tehnologija kada je u pitanju izrada web aplikacija. Prema rezultatima ankete koju je 2021. godine provela poznata stranica Stackoverflow, nosi i titulu najkorištenije nakon što ga je kao svoj odabir potvrdilo 40% ispitanika te je time pretekao jQuery (s 35%) [13]. Definirati React kao radni okvir ne bi bilo u potpunosti točno jer za razliku od standardnih radnih okvira, poput primjerice Angulara ili već spomenutog Vue.js-a, ne dolazi s toliko ugrađenih opcija potrebnih za izradu velikih aplikacija. Nadalje, za razliku od Vue.js-a koji je usko povezan uz MVVM arhitekturu, React ne prati striktno niti jednu arhitekturu. On na neki način predstavlja *Pogled* dio najpoznatijih arhitektura (MVVM i MVC), međutim na *developeru* je da odabere koji mu arhitekturalni pristup najviše odgovara za potrebe projekta. Stoga ga je preciznije definirati kao Javascript knjižnicu otvorenog koda. Poput Vue.js-a baziran je na komponentama te se potiče kreiranje što većeg broja istih. Poželjno je i da su one ponovno iskoristive i da služe samo jednoj svrsi. S obzirom na sintaksu njihovog kreiranja, razlikujemo funkcijske i klasne komponente. Prve su u suštini Javascript funkcije, a druge Javascript klase. Životni ciklus komponente sastoji se od tri faze:

- *Mounting* - elementi se postavljaju na DOM
- *Updating* - faza osvježavanja uzrokovana promjenom stanja ili *propsa* u komponenti (detaljnije objašnjenje ova dva pojma pružano je u idućem poglavlju ovoga rada)
- *Unmounting* - elementi se miču iz DOM-a

React pruža deklarativan pristup razvijanju korisničkog sučelja, odnosno eliminira potrebu zadiranja u DOM. Posjeduje odlične performanse i podržava Typescript. Jedna od najbitnijih značajki ove knjižnice, koju dijeli s Vue-om jest virtualni DOM: umjesto ponovnog renderiranja čitave stranice renderiraju se samo objekti koji su se mijenjali. React koristi JavaScript XML, skraćeno JSX, koji olakšava pisanje HTML-a. Shodno tome nema stroge podjele između

JavaScripta i HTML-a. Što se tiče povijesti ove knjižnice, ona je prvi puta javno objavljena u svibnju 2013. godine. Za to je zaslužna Meta kompanija, nekoć Facebook, koja ga i dalje održava uz pomoć zajednice. Može se reći da je i samo vrijeme objavljivanja izuzetno povoljno utjecalo na popularnost Reacta. Naime, upravo u to vrijeme Angular je objavio novu verziju koja je sa sobom nosila ogromne promjene zbog čega su mnogi programeri krenuli eksperimentirati s Reactom. Uz aplikacije Mete još neki od poznatijih primjera web aplikacija u kojima je React korišten jesu Netflix, Paypal, Reddit...

2.2.2 Sintaktičke osnove React.js-a

Kao što je već spomenuto, u Reactu se sve temelji na komponentama. Ustaljena je praksa prilikom izrade aplikacija imati jednu glavnu *App* komponentu koja sadrži u sebi ostale komponente. Isječci koda 2.3 i 2.4 prikazuju primjer kreiranja jednostavne komponente na dva načina: kao klasne i kao funkcionalne.

```
class ExampleComponent extends React.Component {
  render() {
    return <h1>Klasna komponenta</h1>;
  }
}
```

Kodni isječak 2.3 *Klasna komponenta*

```
const ExampleComponent=()=>
{
    return <h1>Funkcionalna komponenta</h1>;
}
```

Kodni isječak 2.4 *Funkcionalna komponenta*

Sintaksa nalik HTML-u zapravo je JSX. JSX je ekstenzija ECMAScripta kojom se definira željeni izgled korisničkog sučelja, a ta se sintaksa potom prevodi u JavaScript objekte. Svaka HTML oznaka poput npr. *<ul>* transformira se u *React.createElement()* s odgovarajućim parametrima. Na ovaj način omogućeno je zajedničko korištenje JavaScripta i HTML-a. Varijabla se u Reactu deklarira ključnom riječju *const*, a njezinoj se vrijednosti pristupa na sljedeći

način: *{imeVarijable}*. Na primjeru isječka koda 2.5 vidljivi su i u nastavku objašnjeni neki od najznačajnijih elemenata sintakse Reacta.

```
1  const ExampleComponent = () => {
2
3    //Kreiranje funkcije
4    const handleClick = () => {
5      console.log('Clicked');
6    }
7
8    //Koristenje useState hooka
9    const [number, setNumber] = useState(1);
10
11   return (
12     <div className="example">
13       <h2>Example</h2>
14       <button onClick={handleClick}>Handle click event</button>
15       <ChildComponent passedString="Child component parameter"/>
16
17       <h1>Chosen number {number}!</h1>
18       <button onClick={() => setNumber(1)}> 1 </button>
19       <button onClick={() => setNumber(2)}> 2 </button>
20       <button onClick={() => setNumber(3)}> 3 </button>
21     </div>
22   );
23 }
```

Kodni isječak 2.5 *Osnove sintakse Reacta*

Događaji u Reactu vrlo su slični DOM događajima. Nakon kreiranja *handleClick* funkcije u gornjem isječku koda, u 14. liniji koda ista se povezuje s gumbom. Ukoliko bi ista funkcija primala parametar, *onClick=handleClick* u 14. liniji bilo bi potrebno zamijeniti s

- *onClick=() => handleClick('parametar')*

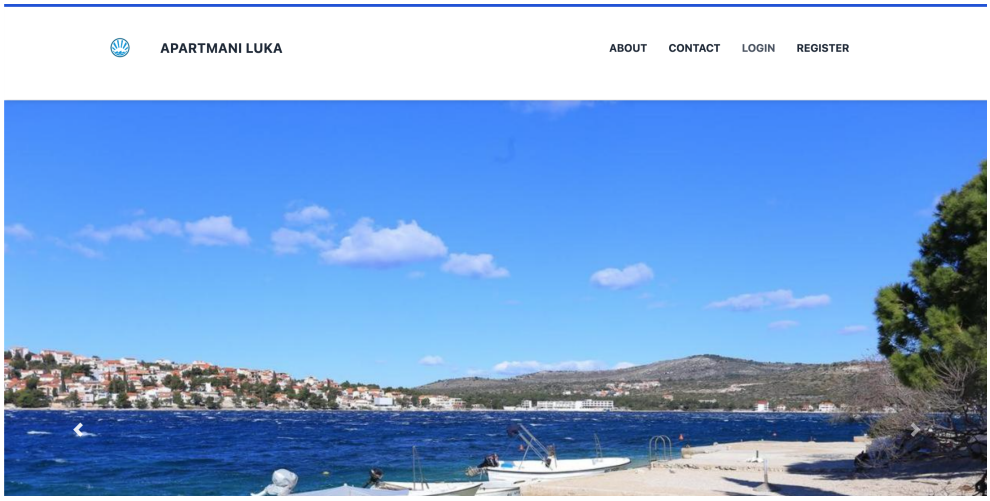
Na ovaj način, korištenjem tzv. *arrow* funkcije sprječava se poziv funkcije po učitavanju stranice. U Reactu je prosljeđivanje podataka od roditeljskih k dječjim komponentama vrlo jednos-

tavno pomoću argumenata (engl. *props*). U 15. liniji koda prikazano je korištenje argumenata među komponentama: komponenti *ChildComponent* prosljeđujemo argument *passedString* te je unutar te komponente jednostavno pristupiti njegovoj vrijednosti. Također je moguće i funkcije proslijediti kao argument. Prilikom prosljeđivanja argumenata treba imati na umu da se njihova vrijednost ne može mijenjati u dječjoj komponenti. Osim argumenata, tu je i stanje (engl. *state*). Pod pojmom stanja podrazumjeva se objekt koji predstavlja podatke korištene u komponenti u određenom vremenu. Za razliku od argumenata, vrijednosti stanja mijenjaju se unutar same komponente. *Hooks* su posebne funkcije koje omogućavaju da funkcionalne komponente imaju pristup stanju. U isječku koda 2.5 vidljiv je primjer korištenja *useState() hooka*. U 9. liniji koda definiraju se dvije vrijednosti: *number* koji sadrži trenutno stanje varijable i funkcija *setNumber* koja to stanje mijenja. Nakon pozivanja *setNumber* funkcije (linije 18.-20.), sadržaj teksta definiranog u 17. liniji koda, odnosno varijable *number* ažurira se u ovisnosti o kliknutom gumbu. Ovime se postiže reaktivnost varijable. Uz *useState* valja spomenuti i *useEffect hook*. Ona se pokreće prilikom svakog renderiranja i idealno je mjesto za postavljanje *listenera* i dohvaćanje podataka s poslužitelja. Samim kreiranjem React projekta u njega nije ugrađeno rješenje za preusmjeravanje. Ono se u aplikacijama najčešće postiže ubacivanjem *React routera*, uvjerljivo najčešće korištene zavisnosti u ovu svrhu. Pomoću njega je moguće vrlo jednostavno definirati putanju: *<Route path="/register">*. Detaljniji opis *React routera* uz primjer njegovog korištenja u izrađenoj aplikaciji za vođenje apartmana dan je u poglavlju 3.2.3 ovoga rada.

