Razvoj web aplikacije za automatsko generiranje web stranica za iznajmljivanje apartmana

Kirinčić, Luka

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Prijediplomski sveučilišni studij računarstva

Završni rad

Razvoj web aplikacije za automatsko generiranje web stranica za iznajmljivanje apartmana

Rijeka, srpanj 2024.

Luka Kirinčić

0069093295

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Prijediplomski sveučilišni studij računarstva

Završni rad

Razvoj web aplikacije za automatsko generiranje web stranica za iznajmljivanje apartmana

Mentor: Izv. prof. dr. sc. Marko Gulić

Rijeka, srpanj 2024.

Luka Kirinčić

0069093295

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA ZAVRŠNE ISPITE

Rijeka, 21.03.2024.

Zavod: Zavod za računarstvo

Predmet: Razvoj web aplikacija

ZADATAK ZA ZAVRŠNI RAD

Pristupnik: **Luka Kirinčić (0069093295)**

Studij: Sveučilišni prijediplomski studij računarstva (1035)

Zadatak: **Razvoj web aplikacije za automatsko generiranje web stranica za iznajmljivanje apartmana / Development of a web application for automatic generation of web pages for renting apartments**

Opis zadatka:

Razviti web aplikaciju koja omogućuje registriranim korisnicima samostalno generiranje web stranica za kuće ili apartmane koje iznajmljuju. Cilj je omogućiti korisnicima postavljanje početne slike, teksta, podataka o apartmanu, definiranje preferiranih boja na web stranici, dodavanje slika apartmana te kontaktnih informacija. Korisnici će se moći registrirati i prijaviti na platformu radi pristupa alatima za generiranje web stranica. Nakon uređivanja, korisnici će moći pregledati svoje web stranice u stvarnom vremenu i preuzeti gotov HTML/CSS/JavaScript kod na svoje računalo. Za razvoj poslužiteljskog dijela web aplikacije treba koristiti Laravel radni okvir uz proizvoljno odabran sustav za upravljanje bazama podataka. Za razvoj klijentskog dijela aplikacije treba koristiti React JavaScript knjižicu za razvoj korisničkog sučelja uz učinkovito renderiranje aplikacije na uređajima s različitim veličinama zaslona.

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

Zadatak uručen pristupniku: 22.03.2024.

Predsjednik povjerenstva za
završni ispit:
prof. dr. sc. Miroslav Joler

Mentor:
doc. dr. sc. Marko Gulić

Izjava o samostalnoj izradi rada

Izjavljujem da sam samostalno izradio ovaj rad

Rijeka, srpanj 2024.

Lucka Krnčić

Ime prezime

SADRŽAJ

1 UVOD

U ovom radu izrađena je web aplikacija za automatsko generiranje web stranica za iznajmljivanje apartmana koja svakom korisniku na iznimno jednostavan način omogućuje stvaranje vlastite web stranice bez poznavanja programiranja i načina izrade web aplikacija. Aplikacija sadrži isti predložak za sve korisnike koji odluče izraditi svoju web stranicu, a svatko dodaje slike, tekst i ostale potrebne materijale po svom izboru. U konačnici, nakon što je stranica izrađena i pregledana, može se preuzeti u *.html* formatu te je spremna za korištenje na bilo kojem uređaju u bilo kojem trenutku bez posjedovanja ikakvih drugih tehnologija.

Projekt je temeljen na korištenju React-a [1], *open-source* JavaScript knjižnice koja pruža efikasne alate za izgradnju sučelja web aplikacija. React se ističe arhitekturom zasnovanom na komponentama, što olakšava organizaciju i održavanje koda. Komponente omogućavaju programerima da razvijaju modularna i ponovno upotrebljiva sučelja, što je posebno korisno u kompleksnim projektima kao što je izrada web stranice za apartmane.

Za razvoj poslužiteljske strane ovog projekta korišten je Laravel [2], PHP okvir za izradu web aplikacija. Laravel pruža razne ugrađene alate i funkcionalnosti koje olakšavaju proces razvoja na strani poslužitelja. Laravel podržava MVC (Model-View-Controller) arhitekturu, što pomaže u organizaciji koda i održavanju aplikacije. Također, Laravel dolazi s integriranim radnim okvirom Eloquent ORM (object-relational mapper) koji olakšava povezivanje Laravela s odabranim sustavom za upravljanje bazama podataka. Osim toga, Laravel pruža sigurnosne značajke poput ugrađenog sustava za autentifikaciju, pomažući programerima da osiguraju pouzdanu web aplikaciju. Kombinacija Reacta za klijentsku stranu i Laravela za poslužiteljsku stranu aplikacije osigurava efikasan i prilagodljiv okvir za izradu funkcionalne web stranice za apartmane s modernim i korisnički prijateljskim sučeljem.

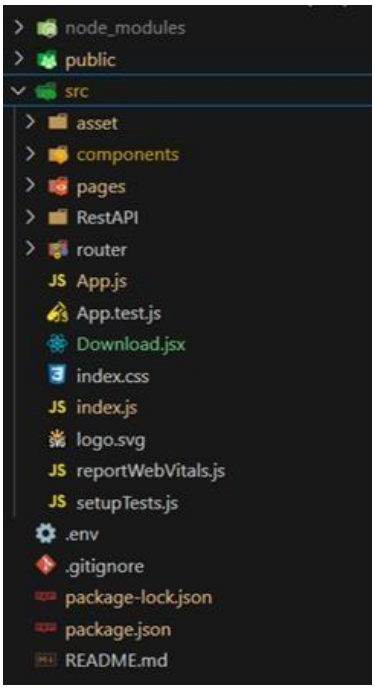
U narednom poglavlju biti će opisane sve tehnologije koje su, uz React i Laravel, korištene pri izradi ove aplikacije. Tu će biti fokus na pojedine React knjižnice te ostale biblioteke i pakete koji su pomogli u razvoju. Poglavlje 3 može poslužiti kao korisnički priručnik za korištenje ove aplikacije s obzirom da je u tom poglavlju opis rada same aplikacije te je tamo opisana svaka funkcionalnost i korak za preuzimanje *.html*

datoteke. U poglavlju 4 je riječ o razvoju i implementaciji specifičnih aplikacijskih funkcionalnosti na razini koda odnosno cijeli postupak kroz programski kod. Opisano je kako klijent pokreće zahtjev za nekom funkcionalnosti u aplikaciji, kako se to prihvaća na serverskoj strani (poslužitelj), pristup bazi (poslužitelj) te vraćanje odgovora korisniku. U konačnici, zadnje poglavlje, zaključak, je poglavlje u kojem se može pročitati sumirano sve o čemu se pisalo u poglavljima koji su mu prethodili.

2 TEHNOLOGIJE

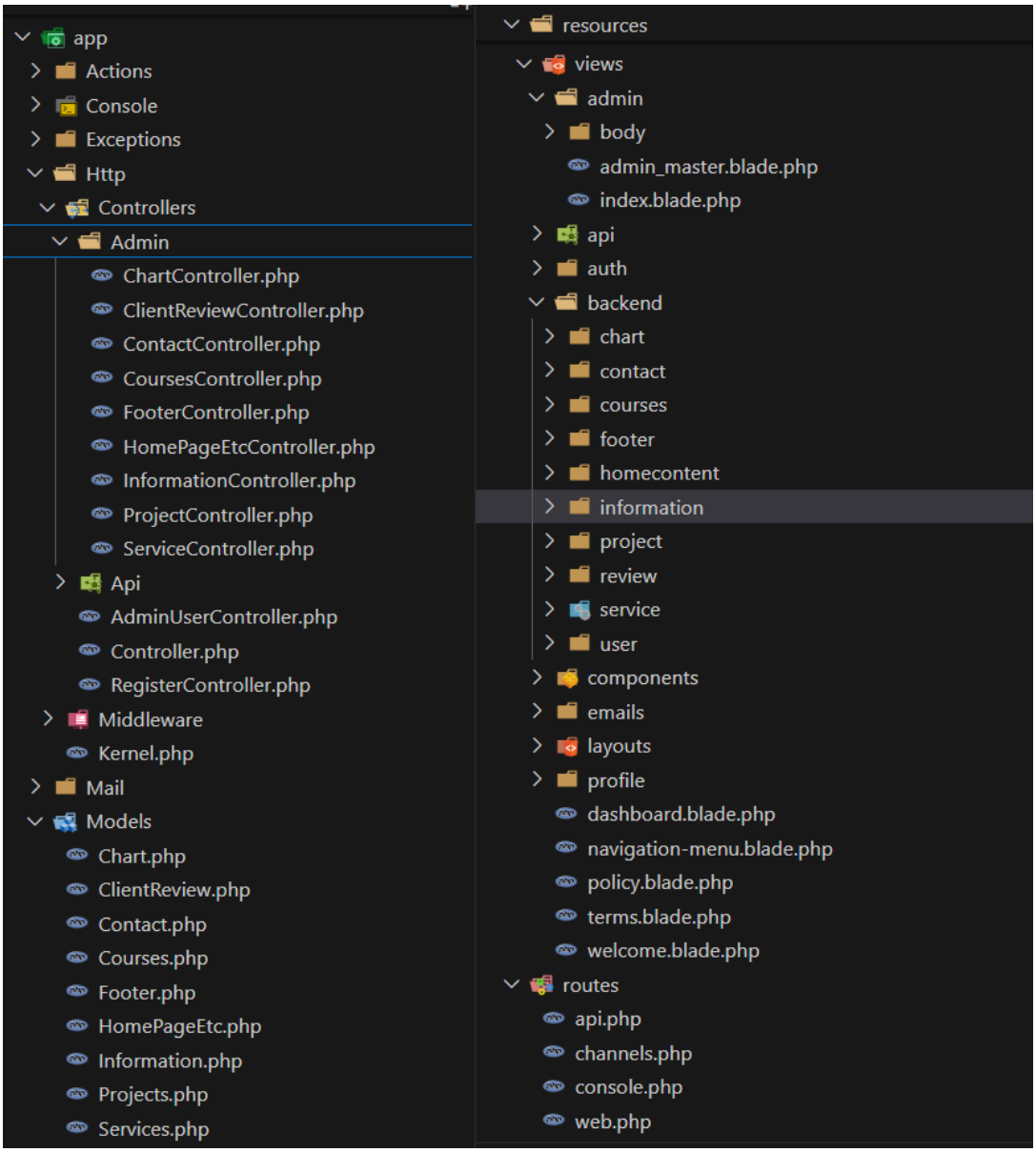
Izabrana tehnološka kombinacija za razvoj web stranice za apartmane obuhvaća React [1] za klijentsku stranu i Laravel [2] za poslužiteljsku stranu. React i Laravel zadovoljavaju visoke standarde performansi i korisničkog iskustva. Koristeći ovo okruženje, web aplikacija je suvremena i brza, a također ima i laku mogućnost proširenja ili nadogradnje.

Na klijentskoj strani, projekt je formatiran na način da su sve komponente poput navigacijskog okna, podnožja i ostalih implementirani u mapi *components* u *.jsx* formatu. Te komponente su uključene u stranice za koje su namijenjene u mapi *pages*. Stranice su također implementirane u *.jsx* formatu (slika 2.1). [3]



Slika 2.1 Organizacija datoteka u Reactu

Na poslužiteljskoj strani, u mapi *app* definirani su svi kontroleri i modeli, a u mapi *resources* su definirani svi pogledi. Rute koje koristimo za povezivanje s klijentskom stranom su definirane u mapi *routes*. Kompletna organizacija datoteka u Laravelu vidljiva je na slici 2.2.



Slika 2.2 Organizacija datoteka u Laravelu

2.1 React

U okviru Reacta, koji predstavlja biblioteku za izradu korisničkih sučelja, projekt se temelji na pristupu gdje se komponente koriste za izgradnju složenih i dinamičnih web aplikacija. React omogućava modularnost, ponovnu upotrebu koda i jednostavno održavanje, čime se olakšava razvoj sofisticiranih korisničkih sučelja. Reactova verzija 18, koja se koristi u projektu, je starija verzija te zahtijeva upotrebu konstruktora i *setState* umjesto modernijih funkcija poput *useState* i *useEffect*. Unatoč razlikama u

sintaksi, ovo omogućava praćenje stanja komponenti i održavanje interaktivnosti aplikacije. Za započeti projekt u Reactu, potrebno je u terminalu unijeti komandu *npm create-react-app@ime_mape* kako bi se u željenoj mapi instalirale sve potrebe komponente za izradu projekta u Reactu. Kako bi se pak pokrenuo server, potrebno je unijeti komandu *npm start*.

2.1.1 React Bootstrap

Unutar Reacta korištena je i React Bootstrap knjižnica [4] kako bi se pojednostavila integracija Bootstrap komponenata u React aplikaciju. React Bootstrap omogućuje korištenje Bootstrap dizajna unutar React okoline, nudeći set responzivnih komponenata koje pružaju konzistentan izgled i ponašanje na različitim uređajima. Ključne komponente koje su korištene poput *Container*, *Row* i *Col* omogućavaju organizaciju elemenata na stranici, što doprinosi preglednosti i prilagodljivosti. *Container* ograničava širinu elemenata te se često koristi za grupiranje elemenata unutar jednog logičkog bloka. Komponenta *Row* predstavlja redak unutar *Containera*. *Row* se koristi za postizanje horizontalnog rasporeda elemenata. Ova komponenta definira stupce unutar *Rowa*. Col se koristi za postizanje vertikalne strukture elemenata unutar reda. Svaki Col može sadržavati specifične sadržaje i prilagoditi svoj prostor prema sadržaju. Primjer korištenja komponenti iz React Bootstrap knjižice vidljiv je na slici 2.3.

```
return (
    <Fragment>
        <Container fluid={true} className="footerSection">
            <Row>
                <Col lg={3} md={6} sm={12} className="p-5 text-center">
                    <h2 className="footerName text-center" >Follow us</h2>
                    <div className="social-container">
                    <a className="facebook social" href={this.state.facebook}>
                        <FontAwesomeIcon icon={faFacebook} size="2x"></FontAwesomeIcon>
                    </a>
                    <a className="youtube social"href={this.state.youtube}>
                    <FontAwesomeIcon icon={faYoutube} size="2x"></FontAwesomeIcon>
                    </a>
                    <a className="twitter social" href={this.state.twitter}>
                    <FontAwesomeIcon icon={faTwitter} size="2x"></FontAwesomeIcon>
                    </a>
                    </div>
                </Col>
                <Col className={this.state.loaderClass}>
                    <Loading/>
                </Col>
                <Col lg="3" md={6} sm={12} className={this.state.mainDivClass}>
                    <h2 className="footerName">Adresa</h2>
                    <p className="footerdescription">
                        {this.state.address}<br />
                        <FontAwesomeIcon icon={faEnvelope}></FontAwesomeIcon>
                        Email: {this.state.email}<br />
                        <FontAwesomeIcon icon={faPhone}></FontAwesomeIcon>
                        Mobitel: {this.state.phone}<br />
                    </p>
                </Col>
```

Slika 2.3 Korištenje komponenata iz react-bootstrap knjižnice

2.1.2 React Router Dom

React Router DOM [5] je biblioteka koja omogućava upravljanje navigacijom i rutama u React aplikacijama koje se izvode u web preglednicima. Omogućava da se kreiraju aplikacije s više stranica, koje mogu dinamički mijenjati sadržaj na osnovu URL-a, bez potrebe za ponovnim učitavanjem stranice. Najveća prednost ove biblioteke je što aplikaciju čini bržom. Komponenta *Route* je osnovna gradivna jedinica React Routera. Koristi se za definiranje veze između URL putanje i komponente koja se treba renderirati kada je ta putanja aktivna. U starijim verzijama *React Router DOM*, *Switch* komponenta se koristila za obuhvaćanje više *Route* komponenti i osiguravanje da se renderira samo prva *Route* komponenta koja se podudara s trenutnim URL-om. U

najnovijim verzijama komponenta *Switch* je zamijenjena komponentom *Routes*. Korištenje komponenti ove biblioteke vidljivo je na slici 2.4.

```
<Fragment>
    <Switch>
        <Route exact path="/" component={HomePage} />
        <Route exact path="/gallery" component={ServicePage} />
        <Route exact path="/projects" component={ProjectsPage}/>
        <Route exact path="/about" component={AboutPage}/>
        <Route exact path="/contact" component={ContactPage}/>
        <Route exact path="/refund" component={RefundPage}/>
        <Route exact path="/terms" component={TermsPage}/>
        <Route exact path="/privacy" component={PrivacyPage}/>
        <Route exact path="/projectdetails/:projectID/:projectName" component={ProjectDetailsPage}/>
        <Route component={PageNotFound}/>
    </Switch>
</Fragment>
```

Slika 2.4 Korištenje komponenti iz React Router DOM knjižnice

2.2 Laravel

Laravel je zadužen za poslužiteljsku stranu, tzv. *backend*. Princip rada je na *Model-View-Controller* arhitekturi. MVC (*Model-View-Controller*) arhitektura razdvaja aplikaciju na tri međusobno povezane komponente: *Model* (koji upravlja podacima i logikom aplikacije), *View* (koji predstavlja korisničko sučelje) i *Controller* (koji obrađuje korisničke unose i ažurira *Model* i *View*).

Za uspješno postavljanje Laravel okruženja, prvo je potrebno stvoriti Laravel projekt naredbom *composer create-project laravel/laravel example-app*. Kada se pokreće server, onda se koristi naredba *php artisan serve*. [7]

Rute definirane u api.php usmjeravaju zahtjeve prema odgovarajućim kontrolerima. Kontroleri sadrže logiku za dohvaćanje, ažuriranje, brisanje i spremanje podataka. Modeli služe kao sloj apstrakcije za rad s bazom podataka, dok se migracije koriste za definiranje strukture baze podataka putem koda. [6]

U Laravelu, *Blade templating engine* pruža snažan alat za definiranje dinamičkih i statičkih stranica unutar web aplikacija. Blade omogućuje lakše upravljanje kodom i prikazivanje podataka na frontendu. Također, u Bladeu se pozivaju imena ruta definiranih u *web.php* datoteci kao što je vidljivo na slici 2.5. Jedna od ključnih mogućnosti Blade-a je organizacija koda i jednostavno uključivanje dijelova stranica