3. Izrada aplikacije

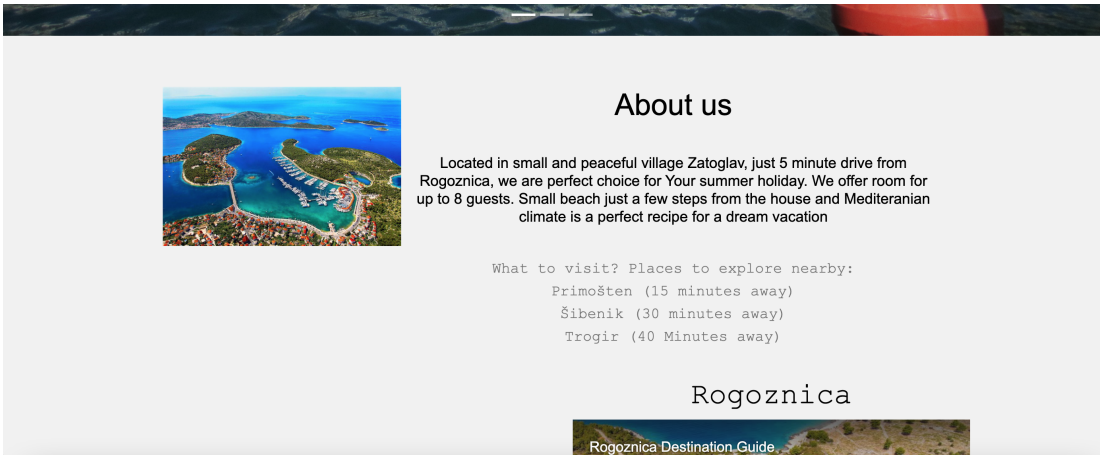
3.1 Opis aplikacije

Web aplikacija izrađena u sklopu ovog završnog rada, a čiji je opis i tijek izrade dan u nastavku rada, za svrhu ima promociju privatnih apartmana putem interneta. Osim samih informacija u svezi apartmana poput primjerice njihove lokacije i okolnih mjesta koje gosti mogu posjetiti, implementirana je i autentifikacija kao dio zadatka. Nakon uspješne prijave ili registracije korisnik ima mogućnost zatražiti rezervaciju apartmana na željeni datum ispunjavanjem ponuđenog obrasca. Korisnici kojima je dodjeljena uloga administratora potom dobiveni zahtjev mogu prihvatiti ili odbiti. Naslovna stranica vidljiva na slikama 3.1 - 3.4 sastoji se od nekoliko djelova: *navbar* koji je zajednički svim putanjama i nudi korisniku različite opcije navigiranja po stranici, *carousel* sekcija na kojoj se vrte slike samih apartmana, *about* sekcija na kojoj korisnik može pročitati informacije o mjestima u blizini te pogledati promotivne Youtube videe tih mjesta te u konačnici podnožje stranice s poveznicama na društvene mreže samih apartmana. Poslužiteljski dio aplikacije implementiran je korištenjem Laravela, dok je klijentski izrađen na dva različita načina: korištenjem Reacta [3] i Vuea [4].

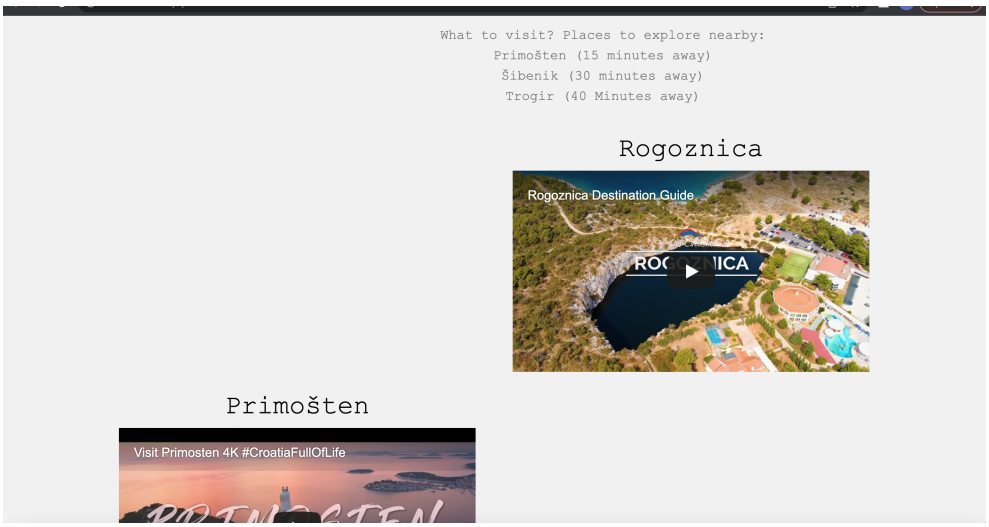


Slika 3.1 *Naslovnica izrađene aplikacije*

Područje za navigiranje ovom aplikacijom vidljivo na slici iznad gdje se korisniku nudi 4 različita gumba. *About* je jedini gumb čijim pritiskom putanja ostaje ista, međutim stranica se spušta dolje na *About* sekciju prikazanu na slikama 3.2 i 3.3. Ukoliko je korisnik prijavljen, područje za navigaciju se mijenja te gumbovi za prijavu i registraciju bivaju zamijenjeni gumbom za odjavu. Korisnik je odjavom preusmjeren na početnu stranicu.

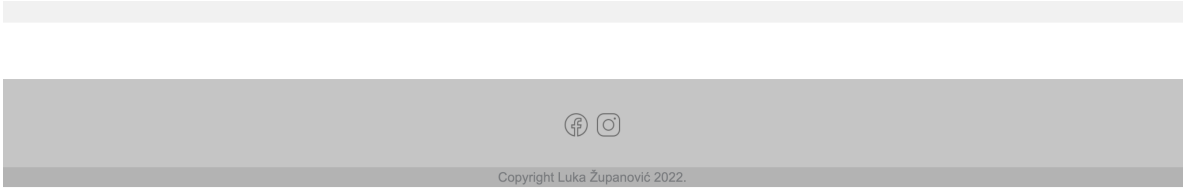


Slika 3.2 *About sekcija naslovne stranice*



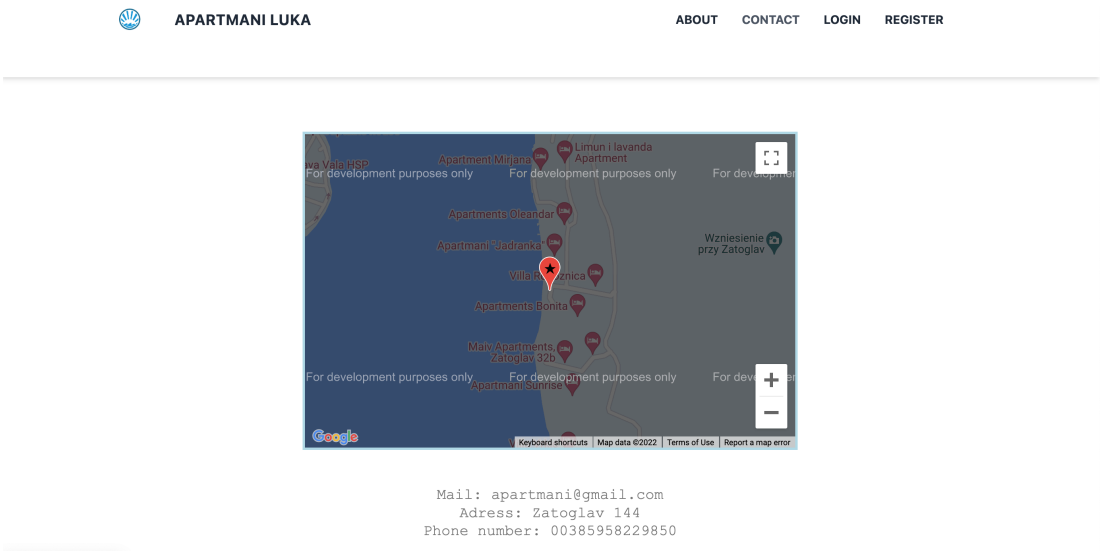
Slika 3.3 *About sekcija naslovne stranice*

Još jedan dio zajednički svim putanjama jest već spomenuto podnožje vidljivo na slici 3.4.