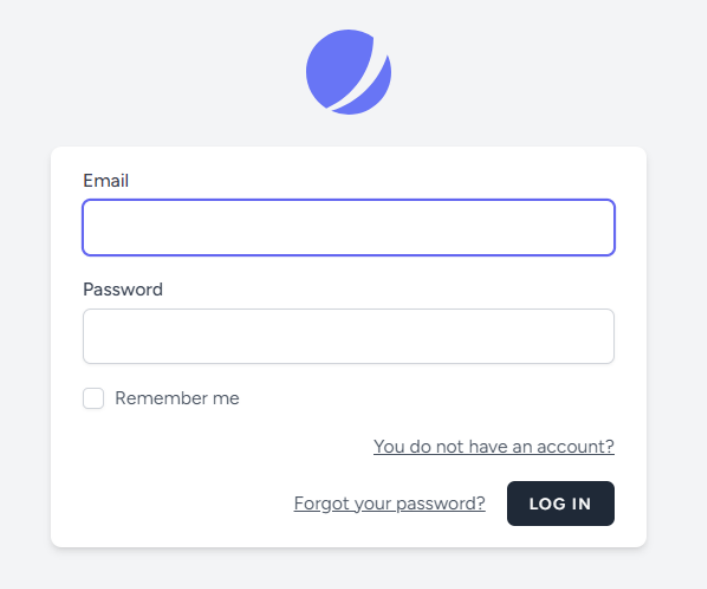
putem direktiva. Dijelove web stranice moguće je definirati u različitim blade datotekama i uključivati ih gdje god je to potrebno koristeći *@extends*, *@section* i *@endsection*.

```
<div class="deznav">
            <div class="deznav-scroll">
                <ul class="metismenu" id="menu">
                    <li><a class="has-arrow ai-icon" href="javascript:void()" aria-expanded="false">
                            <i class="flaticon-077-menu-1"></i>
                            <span class="nav-text">User Profile</span>
                        </a>
                        <ul aria-expanded="false">
                            <li><a href="{{ route('user.profile') }}">User Profile</a></li>
                            <li><a href="{{ route('change.password') }}">Change Password</a></li>
                        </ul>

                    </li>

                    <li><a class="has-arrow ai-icon" href="java61script:void()" aria-expanded="false">
                            <i class="flaticon-061-puzzle"></i>
                            <span class="nav-text">Information</span>
                        </a>
                        <ul aria-expanded="false">
                            <li><a href="{{ route('all.information') }}">All Information</a></li>
                            <li><a href="{{ route('add.information') }}">Add Information</a></li>
                        </ul>

                    </li>
```

Slika 2.5 primjer ruta u blade datoteci

2.2.1 Laravel JetStream

Laravel Jetstream [8] je biblioteka za projekt u Laravelu koja pruža snažan i sigurnosnI temelj za web aplikaciju. Uključuje značajke za autentifikaciju, upravljanje korisnicima, autentifikaciju u 2 koraka, API podršku putem Laravel Sanctuma i još mnogo toga. U okviru ovog projekta, *JetStream* knjižnica je korištena za registraciju i prijavu korisnika. Također, moguće je i izmijeniti lozinku odnosno zatražiti novu ukoliko je postojeću lozinku korisnik zaboravio. Izgled okvira je moguće izmijeniti, no za potrebe ovog projekta ostavljen je zadani dizajn iz *JetStream* knjižnice. Dizajn okvira moguće je vidjeti na slici 2.6, a za sve funkcionalnosti (prijava, registracija, promjena lozinke), prozor je istog izgleda samo s različitim input poljima.



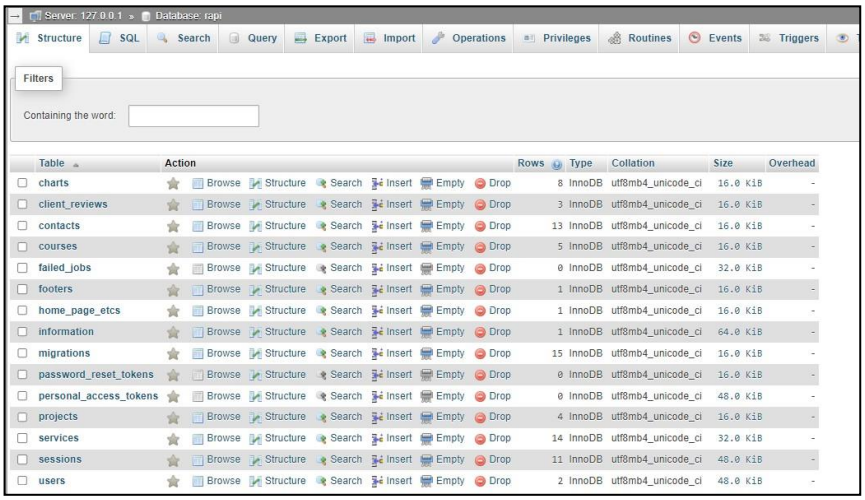
Slika 2.6 JetStream okvir za prijavu korisnika

2.3 MySQL

MySQL je *open-source* sustav za upravljanje relacijskim bazama podataka koji koristi SQL za pristup i upravljanje podacima. MySQL je poznat po svojoj brzini, pouzdanosti i jednostavnosti upotrebe, što ga čini popularnim izborom za web aplikacije, posebno u kombinaciji s PHP-om u LAMP *stacku* (Linux, Apache, MySQL, PHP/Perl/Python). Također nudi razne mogućnosti poput replikacije, skalabilnosti i podrške za različite vrste skladištenja podataka.

2.4 PhpMyAdmin

Baza podataka, smještena na *phpMyAdmin*, predstavlja grafičko sučelje za manipuliranje bazom podataka. Inicijalni podaci su ručno uneseni, a kasnije su ažurirani putem Laravel kontrolera nakon promjena na poslužiteljskoj strani. Ova sinkronizacija osigurava konzistentnost podataka između klijentskog i poslužiteljskog dijela aplikacije, pružajući sveobuhvatan i funkcionalan web doživljaj za korisnike. U okviru ovog projekta kreirana je baza *rapi* te je vidljiva zajedno sa svim svojim atributima na slici 2.7.



Slika 2.7 Baza podataka prikazana preko PhpMyAdmina

2.5 CSS

CSS (Cascading Style Sheets) [9] je jezik za stiliziranje web stranica koji se koristi za definiranje izgleda i formatiranja HTML elemenata. CSS omogućava razdvajanje sadržaja od stilova, što donosi mnoge prednosti u razvoju web stranica. Razdvajanjem stilova, kod postaje čišći i lakše se održava. CSS je glavni faktor u kreiranju pristupačnijih web stranica koje su lakše za navigaciju i korištenje, uključujući podršku za različite uređaje i korisničke postavke. To je ključan alat za web dizajnere i developere jer omogućava kreiranje vizualno privlačnih web stranica te ga to čini ga čini nezamjenjivim dijelom modernog web razvoja.

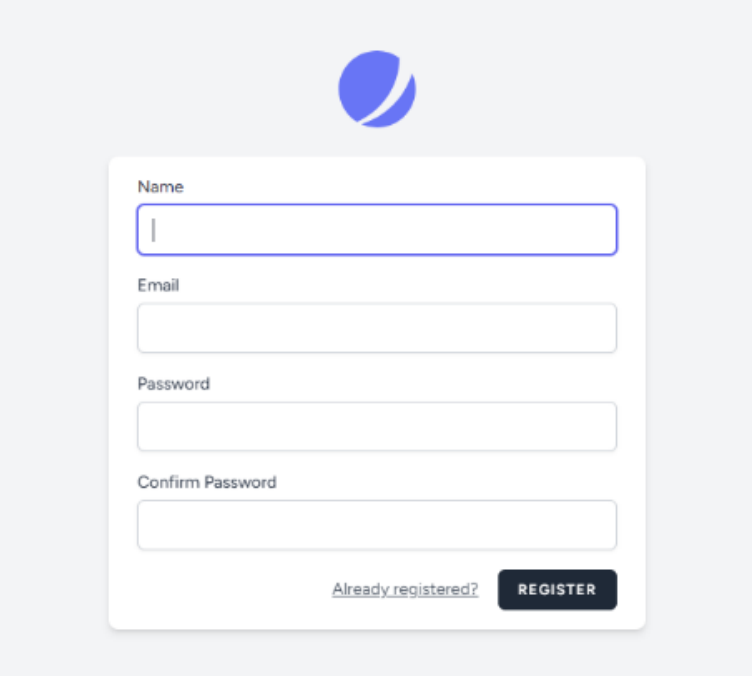
.

3 OPIS RADA APLIKACIJE

U ovom poglavlju biti će opisane sve mogućnosti koje ova aplikacija nudi te kako ju koristiti. Svaki korak od stvaranja profila do korištenja preuzete *.html* datoteke opisan je u ovom poglavlju, stoga bi svaki korisnik aplikacije bez ikakvog znanja o razvoju i korištenju aplikacije nakon čitanja ovog poglavlja trebao razumjeti kako koristiti aplikaciju.

3.1 Prijava i registracija korisnika

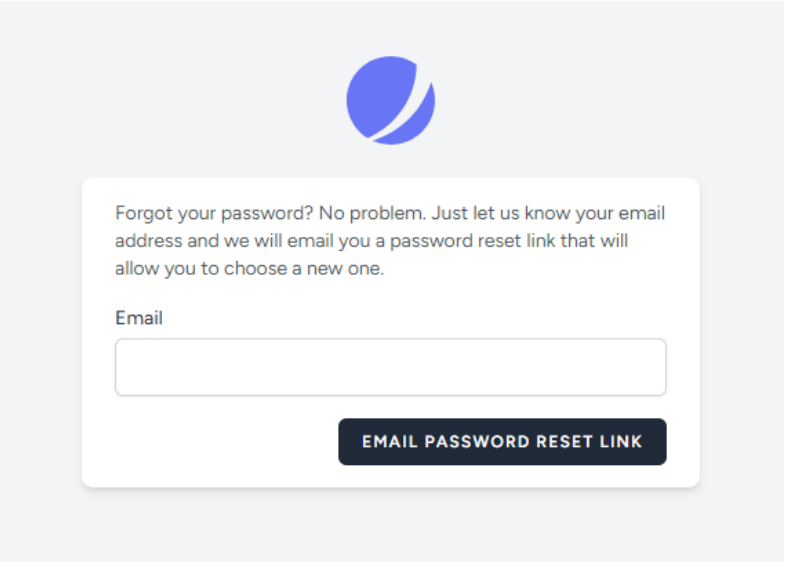
Prvi korak s kojim se korisnik susreće pri početku izrade vlastite web stranice je registracija i stvaranje profila. Za ovu funkcionalnost iskorištena je već spomenuta Laravel JetStream knjižnica [8]. Osoba koja stvara svoj profil obvezna je unijeti ime, email i lozinku (slika 3.1). Klikom na gumb *Register*, korisnik je uspješno stvorio svoj račun i izradio vlastiti profil s kojim se ubuduće može prijaviti te nije potrebno svaki put izrađivati novi profil.



Slika 3.1 JetStream prozor za registraciju novih korisnika

Ukoliko osoba već posjeduje svoj vlastiti račun, onda se ne registrira već se prijavljuje. Tada je potrebno unijeti email i lozinku koji su uneseni pri izradi računa, tj. pri registraciji. Prozor za prijavu je već prikazan na slici 2.6.

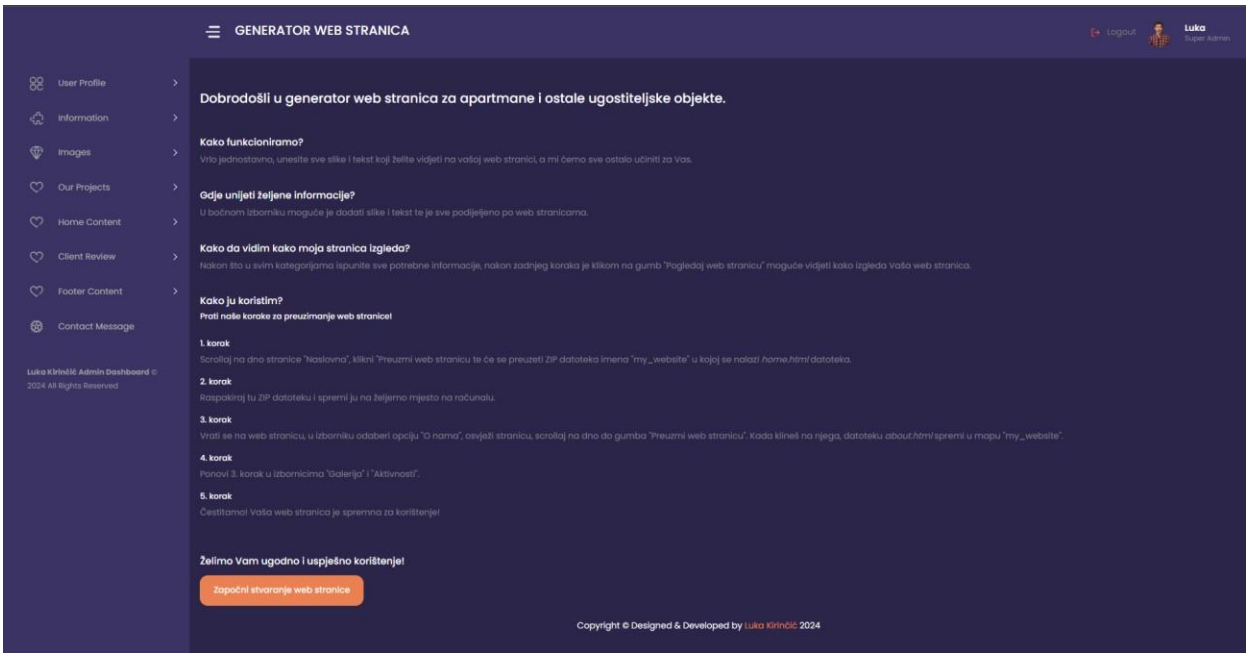
Osim prijave i registracije, već postojeći korisnik može i izmijeniti lozinku ukoliko ju je zaboravio. Mogućnost pristupa tom prozoru je klikom na *Forgot your password?*. Taj gumb se nalazi unutar okvira za prijavu te se klikom na njega otvara okvir vidljiv na slici 3.2. Kako bi korisnik izmijenio lozinku, potrebno je unijeti email s kojim se registrirao te se u daljnjim koracima postojećim funkcijama iz JetStream biblioteke otvaraju prozori za unos nove lozinke.



Slika 3.2 Forgot password prozor

3.2 Generator web stranica

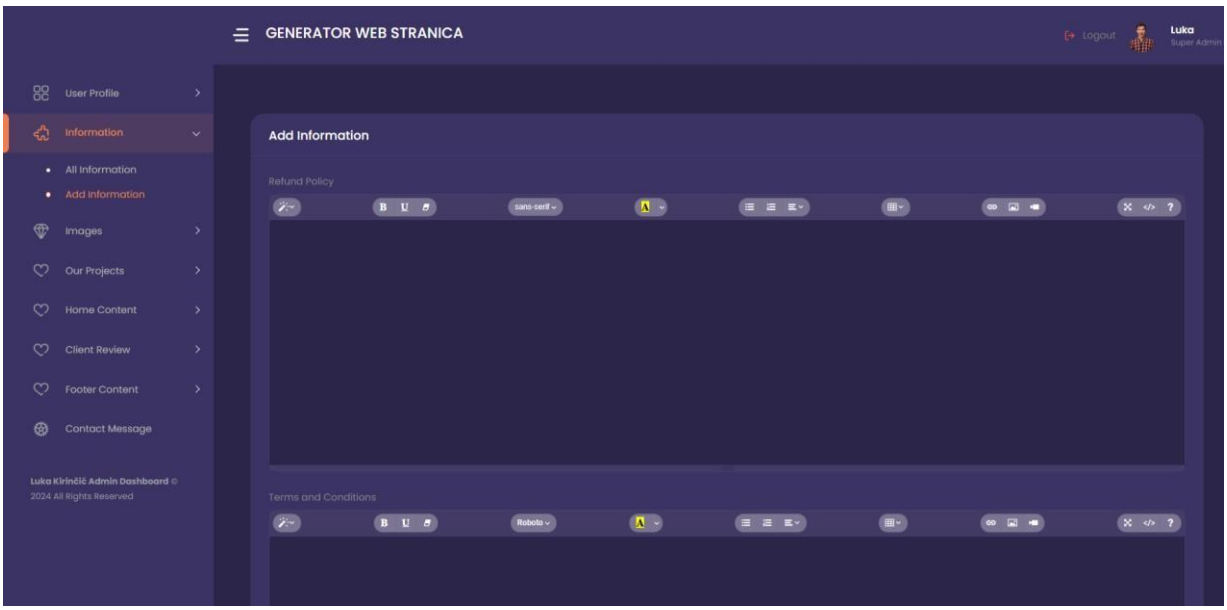
Nakon prijave odnosno registracije dolazi se na početni zaslon (slika 3.3.). Na početnom zaslonu vidljiv je bočni izbornik gdje su vidljive sve moguće opcije što se dodaje na stranicu ove web aplikacije. Na vrhu stranice je ime korisnika te gumb za odjavu. Ostatak stranice zauzimaju upute kako koristiti ovaj generator web stranica. Klikom na gumb *Započni stvaranje web stranice*, kreće se s dodavanjem željenog sadržaja.



Slika 3.3 Početni zaslon

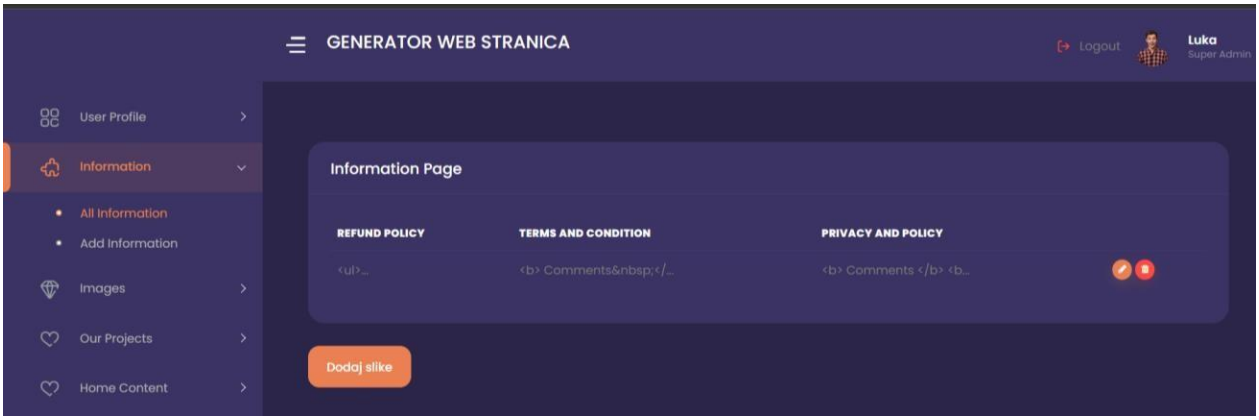
3.3 Dodavanje informacija

Klikom na gumb *Započni stvaranje web stranice* vidljiv na slici 3.3, otvara se prozor za dodavanje informacija (slika 3.4.). Tamo je potrebno dodati Refund Policy, Terms and Conditions te Privacy Policy. U tekstualnom okviru moguće je odabrati željeni font i ostala oblikovanja slova dostupna u jednostavnim uređivačima dokumenata. Na dnu te stranice nalazi se gumb *Add Information* te su klikom na njega dodane sve željene informacije.



Slika 3.4 Add Information ekran

Nakon klika na taj gumb, otvara se prozor sa svim informacijama te je unesene informacije moguće urediti ili izbrisati (slika 3.5). Ukoliko korisnik izbriše i želi dodati nove informacije, to je moguće klikom na *Information* u bočnom izborniku. Tada se otvara padajući izbornik gdje odaberemo *Add Information*.