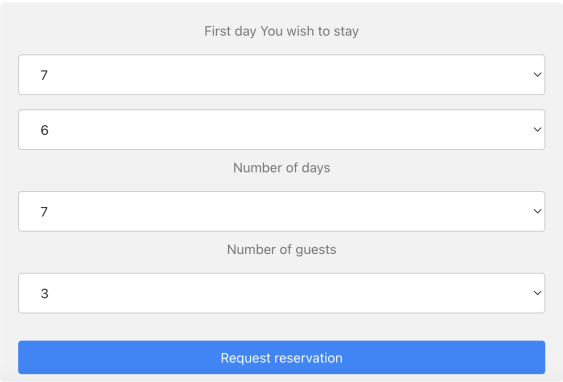
Slika 3.4 *Podnožje izrađene aplikacije*

Klikom na *Contact* korisnik prelazi na dio stranice koji nudi informacije o načinu kontaktiranja vlasnika apartmana kao i lokacijske informacije. Osim navedene adrese apartmana, korisnik može vidjeti i prikaz mape na kojoj je objekt označen. Mapu je moguće uvećati/umanjiti te prikazati na čitavom ekranu. Dio opisanog djela web aplikacije vidljiv je na slici 3.5.



Slika 3.5 *Prikaz apartmana na mapi zajedno s kontaktnim informacijama*

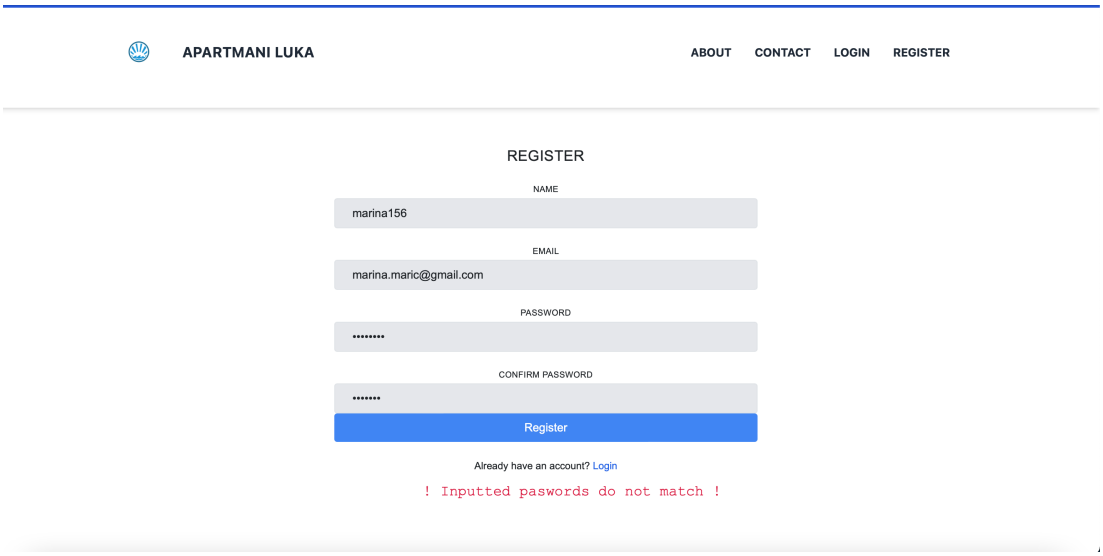
Ova se putanja također mijenja u ovisnosti o tome je li korisnik autentificiran. Ukoliko je, ispod kontakt informacija korisniku je prikazan i obrazac za rezervaciju gdje bira željeni datum (dan i mjesec) dolaska, željeno trajanje odmora i broj gostiju. Ispod obrasca ispisane su sve njegove dosada napravljene rezervacije, a njihov je status predstavljen bojom (crvena, žuta i zelena). Obrazac za rezervaciju vidljiv je na slici 3.6.



Slika 3.6 *Obrazac za podnošenje zahtjeva za rezervaciju*

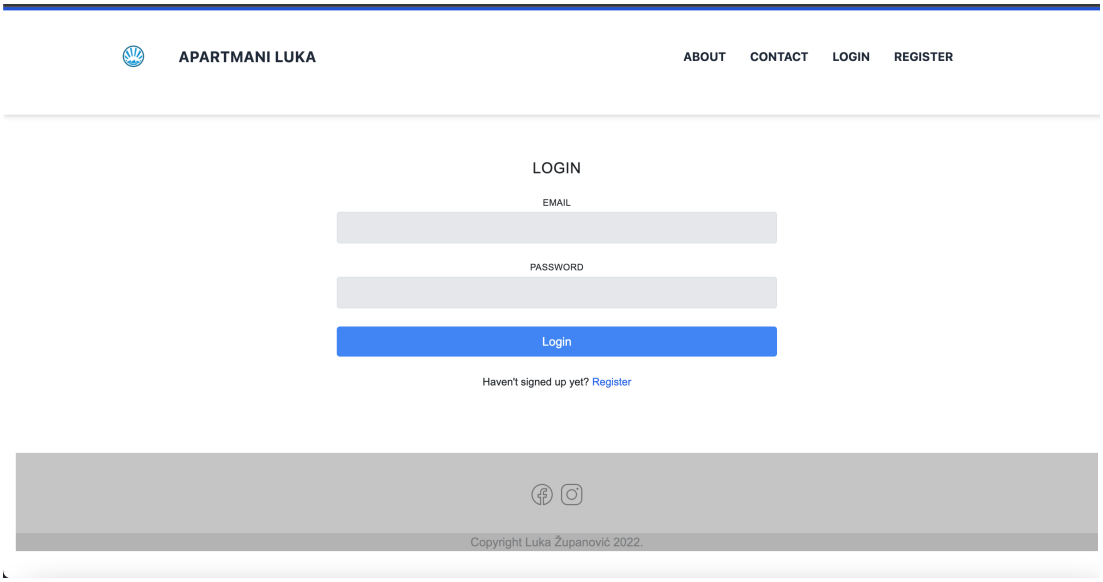
Preostale dvije korisniku vidljive putanje ove web aplikacije služe za prijavu i registraciju. Za uspješnu registraciju potrebno je unijeti korisničko ime, adresu elektroničke pošte te minimalno

6 znakova dugačku lozinku, dok prijava zahtjeva adresu e-pošte i unos zaporke. Izgled obrazaca za registraciju i prijavu nalazi se na slikama 3.7 i 3.8



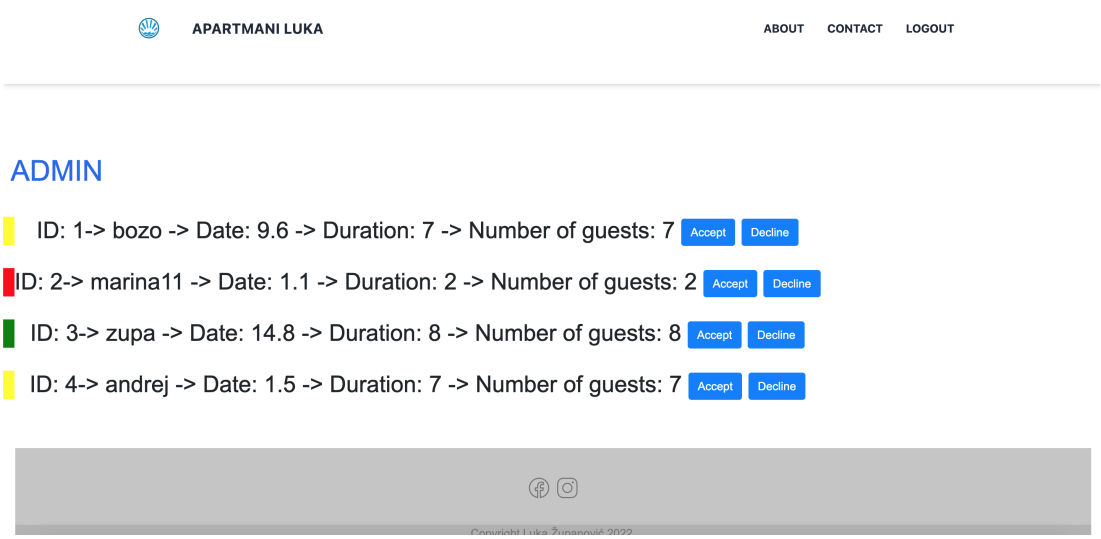
Slika 3.7 *Obrazac za registraciju: slučaj kada korisnik unese različite lozinke pri registraciji*

Ukoliko korisnik pogriješi s unosom lozinke ili ne unese adresu e-pošte prilikom registracije crvenim se slovima ispisuje za to predviđena poruka.



Slika 3.8 *Obrazac za prijavu*

Pojedini korisnici imaju posebne administratorske ovlasti. Ukoliko se korisnik kojemu je dodjeljena takva ovlast prijavi, umjesto na *Contact* putanju biva preusmjeren na administratorski dio stranice. Ovdje je moguće prihvatiti ili odbiti zahtjeve korisnika (slika 3.9).