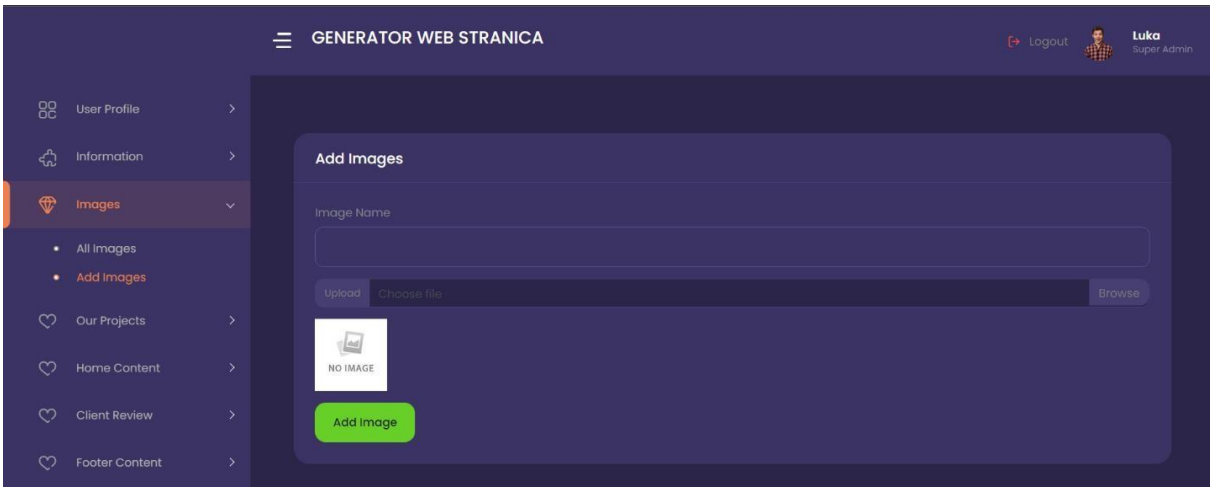


Slika 3.5 All information ekran

Tu je važno napomenuti kako na stranici nije potrebno niti je moguće prikazati više različitih verzija *Terms and conditions*, *Privacy Policy* i ostaloga, ali ih je moguće unijeti u bazu. Na stranici se uvijek prikazuju informacije koje su prve nadodane u bazu, no ukoliko se primjerice uklone informacije prve po redoslijedu, na web stranici će se prikazati one nadodane nakon tih koje su uklonjene. Ovo se odnosi na sve elemente koji se mogu dodati na web stranicu, a o kojima će biti riječ u ovom poglavlju.

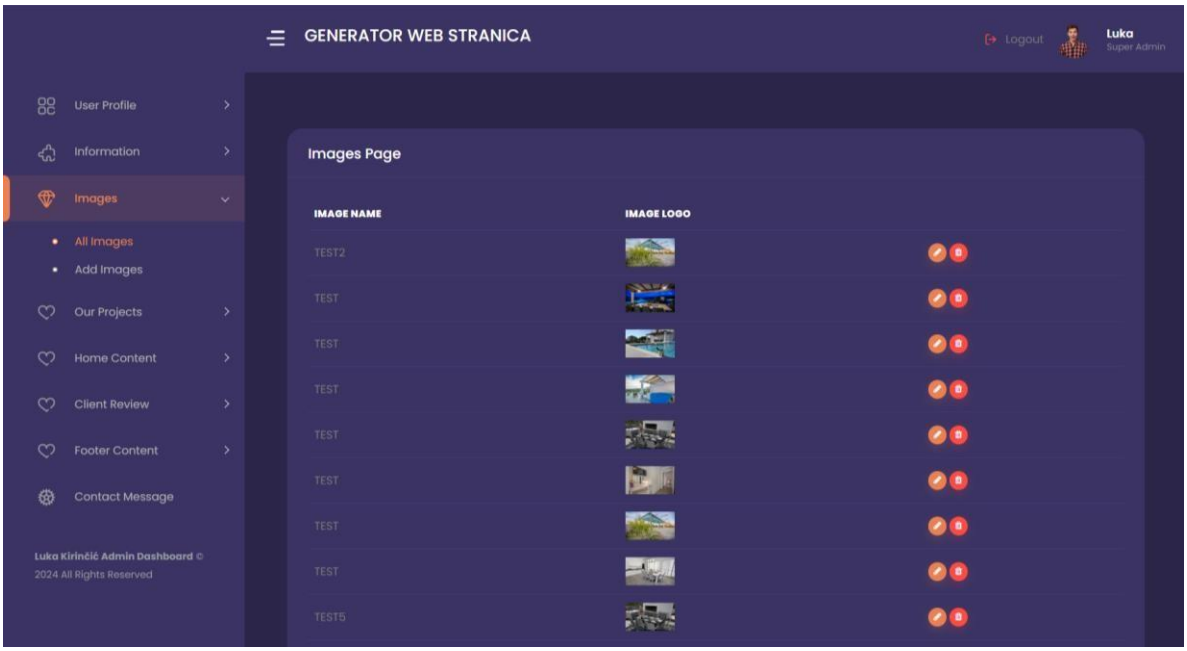
3.4 Dodavanje slika

Klikom na *Dodaj slike* otvara se novi prozor, možda i najvažniji za funkcionalnost ovog projekta. U tom prozoru potrebno je dodati željenu sliku te svakoj slici dodijeliti proizvoljni naziv (slika 3.6).



Slika 3.6 Ekran za dodavanje slika

Kada se doda željena slika klikom na gumb *Add image*, kao i slučaju s informacijama otvara se prozor sa svim dodanim slikama (slika 3.7). Na tom ekranu vidljiva je mala ikona te slike i njezin naziv.

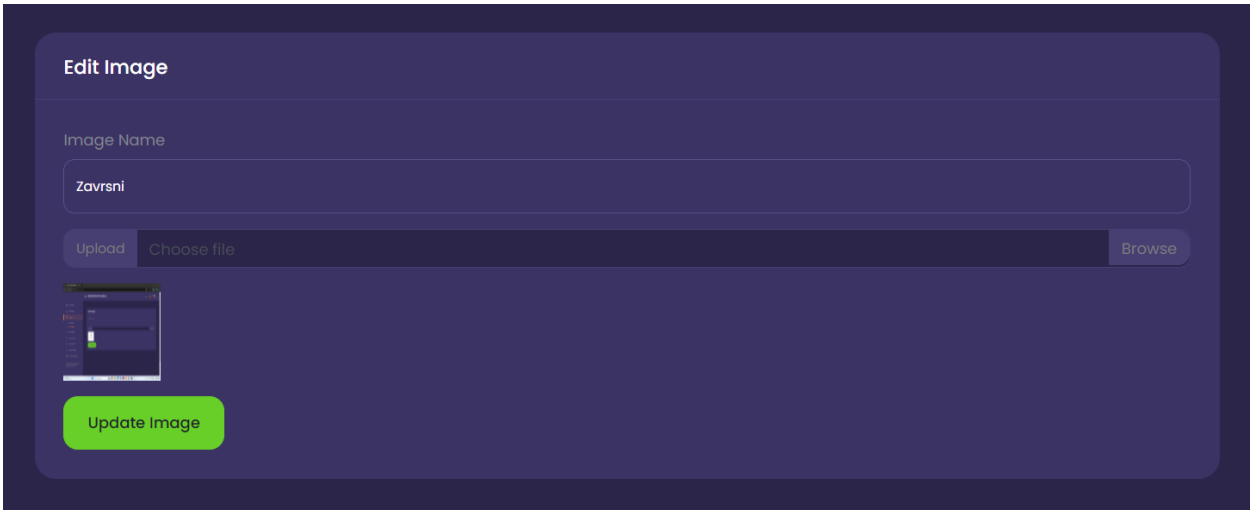


Slika 3.7 Ekran sa svim slikama

Ukoliko je potrebno dodati još slika, potrebno je kliknuti na *Add images* u lijevom bočnom izborniku te se otvara isti prozor kao na slici 3.6.

Kao što je također vidljivo na prethodnoj slici, neku sliku je moguće ukloniti te ona onda neće biti prikazana na web stranici. Svaku sliku je moguće i urediti, odnosno

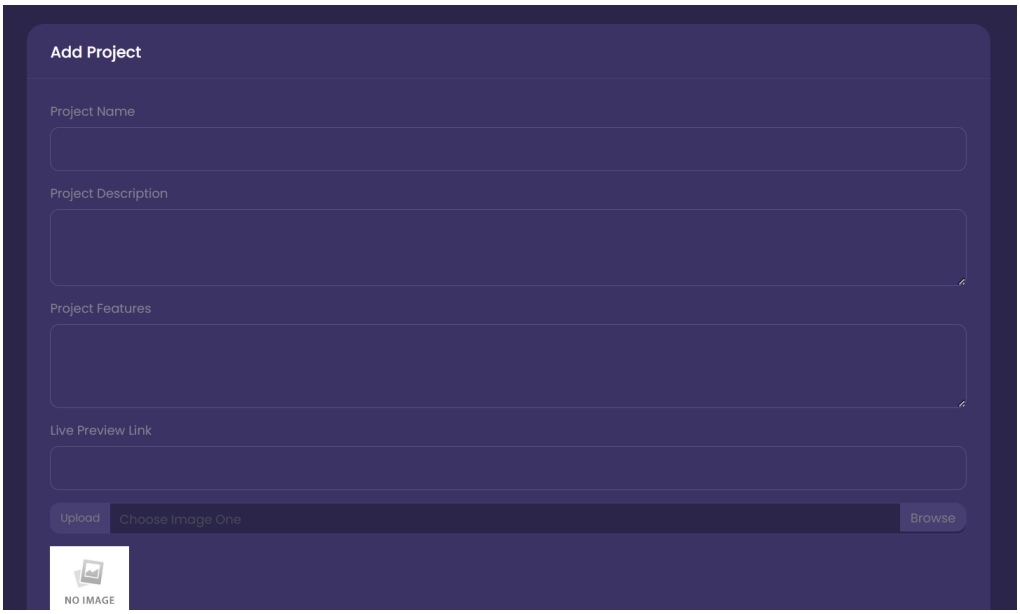
može joj se dodijeliti drugi naziv ili se postojeća slika može zamijeniti nekom drugom (slika 3.8). Stoga ako se želi izbrisati neka slika, a ujedno i dodati nova na njezino mjesto, nije potrebno dodavati novu sliku već urediti postojeću. To je iznimno važno jer ako se dodaje nova slika ta slika će se prikazati posljednja u galeriji, a u ovom primjeru će biti prikazana na mjestu slike koja je uklonjena .



Slika 3.8 Ekran za ažuriranje slike i pripadajućih podataka

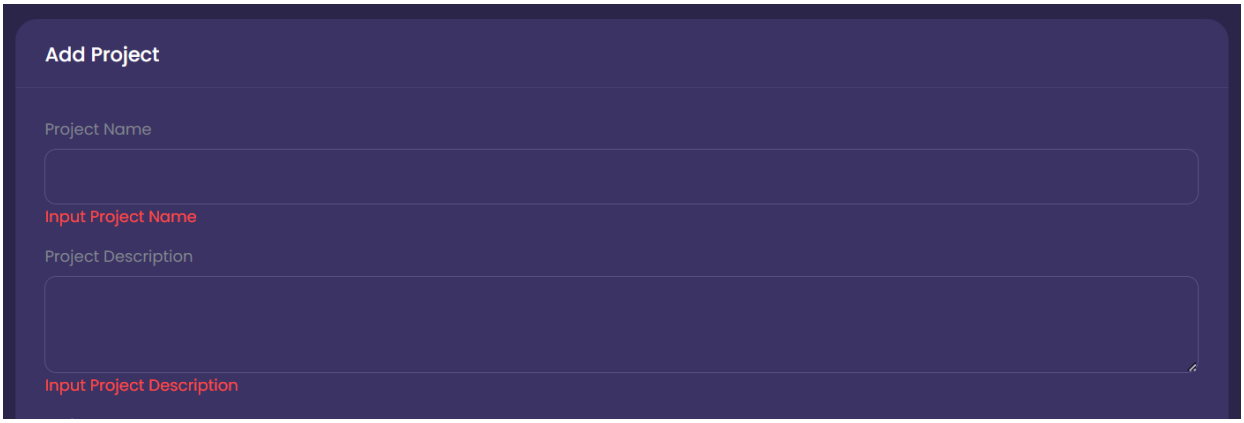
3.5 Dodavanje projekata

Povratkom na *All images* stranicu, ispod svih slika nalazi se gumb *Dodaj aktivnosti* na koji korisnik može kliknuti. Ideja ove funkcionalnosti je da se mogu dodati neke aktivnosti koje su dostupne u blizini tog prostora za najam kao što su rafting, zipline neki festival i slično. Zato je u tom prozoru potrebno dodati naziv projekta, opis projekta, neke njegove značajke, poveznica web stranice za detaljnije informacije te sliku (slika 3.9).



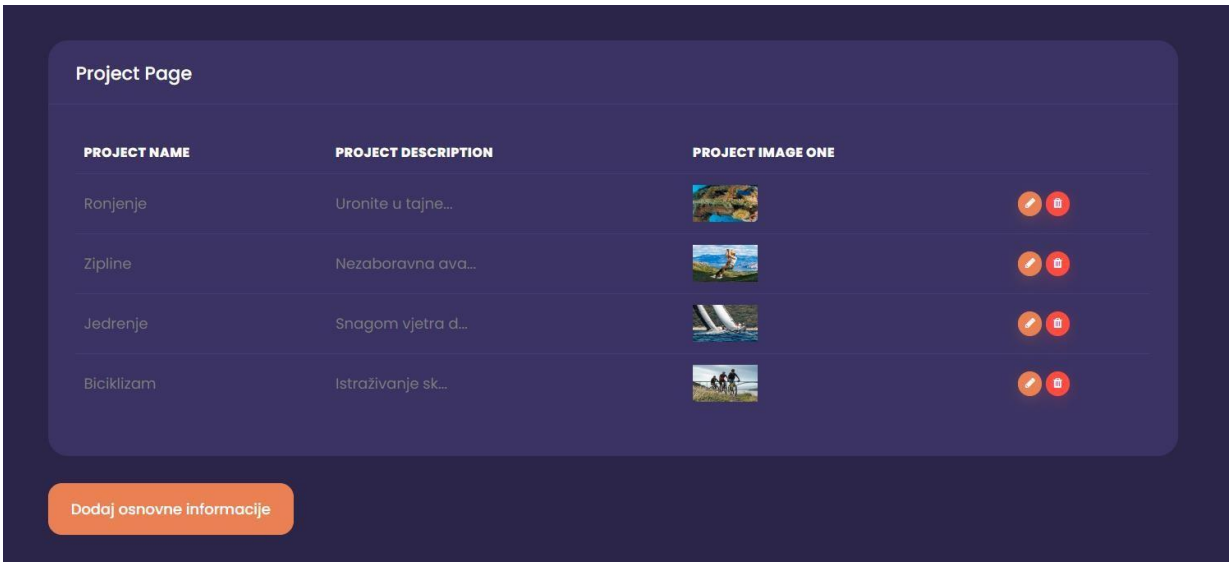
Slika 3.9 Ekran za dodavanje projekta

Ukoliko se primjerice pokušati dodati novi projekt bez da su ispunjena sva polja, onda se javlja poruka upozorenja ispod polja koju je potrebno ispuniti (slika 3.10). Isti oblik upozorenja javlja se na svim elementima koje možemo dodati (slike, informacije) ukoliko se ne ispuni određeno polje.



Slika 3.10 Poruka upozorenja za prazna tekstualna polja

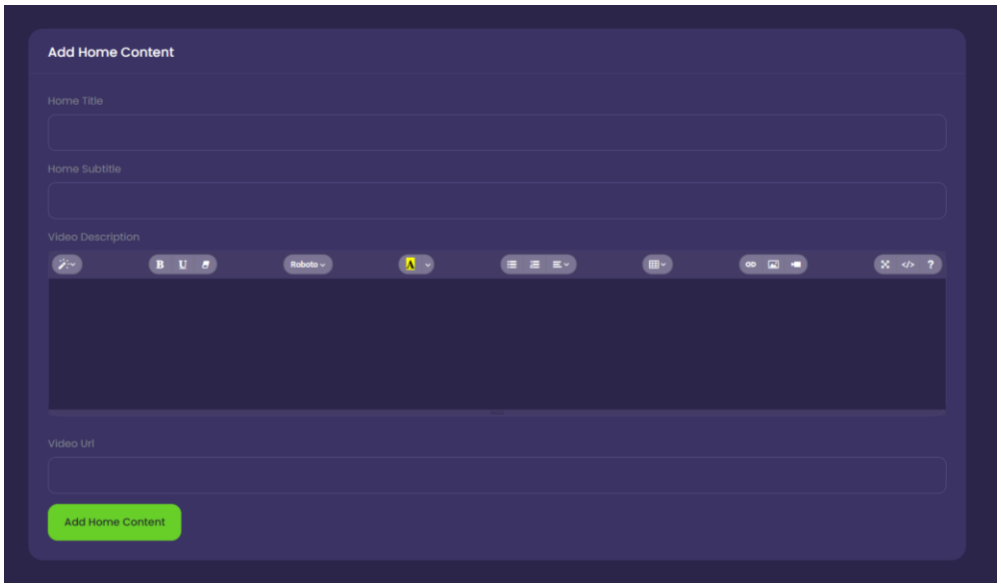
Kao što je bio slučaj i kod dodavanja ostalih elemenata stranice, nakon klika na *Add Project* otvara se stranica sa svim dodanim projektima te je svaki moguće u potpunosti ukloniti ili urediti bilo koji element tog projekta (slika 3.11).



Slika 3.11 Popis dodanih projekata

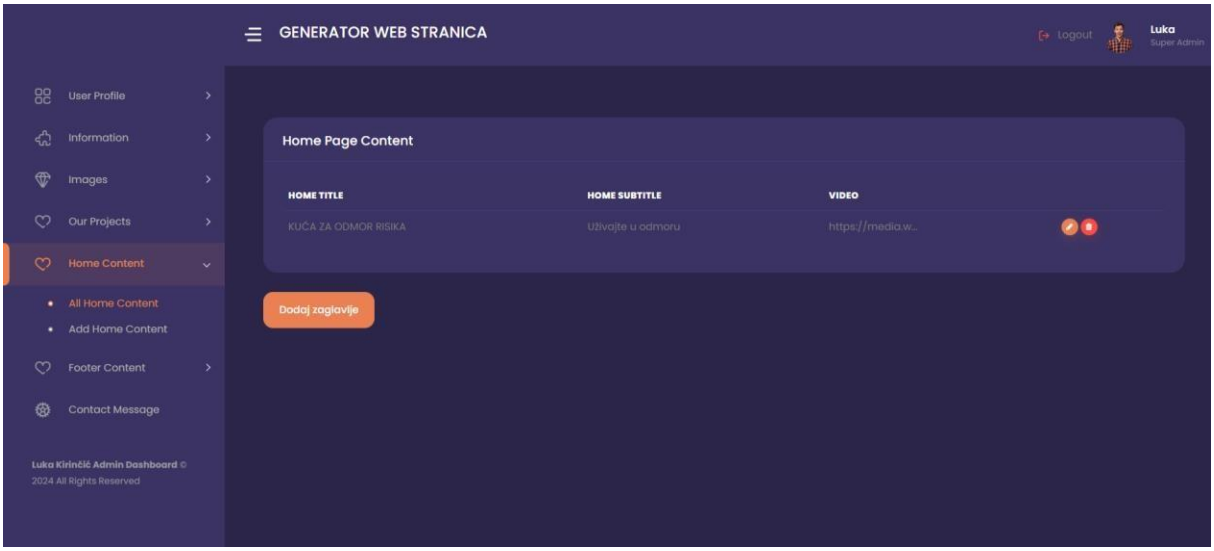
3.6 Dodavanje sadržaja naslovne stranice

Iduća kategorija je dodavanje sadržaja na naslovnu stranicu. Tu je potrebno dodati naziv vlastitog apartmana ili kuće za odmor, podnaslov naslovne stranice, poveznica na video tog objekta i opis uz video (slika 3.12). Kao i prozor za unos informacija (*privacy policy, terms and conditions* i sl.), tako je i prozor za unos opisa videa tekstni okvir u kojem je moguće odabrati željeni font i ostale elemente koje sadržavaju tekstni uređivači.