Slika 3.9 *Putanja za administratora*

3.2 Korištene tehnologije

3.2.1 Node package manager

Izradi ovog projekta prethodila je instalacija Node.js-a. Njegovom instalacijom omogućeno je korištenje zadanog upravitelja paketa za JavaScript *runtime* okruženje Node.js (engl. *Node Package Manager*, skraćeno npm). *npm* se sastoji od tri djela: službene web stranice na kojoj programeri mogu otkrivati nove pakete, komandnog korisničkog sučelja (engl. CLI - *command line interface*) za interakciju s npmom te registra koji u sebi sadrži mnoštvo Javascript paketa pogodnih za razvoj web aplikacija [11]. Treba napomenuti kako je svakom korisniku omogućeno dodavanje paketa zbog čega je prilikom korištenja potrebno pozvati na oprez s obzirom da odabrani paket ne mora nužno biti valjajuć i da u sebi može sadržavati kojekakav zloćudni

softver. Svi paketi korišteni pri izradi nekog projekta, u daljnjem radu zavisnosti (engl. *dependencies*), bivaju zapisani u njegovu *package.json* datoteku. Unosom sljedeće naredbe u terminal paket biva dodan u *dependencies* grupu *package.json* datoteke:

- *npm install <imePaketa> - -save*

Također je moguće specificirati željenu verziju paketa dodavanjem *@brojVerzije* nakon imena paketa.

3.2.2 Vue.js

Nakon uspješne instalacije *npm*-a sljedeći korak bio je instalacija Vue CLI paketa. Ovo se postiže već spomenutom naredbom

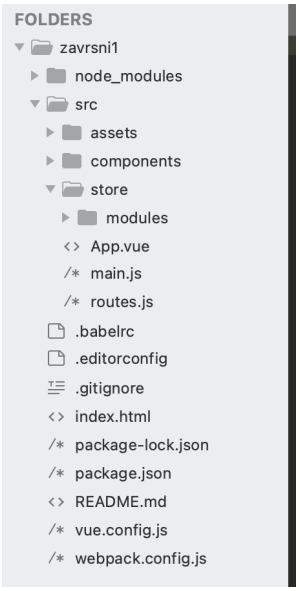
- *sudo npm install -g vue-cli*

Zastavica -g označuje da je paket instaliran globalno te mu je omogućen pristup u terminalu. Naredba:

- *vue init webpack-simple imeProjekta*

instalira *webpack*. Ovu je naredbu također moguće koristiti bez *-simple* nastavka. Korištenjem *-simple* nastavka nisu uključene neke naprednije značajke kao niti testiranje. *Webpack* je statički paket modula. Funkcionira na način da prolazi kroz korištene pakete i kreira graf ovisnosti te, u ovisnosti o tom grafu, kreira novi paket koji sadrži *bundle.js* datoteku. Ta je datoteka na jednostavan način pristupačna HTML datoteci i korištena u aplikaciji. Verzija *Vue.js-a* korištena u projektu jest 2.6.14. Nju je lako moguće provjeriti navigiranjem u željeni direktorij i naredbom:

- *npm list vue*



Slika 3.10 *Konačna struktura Vue projekta*

Na slici 3.10 prikazana je struktura konačne aplikacije za upravljanje apartmanima. U mapi *node_ modules* pohranjene su zavisnosti projekta. Datoteka *App.vue* sadrži glavnu komponentu aplikacije, dok se ostale nalaze u *components* direktoriju. Unutar *main.js* datoteke inicijalizirana je Vue instanca te su uvezeni korišteni paketi, a unutar *routes.js-a* definirane su sve putanje aplikacije. U nastavku ovog poglavlja bit će dan opis nekih korištenih zavisnosti i implementacije nekih od bitnijih djelova ove web aplikacije. Isječak koda 3.1 prikazuje kreiranje *Carousel* komponente.

```
<template>
  <div>
    <b-carousel
      id="carousel-1"
      v-model="slide"
      :interval="4000"
      style="text-shadow: 1px 1px 2px #333;"
      @sliding-start="startSlide"
      @sliding-end="endSlide">

      <b-carousel-slide class="picture"
        img-src="https://www.mojsmjestaj.hr/foto/apartment/6639/IMG_6031.JPG
```

```
    "
      ></b-carousel-slide>

      <b-carousel-slide class="picture"
      img-src="https://i.croatiaimages.com/private-accommodation/12212/k
    -12212/zatoglav-house-bedroom-2-1-l.jpg?v=dbb591c05d">
      </b-carousel-slide>

      <b-carousel-slide class="picture"
      img-src="https://cf.bstatic.com/xdata/images/hotel/max1280x900
    /189679138.jpg?k=372326
    f8267602ee36396523bbe75b3f357365c2b9a5f10a2884afbbbc350feb&o=&hp=1">
      </b-carousel-slide>
    </b-carousel>
  </div>
</template>

<script>
  export default {
    data() {
      return {
        slide: 0,
        sliding: null
      }
    },
    methods: {
      startSlide(slide) {
        this.sliding = true
      },
      endSlide(slide) {
        this.sliding = false
      }
    }
  }
</script>

<style>
.picture {
```

```
    height: 240px;
}
</style>
```

Kodni isječak 3.1 *Kreiranje Carousel komponente u Vue.js-u*

Valja napomenuti kako je u ovoj komponenti korišten Bootstrap [7] koji značajno olakšava posao kreiranja pokretne galerije i smanjuje potreban kod. Rezultat ovog koda već je prikazan na slici 3.1 (samo slike, bez navigacijskog područja koje je zasebna komponenta). Neke od korištenih zavisnosti u ovom projektu su *vue-lazytube* za umetanje Youtube videa i *vue2-google-maps* za umetanje mape. Prije umetanja mape u bilo koji projekt potrebno je iz sigurnosnih razloga s Google računom generirati API ključ kako bi bilo omogućeno korištenje Maps Javascript aplikacijskog programskog sučelja. Nakon što je isti generiran idući korak bio je dodavanje sljedećih linija koda u *main.js* datoteku.

```
import * as VueGoogleMaps from 'vue2-google-maps'

Vue.use(VueGoogleMaps, {
  load: {
    key: 'AIzaSyC2Ff04OPVm93sFb_i-g3yTmcyzZyl9pbw',
    libraries: 'places',
  }
});
```

Kodni isječak 3.2 *main.js datoteka - konfiguracija Google karte*

API ključ zapisan je u *key* parametar te je omogućeno ispravno korištenje *GmapMap* i *GmapMarker* (za označavanje mjesta na mapi) komponenti (kodni isječak 3.3). Isti API ključ iskorišten je i u React projektu.

```
<GmapMap class="center"
      :center='center'
      :zoom='16'
      style='width:45%;  height: 400px; text-align: center'
    >
   <GmapMarker
       v-if="this.center"
       label="*"
```

```
        :position="{
          lat: 43.53231,
          lng: 15.98122,
        }"
    />
</GmapMap>

```

Kodni isječak 3.3 *Korištenje VueGoogleMaps komponenti unutar izrađene komponente*

Hijerarhija Vue aplikacije započinje od glavne *App* komponente. Donji isječak koda (3.4) prikazuje početni dio *App.vue* datoteke.

```
<template>
<div>
<navbar v-if="this.$userName =='Guest'"></navbar>
<navbar2 v-else :tokenProp="this.$userToken"> </navbar2>
<router-view :tokenProp="this.$userToken" :userNameProp="this.$userName">
</router-view>
<footerr></footerr>
</div>
</template>
```

Kodni isječak 3.4 *Glavna komponenta aplikacije*

Svakoj komponenti kao argument (engl. *prop*) pridružujemo dvije varijable: u prvoj je spremljeno korisničko ime, a u drugoj njegov token za autentifikaciju. Ostale komponente na ovaj način tim vrijednostima mogu pristupiti te ih po potrebi mijenjati. 3. linija u gornjem isječku koda primjer je postizanja uvjetovanog renderiranja u Vue.js-u.

3.2.3 React.js

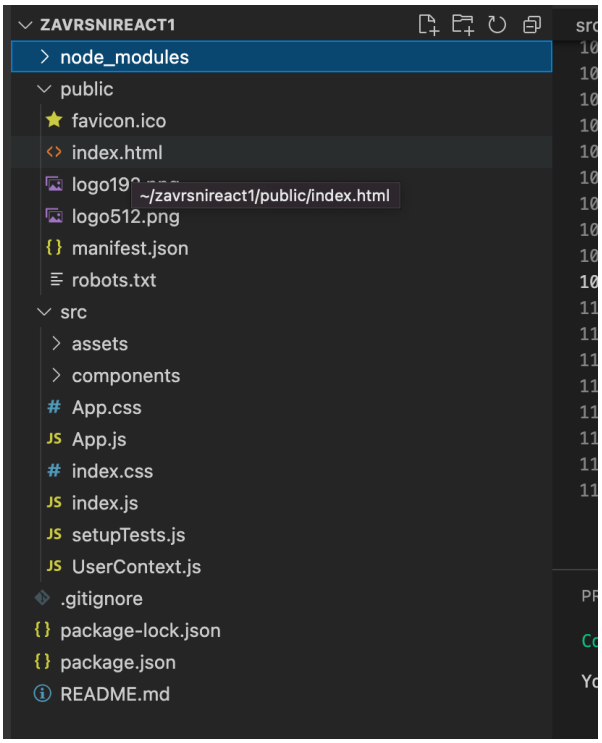
Izrada iste aplikacije u Reactu započinje kreiranjem novog React projekta što se postiže sljedećom naredbom u terminalu:

- *npx create-react-app zavrsniReact1.*

Najčešće korištena naredba prilikom izrade ove aplikacije bez dvojbe je bila

- *npm start.*

kojom se pokreće razvojni server te je time moguće vidjeti u web pregledniku trenutno stanje projekta. Na slici 3.11 prikazane su sadržane mape i datoteke u React projektu.



Slika 3.11 *Konačna struktura React projekta*

U direktoriju *node_ modules* sadržane su sve zavisnosti projekta. Idući direktorij *public* sadrži sve statičke djelove. Najbitnija datoteka za spomenuti unutar ovog direktorija jest *index.html*. Ovdje se nalazi referenca na sav napisani kod Reacta. Ova datoteka na neki način predstavlja sponu između web preglednika i Reacta. Unutar *src* direktorija ide najveći dio koda za kreiranje aplikacije. Direktoriji *assets* i *components* naknadno su kreirani radi lakšeg snalaženja po projektu te se u njima nalaze potrebne slike i kreirane komponente. Od datoteka unutar *src-a* generiranih pri pokretanju projekta bitno je izdvojiti iduće:

- *index.js* - početna točka aplikacije, uzima komponente i postavlja ih na DOM

- *App.js* - glavna komponenta koja sadrži sve ostale komponente
- *App.css* - datoteka za stiliziranje *App* komponente
- *index.css* - datoteka za stiliziranje aplikacije

Datoteka *package.json*, isto kao i u Vueu sadrži informacije o projektu poput imena, verzije, popisa zavisnosti i sl. *App* komponenta od koje sve započinje vidljiva je na slici 3.12.

Slika 3.12 *Kod unutar App.js komponente*

Prvo što se da primijetiti jest da je ona funkcionalna komponenta. Korištenjem *useState hooka* omogućeno je praćenje promjena u stanju *user* varijable te ponovno renderiranje pojedinih djelova aplikacije po potrebi. Naime, nakon uspješne autentifikacije korisnik biva preusmjeren

na *Contact* putanju, a njegovo se korisničko ime i JWT token spremaju u *localstorage* (to se postiže na sljedeći način:
localStorage.setItem("name", response.data.user.name);
localStorage.setItem("token",response.data.token);).
Stoga je odgovarajućoj *Map* komponenti prosljeđena *setUser* funkcija kao argument (*props*), a unutar same komponente njezina se vrijednost postavlja u novo stanje *localstoragea*. Ta je funkcija također prosljeđena *Navbar2* komponenti koja je zadužena za odjavu korisnika. Na istoj slici je također vidljivo korištenje *React Routera* u aplikaciji. *<Switch>* renderira prvu komponentu koja odgovara navedenoj putanji. Dodavanjem ključne riječi *exact* renderiranje se vrši samo ako je putanja identična. U protivnom bi se uzela prva putanja koja sadrži uneseni uzorak te bi se u ovom slučaju uvijek renderirala naslovna putanja. Kao i kod izrade projekta u Vue.js-u, za potrebe ubacivanje Youtube videa i Google karte korištene su zavisnosti (*React Google Maps* i *React Youtube*). Primjer JSX sintakse koji rezultira prikazom mape vidljiv je u sljedećem isječku koda.

```
<div style={{display:'flex',
   alignItems: 'center',
   justifyContent: 'center',height: '70vh',textAlign: 'center'}}>
     <GoogleMap class="w3-center"  style={{ width:390}}
       mapContainerStyle={containerStyle}
       center={center}
       zoom={17}
       onLoad={onLoad}
       onUnmount={onUnmount}
     >
       {}
       <Marker
         position={position}
         label={"*"}
       />
     </GoogleMap>
</div>
```

Kodni isječak 3.5 *Ubacivanje Google karte u odgovarajuću React komponentu*

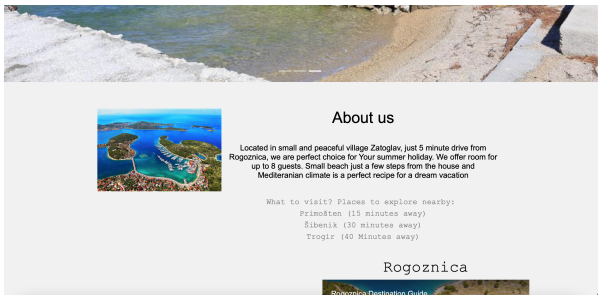
3.2.4 Stiliziranje stranice

Za potrebe stiliziranja izrađene aplikacije, uz CSS, korišteni su i Bootstrap [6] i Tailwind [8]. Ovi alati značajno ubrzavaju i olakšavaju dizajniranje web aplikacija. Jedan od primjera korištenja CSS-a unutar aplikacije izdvojen je na slici 3.13.

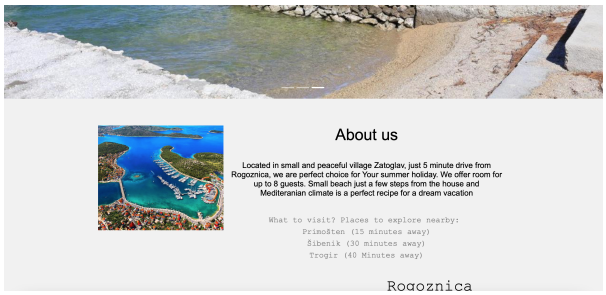
```css
.imageToHover {
    width: 300px;
    height: 300px;
    overflow: hidden;
    margin: 0 auto;
}

.imageToHover img {
    width: 100%;
    transition: 0.5s all ease-in-out;
}

.imageToHover:hover img {
    transform: scale(1.5);
}
```

Slika 3.13 *CSS kod za positzanje efekta nakon prelaska mišem*

Primjena ovog koda na odgovarajući element rezultira efektom kao na slikama 3.14.



(a) *Pozicija miša izvan područja slike*



(b) *Pozicija miša na području slike*

Slika 3.14 *Promjena okvira slike prelaskom miša*

Unutar *index.html* dokumenta oba projekta također je uključen i W3.CSS. W3 je jedna od najpoznatijih internet platformi za učenje razvoja web aplikacija, a ovaj njihov predložak također ubrzava dizajn web aplikacija. U projektu je korišten na nekim mjestima za jednostavnije kreiranje sekcija. Tako je za kreiranje centriranog elementa dovoljno dodijeliti mu klasu *w3-center*,

za kreiranje sekcije širine jedne trećine stranice klasu *w3-third* i sl.

3.2.4.1 Bootstrap

Vue i React oba imaju svoju implementaciju popularne Bootstrap knjižnice. React-Bootstrap i BootstrapVue dijele mnogo toga te funkcioniraju na vrlo sličan način. Opcije koje ova knjižnica nudi koje olakšavaju posao *developerima* uistinu su brojne te će u ovom poglavlju biti istaknute neke od njih korištene u ovome projektu. Ukoliko je slici dodijeljen argument *fluid*, ona postaje responzivna tj. prilagođava se veličini ekrana. Kreiranje pokretne galerije uz pomoć BootstrapVuea već je opisano u poglavlju 3.2.2, a u React-Bootstrapu je također jednostavno. Unutar *<Carousel>* elementa redaju se željeni *<Carousel.Item>* elementi. Dok kreiranje prilagođenih gumbova u CSS-u zahtijeva dosta pisanja koda, Bootstrap sa sobom nudi nekoliko predefiniranih stilova responzivnih gumbova te je za njihovo kreiranje dovoljna svega jedna linija koda.

- React-Bootstrap - *<Button variant="primary">Request reservation</Button>*
- BootstrapVue - *<b-button variant="primary">Request reservation</b-button>*

3.2.4.2 Tailwind

Tailwind je brz i fleksibilan radni okvir utemeljen na CSS-u koji također uvelike pomaže pri dizajniranju web aplikacija [8]. Najjednostavniji i odabrani način njegova dodavanja u projekt jest dodavanjem sljedeće linije unutar *<head>* oznake *index.html-a* datoteke:

- *<script src="https://cdn.tailwindcss.com"></script>*

Tailwind nudi mnoštvo predizajniranih ključnih riječi koje se primijenjuju na elemente. Osim bržeg upravljanja bojama teksta i bojama elemenata, u nastavku je dan pregled nekih od ostalih Tailwind klasa pridodijeljenih različitim elementima u aplikaciji za vođenje apartmana te je objašnjeno što je time postignuto.