Slika 3.12 Ekran za dodavanje sadržaja naslovne stranice

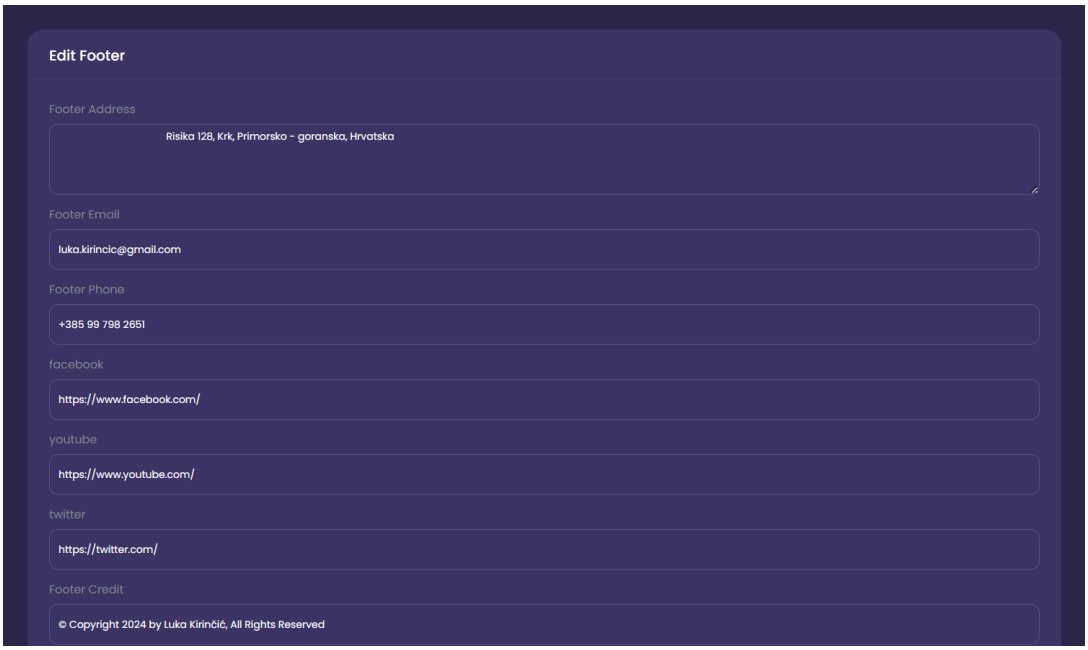
Kada se doda sadržaj naslovne stranice, moguće ga je pregledati, urediti ili obrisati na ekranu koji se otvara klikom na *Add home content* ili klikom na element *All home content* u bočnom izborniku (slika 3.13).



Slika 3.13 Ekran sa sadržajem naslovne stranice

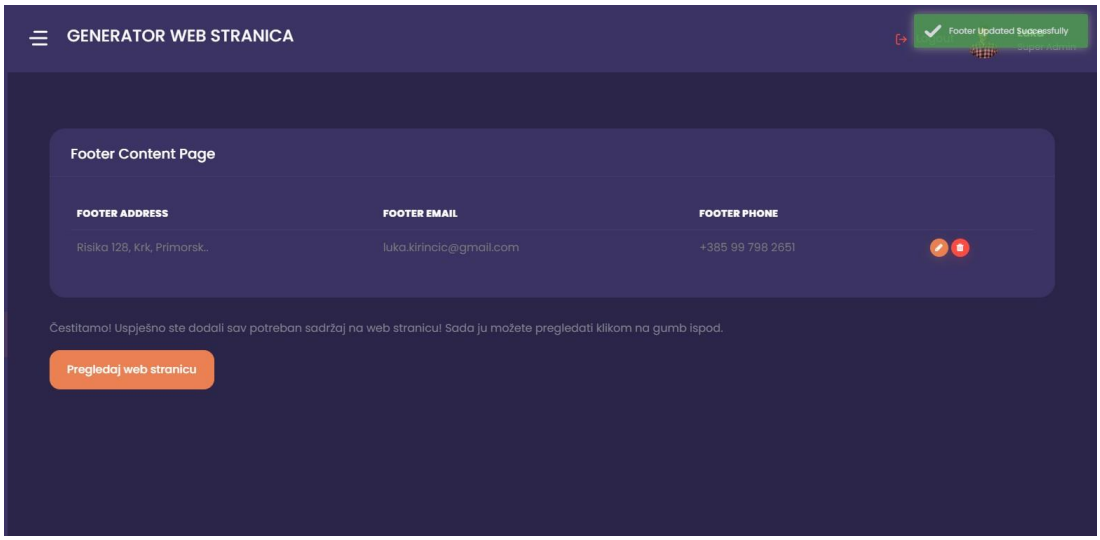
3.7 Dodavanje sadržaja zaglavlja

Iduća, a ujedno i posljednja kategorija elemenata koji se mogu dodati na web stranicu su elementi zaglavlja. U zaglavlju je moguće dodati adresu objekta, e-mail adresu, kontakt broj, poveznice na Facebook, Youtube i Twitter profile te tzv. *credit*, odnosno kako nitko ne bi mogao bespravno iskoristiti novoizrađenu web stranicu (slika 3.14).



Slika 3.14 Edit Footer ekran

Kada se elementi zaglavlja uspješno dodaju, vraćamo se na ekran gdje su vidljivi svi nadodani elementi zaglavlja. Također, u gornjem desnom kutu prikazuje se *Toast* poruka za uspješno izvršeno nadodavanje (slika 3.15).



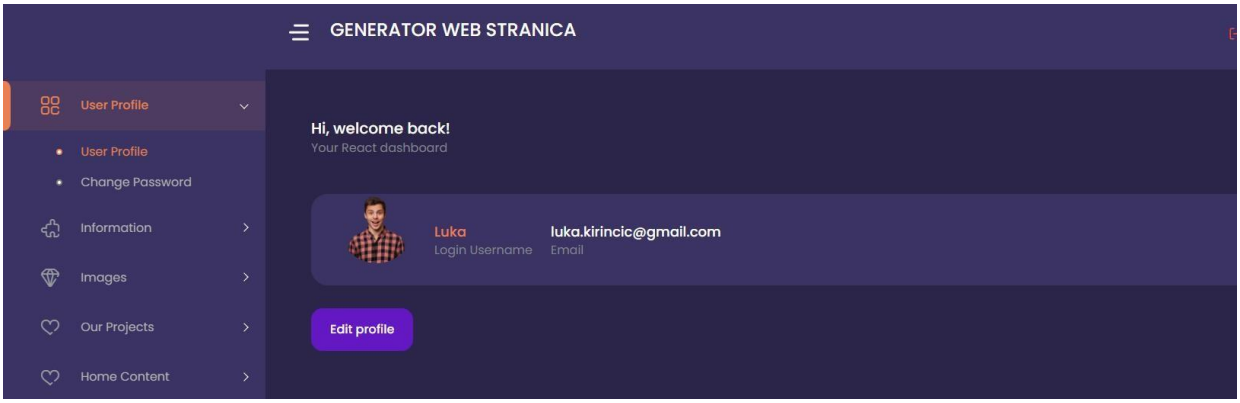
Slika 3.15 Ekran prikaza sadržaja podnožja stranice

Klikom na *Pregledaj web stranicu*, moguće je vidjeti kako izgleda web stranica s proizvoljno nadodanim elementima te će o tome i njezinom preuzimanju riječ biti nešto kasnije.

3.8 Uređivanje profila

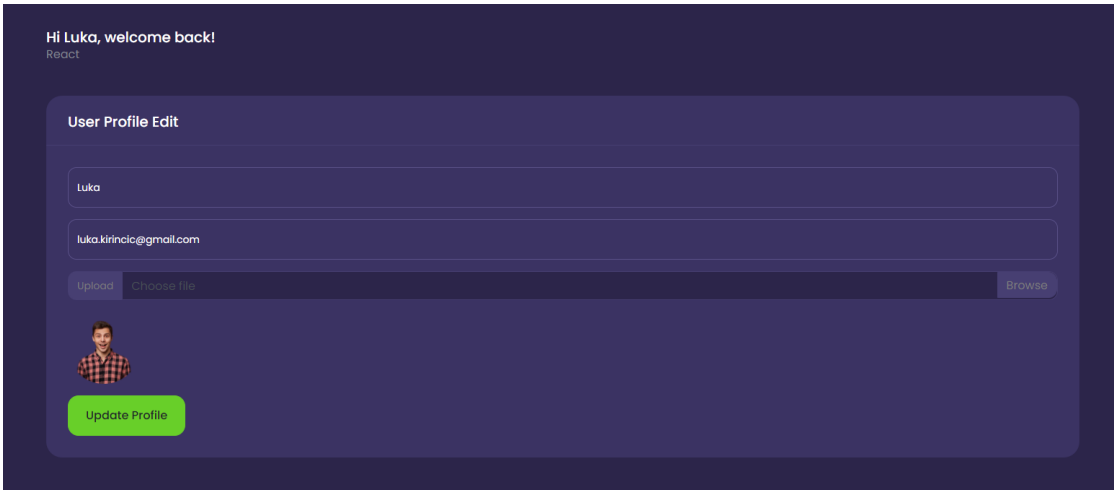
U bočnom izborniku, osim svih navedenih opcija, nalazi se i opcija za uređivanje profila. Klikom na *User profile* otvara se padajući izbornik s dvije opcije: *User profile* i *Change password*.