- *hover:text-gray-600* - efekt promjene boje teksta iz crne u sivu prelaskom mišem preko gumba na području za navigiranje
- *tracking-wide* - dodijeljeno *input* oznaci za lozinku sa svrhom većeg razmaka
- *rounded* - elementu se zaokružuje rub
- *items* - pozicioniranje elemenata unutar sekcije na željeni način (npr. items-center)

Mana Tailwinda jest da kod vrlo lako može postati prenatrpan različitim ključnim riječima što ga čini težim za čitanje. Ovaj je problem posebno naglašen u većim aplikacijama.

3.2.5 Poslužiteljski dio

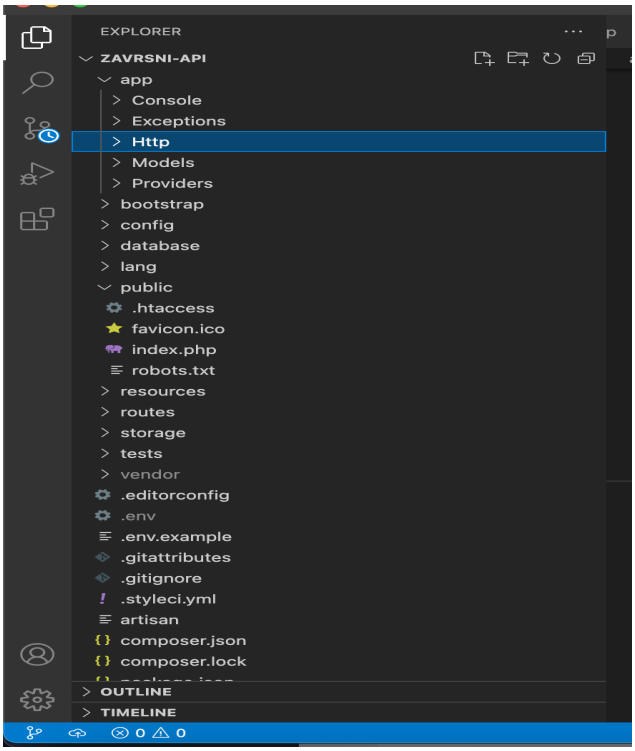
Za razvoj poslužiteljske strane ove web aplikacije korišten je Laravel radni okvir [5], dok je odabrana baza podataka za ovaj projekt MySQL. Prije samog opisa načina implementacije poslužiteljskog djela aplikacije valja objasniti koncept REST aplikacijskog programskog sučelja. REST (engl *Representational State Transfer*) je najčešće korišten arhitekturalni pristup pri dizajniranju mrežnih aplikacija. Baziran je na komunikaciji između klijenta i poslužitelja koristeći protokol bez stanja (najčešće HTTP). Drugim riječima poslužitelju nije dozvoljeno spremanje podataka specifičnog zahtjeva klijenta. Još jedna od glavnih značajki REST arhitekture jest privremena pohrana memorije (engl. *caching*). Ona omogućuje bolje performanse na klijentskoj i veću skalabilnost na poslužiteljskoj strani. U aplikaciji za vođenje apartmana, kao i u ostalim RESTful aplikacijama, koriste se HTTP zahtjevi za vršenje osnovnih CRUD operacija.

3.2.5.1 Laravel

Kao što je već spomenuto, Laravel je jedan od najpoznatijih radnih okvira kada je u pitanju poslužiteljska strana. Baziran je na PHP-u, otvorenog je koda i elegantne sintakse [5]. Spada među progresivne radne okvire. Izrada REST API-ja za potrebe projekta započinje kreiranjem novog Laravel projekta:

- *composer create-project laravel/laravel zavrsniApi*

Struktura projekta vidljiva je na slici 3.15.



Slika 3.15 *Struktura Laravel projekta*

Unutar *app* direktorija nalaze se *Models* i *Controllers* mapa. Laravel prati MVC arhitekturu stoga je dobra praksa kreirati potrebne upravitelje od kojih svaki sadrži logiku za svoju domenu. Ovo se postiže naredbom

- *php artisan make:controller*

Dio koda kreiranog *MessageControllera* vidljiv je na slici 3.16. On u sebi sadrži logiku za CRUD operacije za rezervacije. Polja koja sadrži model rezervacije također su vidljiva na spomenutoj slici (*dateDay*,*dateMonth* itd.).

```php
class MessageController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        //
        return Message::all();
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        //
        $request->validate([
            'dateDay' => 'required',
            'dateMonth' => 'required',
            'durationStay' => 'required',
            'nrOfGuests' => 'required',
            'userName' => 'required',
            'state' => 'required',//0 pending 1 accepted 2 denied
        ]);
        return Message::create($request->all());
    }
```

Slika 3.16 *MessageController*

Nadalje, *database* mapa između ostalog sadrži migracije. Pomoću njih se kreiraju tablice u bazi podataka te na neki način predstavljaju kontrolu verzije baze podataka. Naredba za kreiranje migracije je

- *php artisan make:migration create_ messages_ table*

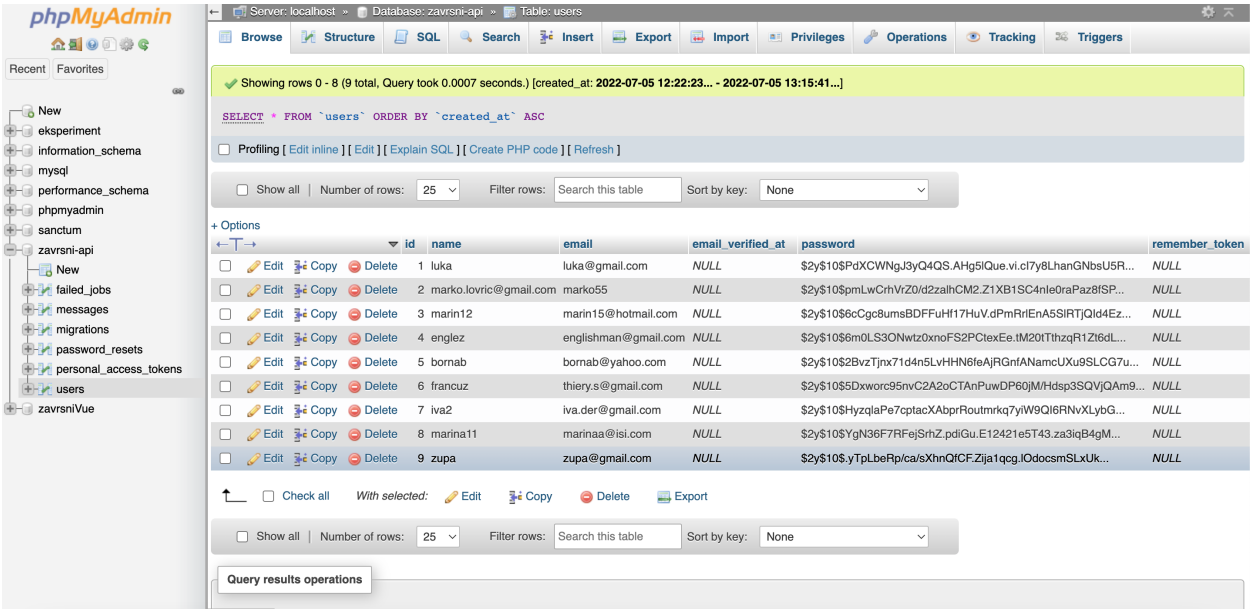
Još jedna mapa koju je potrebno istaknuti svakako je *routes*. Ovdje se definiraju sve putanje. One mogu biti dostupne svima ili zaštićene. Putanje kreirane za aplikacijsko programsko sučelje izrađene aplikacije definirane su u *api.php* datoteci. Putanje kreirane u ovoj datoteci su bez stanja. Za kraj vrijedi istaknuti i .env datoteku koja sadrži razne konfiguracijske vrijednosti te je mjesto u kojem se definira korištena baza podataka.

3.2.5.2 Laravel Sanctum

Ovaj paket nudi jednostavno i elegantno rješenje za implementaciju REST API autentifikacije. Jednostavan je za namještanje i veoma brz. Odličan je odabir za jednostranične aplikacije koje komuniciraju s Laravelovim aplikacijskim programskim sučeljem. Sanctum se instalira pokretanjem naredbe

- *composer require laravel/sanctum*

Sanctum generira korisnicima prilikom autentifikacije jedinstvene API tokene bez komplikacije OAutha. Navedeni tokeni imaju veoma dugo vrijeme trajanja (godinama), ali se mogu i ručno ukloniti od strane korisnika. Kod kreiranja zahtjeva token se pridružuje njegovom autorizacijskom zaglavlju. Prilikom namještanja Sanctum paketa potrebno je pokrenuti migraciju. Ovime se kreira *personal_tokens* tablica u bazi podataka. Za svrhu autentifikacije jednostranične aplikacije također je potrebno ručno unijeti Sanctumov *middleware* unutar *Kernel.php* datoteke [10]. Prikaz kreiranih korisnika u phpMyAdmin web alatu vidljiv je na slici 3.17.



Slika 3.17 *phpMyAdmin: kreirani korisnici*

3.2.5.3 Paket axios

Ovaj npm paket ubačen je u Vue i React projekte kao zavisnost. Paket axios je odličan za slanje asinkronih HTTP zahtjeva prema REST API-ju [9]. Kao njegove glavne prednosti mogu se izdvojiti automatsko pretvaranje podataka u JSON i jednostavno rukovanje greškama. Na klijentskoj strani koristi *XMLHttpRequests*. Primjeri slanja zahtjeva prema poslužitelju s axiosom u React i Vue projektima nalaze se na idućim isječcima koda.

```
handleLogin() {
    axios.get('http://127.0.0.1:8000/api/sanctum/csrf-cookie').then(
  response => {
        axios.post("http://127.0.0.1:8000/api/login", this.formData,
        {
          headers: {
            'Content-Type':'application/json',
            'Authorization':'Bearer {$token}'
          }
        })
        .then(response => {
          console.log("Uspjesna prijava")
           if(response.data.user.name == "luka") {
              this.$router.push({name: 'admin', params: { tokenProp:
  response.data.token, userNameProp: response.data.user.name },props: true
  });
           } else {
           this.$router.push({name: 'contactPage', params: { tokenProp:
  response.data.token, userNameProp: response.data.user.name }, props: true
  });
         }
           this.$userName = response.data.user.name
           this.$userToken = response.data.token
           this.$parent.changeNavbar(this.$userName, this.$userToken);
        })
        .catch(error => {
          console.log("Neuspjesna prijava")
          this.passwordSuccessString = '! User credentials not found !'
        })
```

```
25        }).then(response=>{
26
27        })
28        .catch(error=>{
29          console.log("Neuspjesna prijava")
30          this.passwordSuccessString = '! User credentials not found !'
31          console.log(error.response);
32        });
33  }
```

Kodni isječak 3.6 *Vue: korištenje axiosa u funkciji za prijavu korisnika*

U isječku koda 3.6 prema poslužitelju je poslan POST zahtjev. Prilikom autentifikacije jednostranična aplikacija najprije mora napraviti zahtjev */sanctum/csrf-cookie* krajnjoj točki. Ovime se inicijalizira CSRF (engl. *Cross Site Request Forgery*) zaštita. Ona sprječava neželjene napade koji za cilj imaju vršenje određenih akcija bez dozvole od strane neutentificiranog korisnika. Zaštita od ovakvih napada od iznimne je važnosti jer u suprotnom napadač ima mogućnost jednim klikom preuzeti potpunu kontrolu računa žrtve u čije su ime zahtjevi kreirani. Prilikom inicijalizacije CSRF zaštite Laravel postavlja XSRF-TOKEN kolačić na vrijednost trenutnog CSRF tokena. Ta je vrijednost zatim proslijeđena X-XSRF-TOKEN zaglavlju sljedećeg zahtjeva [10]. U isječku React koda 3.7 također se šalje POST zahtjev, ali s ciljem kreiranja rezervacije.

```
1  const makeRequest = () => {
2      const token = localStorage.getItem("token");
3      axios.defaults.headers.common['Authorization'] = `Bearer ${token}`;
4      console.log(axios.defaults.headers.common['Authorization'])
5
6      axios.get('http://127.0.0.1:8000/api/sanctum/csrf-cookie').then(response
       => {
7      axios.post("http://127.0.0.1:8000/api/messages", {
8        dateDay: dateDay,
9        dateMonth: dateMonth,
10       durationStay: durationStay,
11       nrOfGuests: nrOfGuests,
12       userName: localStorage.getItem("name"),
13       state: 0
```

```
14      },{
15            headers: {
16              'Content-Type':'application/json'
17            }
18          })
19      .then(response => {
20        console.log("Rezervacija uspjeh")
21          }).catch(error => {
22                console.log(error.response);
23              })
24          }).then(response=>{
25
26      }).catch(error=>{
27         console.log(error.response);
28      });
29    }
30  }
```

Kodni isječak 3.7 *React: korištenje axiosa u funkciji za kreiranje rezervacije*

4. Usporedba Vue.js-a i React.js-a

Iako su ova dva radna okvira po mnogočemu slična, kada je riječ o odabiru između njih svaki pruža neke svoje specifičnosti. Za početak, u nastavku je dan pregled značajki koje su zajedničke Vueu i Reactu, neke od kojih su već spomenute u prijašnjim poglavljima ovoga rada.

- Primarno korišteni za izradu jednostraničnih aplikacija, ali omogućuju i izradu višestraničnih
- Oba su bazirana na komponentama
- Virtualni DOM
- *Data down, events up* - koncept koji podrazumijeva model komunikacije između komponenti gdje dječje komponente primaju podatke od roditelja, a roditeljske osvježavaju svoje stanje nakon što djeca emitiraju određene akcije
- Fleksibilnost - oba nude veliku slobodu kada su u pitanju stvari poput preusmjeravanja, testiranja, globalnog upravljanja stanjem i sl.
- Brzi i male veličine
- Podržan TypeScript
- Oba su otvorenog koda i imaju mnoštvo knjižnica i alata na raspolaganju

Jedna od najvećih razlika između Vue.js-a i React.js-a jest sintaksa: Vue koristi HTML, a nudi i opciju JSX-a koji je u Reactu jedini odabir. Primjer sintakse JSX-a prikazan je na slici 4.1.

```jsx
import fb from "../assets/fb2.png";
import insta from "../assets/in2.png";

const pStyle = {
    color: 'gray',
  };

   const images = {
    alignSelf: 'center',
   };

const Footer = () => {

    return (

                <footer class="w3-container w3-padding-64 w3-center w3-opacity" >
  <div class="w3-xlarge w3-padding-32 w3-gray">
      <div style={images}>

    <img src={fb} style={{display : 'inline-block'}}/>

    <img src={insta} style={{display : 'inline-block'}} />

</div>

 </div>
 <p style={pStyle}>Copyright Luka Županović 2022.</p>

</footer>

    );
  }

  export default Footer;
```

Slika 4.1 *JSX na primjeru Footer komponente*

Navedeni kod rezultira kreiranjem podnožja, najjednostavnije komponente u izrađenoj aplikaciji. Ista je komponenta u Vueu kreirana na način prikazan na slici 4.2.

```
<template>
<footer class="w3-container w3-padding-64 w3-center w3-opacity" >
  <div class="w3-xlarge w3-padding-32 w3-gray">
      <div class="images" >

    <img fluid src="../assets/fb2.png" class="fblogo img" >

    <img fluid src="../assets/in2.png" class="fblogo">

</div>

 </div>
 <p style="background-color: grey;">Copyright Luka Županović 2022.</a></p>

</footer>
</template>

<script>

</script>

<style>
.column {
  float: left;
  width: 33.33%;
  padding: 5px;
}

.fblogo {
    display: inline-block;
    margin-left: auto;
    margin-right: auto;
    height: 30px;
}
#images{
   text-align:center;
}
```

Slika 4.2 *Footer komponenta u Vue.js-u*

Mnogi se slažu da je Vue.js bolji izbor za početnike upravo zbog krajnje jednostavne sintakse čijim je osnovama lako brzo ovladati. Odvajanje HTML-a, CSS-a i Javascripta čini ga intuitivnim za sve one koji su upoznati s osnovama ovih tehnologija. Nadalje, dokumentacija Vuea je vrlo temeljita, po mnogima najbolja kada su u pitanju JavaScript radni okviri. To su i neki od razloga zbog kojih mnogi smatraju da je Vue bolji izbor od Reacta kada su u pitanju manji projekti. U Reactu je povezivanje podataka jednostruko, dok je u Vue.js-u dvostruko. Kod jednostrukog povezivanja podataka ispunjen je jedan od sljedećih uvjeta:

- Svaka promjena podataka komponente bude reflektirana u korisničkom sučelju
- Svaka promjena korisničkog sučelja reflektirana je na podatke komponente

Prvi uvjet zapravo znači da će renderiranjem sljedeće linije

- * {imeVarijable} *

promjena vrijednosti zadane varijable utjecati na promjenu korisničkog sučelja. Drugi naveden uvjet pak nije izravno ostvariv u Reactu, već je potrebno rukovanje događajima kao što je to slučaj u kodnom isječku 4.1.

```
const Login = () => {
 const [password, setPassword] = useState('');

 //...
 <div class="form-group mb-4 child" >
                 <label for="password" class="block font-normal uppercase
   tracking-wide text-xs mb-1">Password</label>
                 <input type="password" onChange={(e) => setPassword(e.
   target.value)} value={password} id="password" name="password" class="
   border px-4 py-2 w-full rounded bg-gray-200" style={{ width:500}}/>
 </div>
 //...
}
```

Kodni isječak 4.1 *Unos lozinke u Reactu*

Uz *onChange* događaj se postiže da se promijeni vrijednost *password* varijable kada korisnik unese lozinku u odgovarajuće polje. U Vueu je dvostruko povezivanje omogućeno direktivom *v-model*. Ekvivalent gornje prikazanog isječka koda u Vueu slijedi u kodnom isječku 4.2.

```
<template>

    //...
    <div class="form-group mb-4 child" style="width: 550px">
         <label for="password" class="block font-normal uppercase tracking-
   wide text-xs mb-1">Password</label>
        <input v-model="formData.password" type="password" id="password"
   name="password" class="border px-4 py-2 w-full rounded bg-gray-200">
    </div>

```

```
    //...

</template>

 <script>
  //...
    export default {
    data() {
      return {
        formData: {
          email: '',
          password: ''
        },
        passwordSuccessString: ''
        }
    },
 //...
}
```

Kodni isječak 4.2 *Unos lozinke u Vueu*

Ono što se nedvojbeno može izdvojiti kao prednost Reacta jest njegova popularnost. Ona doprinosi lakšem razvoju aplikacija jer kad programer naiđe na neki problem veće su šanse da se netko prije već susreo s nečim sličnim. Osim toga, veličina zajednice utječe i na potražnju na tržištu koja je značajno veća za React *developerima*, kao i na brzinu razvoja zavisnosti koje olakšavaju posao programerima. Iako je ekosustav Reacta znatno bogatiji, valja napomenuti kako se Vue aktivno fokusira i na razvoj rješenja za upravljanje stanjem (Vuex) i preusmjeravanje (VueRouter). React pak brigu o istim rješenjima prepušta zajednici koja je uistinu velika. Što se tiče upravljanja stanjem najpopularniji izbor u Reactu je Redux. Vue podržava Redux, ali najčešće je korišten Vuex. Oba su bazirana na Fluxu. Središte svake Redux aplikacije je kontejner *store* koji sadrži globalno stanje aplikacije. Stanje se nikada ne mijenja direktno već se pribavlja novo stanje. U *store-u* su smještene posebne funkcije zvane reduktori. Njihova je uloga ažuriranje stanja u ovisnosti o primljenoj akciji. Akcije opisuju što se dogodilo unutar aplikacije. Reduktor koji sadrži sve ostale reduktore zove se korijenski. Vuex umjesto reduktora koristi mutacije. Za razliku od Reduxa gdje je stanje uvijek nepromjenjivo, u Vuexu

se vršenjem mutacija podaci mijenjaju. Vuex je ipak jednostavniji za korištenje te zahtijeva manje koda. Što se tiče životnog ciklusa komponenti, razlike između ova dva radna okvira su minimalne. Glavne metode životnog ciklusa razlikuju se tek po imenu (primjerice *componentDidMount* i *componentWillUnmount* u Reactu ekvivalentne su *mounted* i *beforeDestroy* metodama u Vue.js-u). Reactove metode *componentDidCatch* za rukovanje greškama i *shouldComponentUpdate* koja omogućuje prekid osvježavanja životnog ciklusa radi optimizacije performansi ne postoje u Vue.js-u. React nadmašuje Vue kada je u pitanju testiranje. Još jedna prednost Reacta jest činjenica da je njegovo ovladavanje odlična baza za React Native pomoću kojeg se mogu kreirati mobilne aplikacije na sličan način. React Native također koristi JavaScript XML sintaksu. Kao glavni nedostatak Reacta može se izdvojiti manjak konzistentnosti koji proizlazi i iz činjenice da je na programeru velika sloboda te su struktura i dizajn aplikacije možebitno promjenjivi.

5. Zaključak

Sa sigurnošću se može utvrditi kako je softverska industrija jedna od onih koja se konstantno razvija i neprestance ide naprijed. Kako bi se *developeri* mogli što bolje fokusirati na razvoj funkcionalnosti svojih aplikacija, bitno je pritom što je manje vremena moguće utrošiti na razvoj detalja niže razine. Traganje za novim rješenjima stoga je od presudne važnosti te je ono dovelo do pojave kojekakvih radnih okvira. Glede klijentske strane razvoja web aplikacija posljednjih se godina kao najpopularniji izbori izdvajaju JavaScript radni okviri. Vue.js i React.js sagledani u ovome radu, uz Angular, daleko su najpopularniji odabir. Oba se koriste za izradu jednostraničnih aplikacija. Kada je riječ o odabiru između njih, on ponajprije ovisi o potrebama projekta, međutim valja napomenuti kako oba dijele mnogo sličnosti te ukoliko se svladaju koncepti jednog od njih, drugi ne bi trebao biti preveliki izazov. React se posljednjih godina profilirao u vodeći izbor i posjeduje veliku zajednicu. Vue je nešto mlađi od Reacta, ali njegova ga jednostavnost i izrazito temeljita dokumentacija čini odličnim izborom za početnike. Stoga je za očekivati je kako će rasti u budućnosti. Kao praktični dio ovoga rada izrađene su dvije identične RESTful aplikacije za vođenje apartmana pritom koristeći Laravel radni okvir na poslužiteljskoj strani. Korisniku je omogućena registracija i prijava kao i slanje zahtjeva za rezervacijom. Kombinacijom svih tehnologija opisanih u radu znatno je olakšan postupak kreiranja ove web aplikacije i poboljšana njezina kvaliteta. Iste također otvaraju mogućnost poboljšanja izrađene aplikacije u budućnosti i lakšeg potencijalnog uvođenja naprednijih značajki u aplikaciju po potrebi.

Bibliografija

[1] ANDREA CHIARELLI: "Beginning React: Simplify your frontend development workflow and enhance the user experience of your applications", Packt Publishing, Birmingham, 2018.

[2] BRETT NELSON: "Getting to Know Vue.js", Apress, New York, 2018.

[3] React dokumentacija, `https://reactjs.org`

[4] Vue dokumentacija, `https://vuejs.org`

[5] Laravel dokumentacija, `https://laravel.com/docs/8.x`

[6] React Bootstrap dokumentacija, `https://react-bootstrap.github.io`

[7] BoostrapVue dokumentacija, `https://bootstrap-vue.org`

[8] Tailwind dokumentacija, `https://tailwindcss.com`

[9] axios dokumentacija, `https://axios-http.com`

[10] Laravel Sanctum dokumentacija, `https://laravel.com/docs/8.x/sanctum#main-content`

[11] npm dokumentacija, `https://docs.npmjs.com/about-npm`

[12] `https://huspi.com/blog-open/definitive-guide-to-spa-why-do-we-need-single-page-applications/`, posjećeno 25. lipnja 2022.

[13] https://insights.stackoverflow.com/survey/2021#section-most-popular-technologies-web-frameworks, posjećeno 25. lipnja 2022.

[14] https://www.w3schools.com, posjećeno 26. lipnja 2022.

[15] https://www.freecodecamp.org/news/how-to-manage-state-in-your-react-apps/, posjećeno 26. lipnja 2022.

[16] https://blog.logrocket.com/implementing-jwt-authentication-laravel-9/, posjećeno 3. srpnja 2022.

[17] https://www.netsolutions.com/insights/single-page-application/, posjećeno 4. srpnja 2022.

[18] https://www.mindk.com/blog/react-vs-vue/#.Ysc0iS8Rp-U, posjećeno 4. srpnja 2022.

Sažetak

Predmet istraživanja ovoga rada bili su jedni od najpoznatijih radnih okvira za izradu web aplikacija kada je u pitanju klijentska strana: Vue.js i React.js. Objašnjena je potreba i važnost korištenja radnih okvira pri izradi web aplikacija. Na samom početku rada čitatelj je uveden u tematiku, objašnjen je koncept jednostraničnih aplikacija i koje probleme one rješavaju. Nadalje, u sljedećem su poglavlju radni okviri Vue.js i React.js detaljno objašnjeni te su sagledane njihove značajke. Novostečena teorijska saznanja potom su bila primijenjena za izradu dvije identične RESTful jednostranične aplikacije za vođenje apartmana koristeći oba radna okvira. Za razvoj poslužiteljskog dijela web aplikacije korišten je Laravel radni okvir i njegov paket Sanctum. Ovaj cjelokupni postupak je dokumentiran. Pri kraju rada pružana je detaljna usporedba Vuea i Reacta, naglašene su njihove sličnosti i razlike te prednosti i mane.

Ključne riječi **— Vue, React, Jednostranična aplikacija, Klijentska strana, Laravel, REST**

Abstract

This thesis focuses on two of the best known frontend frameworks for web application development: Vue.js and React.js. The need and importance of using such frameworks in web application development is covered in this paper. The beginning of the thesis familiarizes the reader with the topic and explains the concept of single page applications as well as which problems they solve. Furthermore, the detailed explanation of Vue.js and React.js frameworks, along with their charachteristics, is given in the next chapter. Newfound theoretical knowledge was then put to use by creating two identical RESTful single page applications for apartment management using both of the aforementioned frameworks. Laravel framework and its package Sanctum were used for developing the web application's server side. This whole process was fully documented. Towards the end the detailed comparison of Vue and React is given, their key similarities and differences are pointed out, as well as advantages and disadvantages.

Keywords **— Vue, React, Single page application, Frontend, Laravel, REST**