Na prozoru vezanom za profil korisnika vidljivo je korisničko ime, e-mail adresa te slika profila (slika 3.16).



Slika 3.16 Ekran korisničkog profila

Navedene informacije vezano za profil moguće je izmijeniti klikom na *Edit profile*. Tada se otvara prozor vidljiv na slici 3.17.



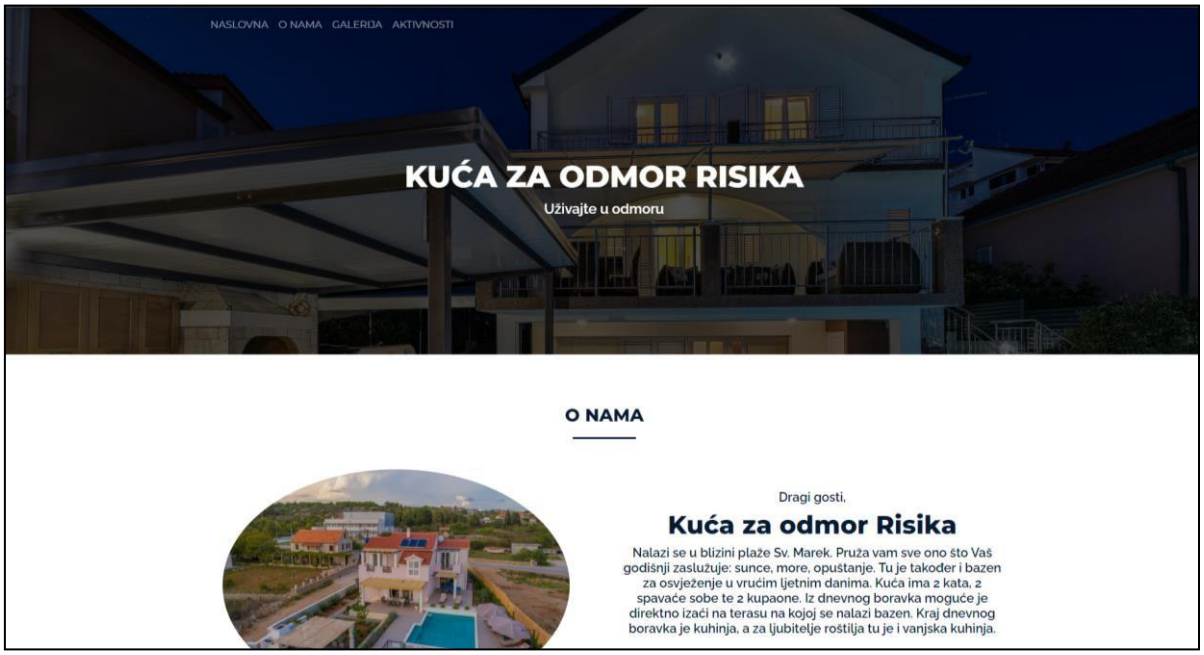
Slika 3.17 Ekran za uređivanje korisničkog profila

U prozoru za izmjenu lozinke moguće je i učiniti istoimenu radnju, a potrebno je unijeti trenutku lozinku te novu lozinku dva puta (slika 3.18).

Slika 3.18 Ekran za izmjenu lozinke

3.9 Pregled web stranice

Kao što je vidljivo na slici 3.15, nakon dodavanja sadržaja podnožja vidljiv je gumb te klikom na njega moguće je pregledati web stranicu i sadržaj koji je dodan (slika 3.19).

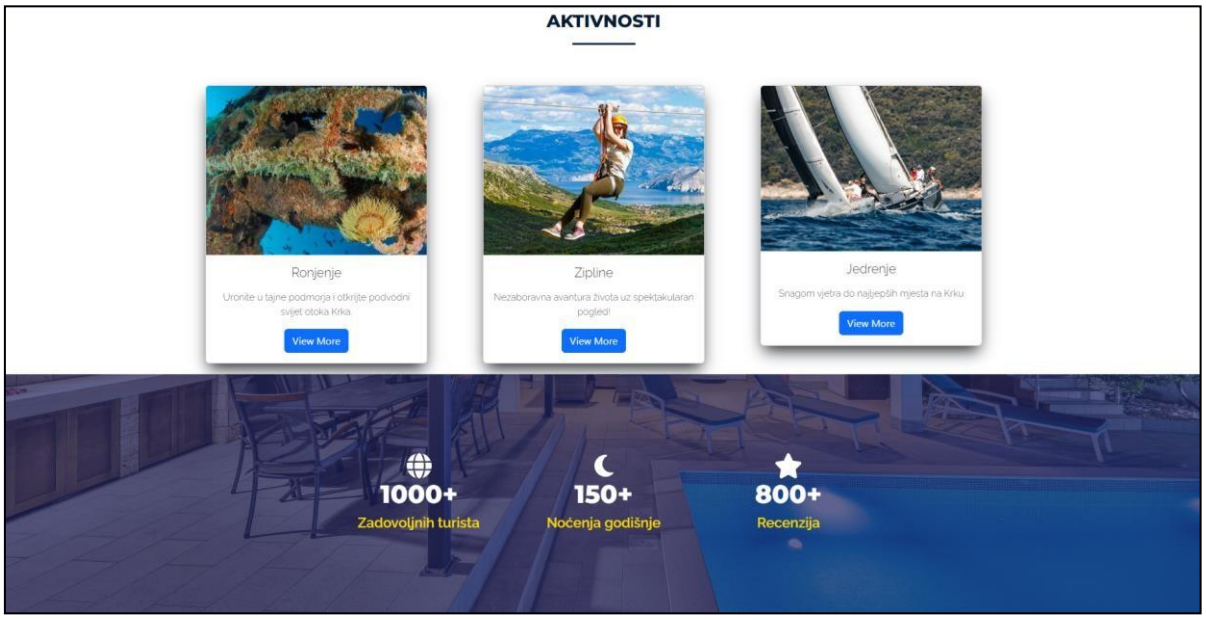


Slika 3.19 Pregled naslovne stranice

Na pregledu web stranice prvo je vidljiva naslovna stranica te se na njezinom vrhu nalaze se dva elementa koja je korisnik imao mogućnost dodati: naslov i podnaslov. Ti elementi su dodani putem izbornika za dodavanje sadržaja naslovne

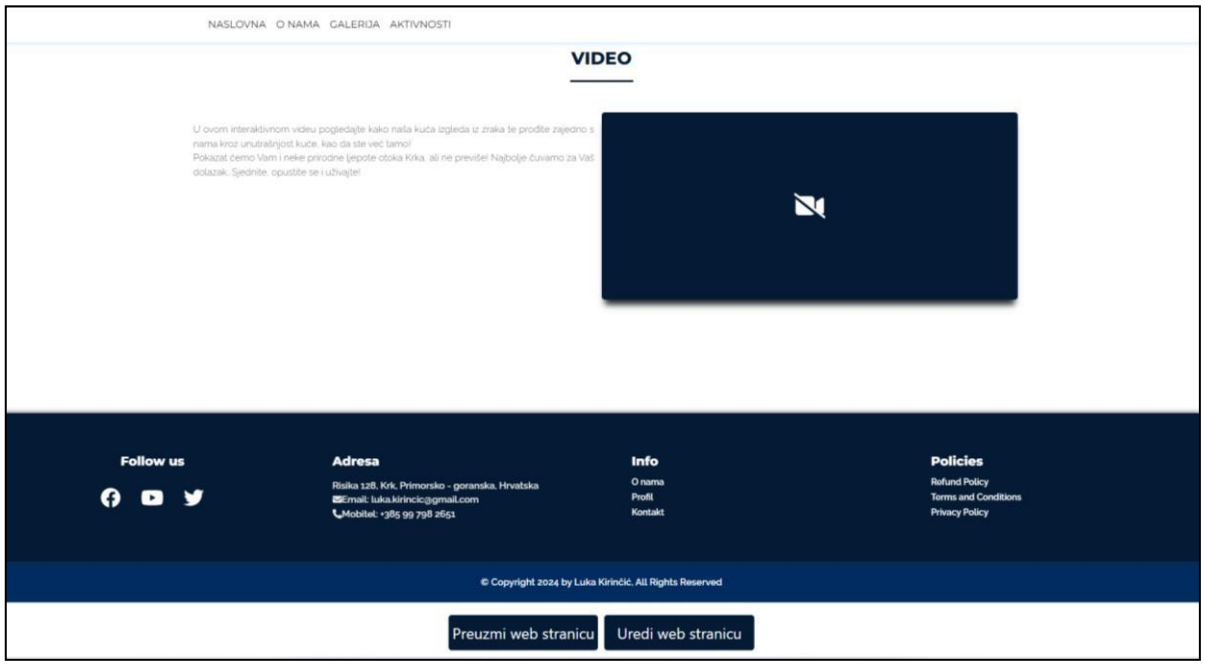
stranice. Ostalo je riječ o statičkim elementima koje je također moguće implementirati na način da se omogući korisniku dodavanje i tih elemenata. Na vrhu naslovne stranice nalazi se navigacijsko okno putem kojeg možemo pregledati i ostale rute na ovoj web stranici.

Listanjem daljnjeg sadržaja naslovne stranice vidljivi su projekti te sažetak (slika 3.20). Projekte je također korisnik imao mogućnost samostalno dodati, no svi dodani projekti vidljivi su klikom na Aktivnosti u navigacijskom oknu. Naslovna stranica sadrži isključivo prve 3 aktivnosti koje je korisnik dodao. Što se tiče dijela naslovne stranice s brojem turista, noćenja i recenzija riječ je o dijelu stranice koje služi kako bi naslovna stranica ostvarila bolji dojam na osobu koja ju pregledava, no moguće je omogućiti i ručno dodavanje tih elemenata.



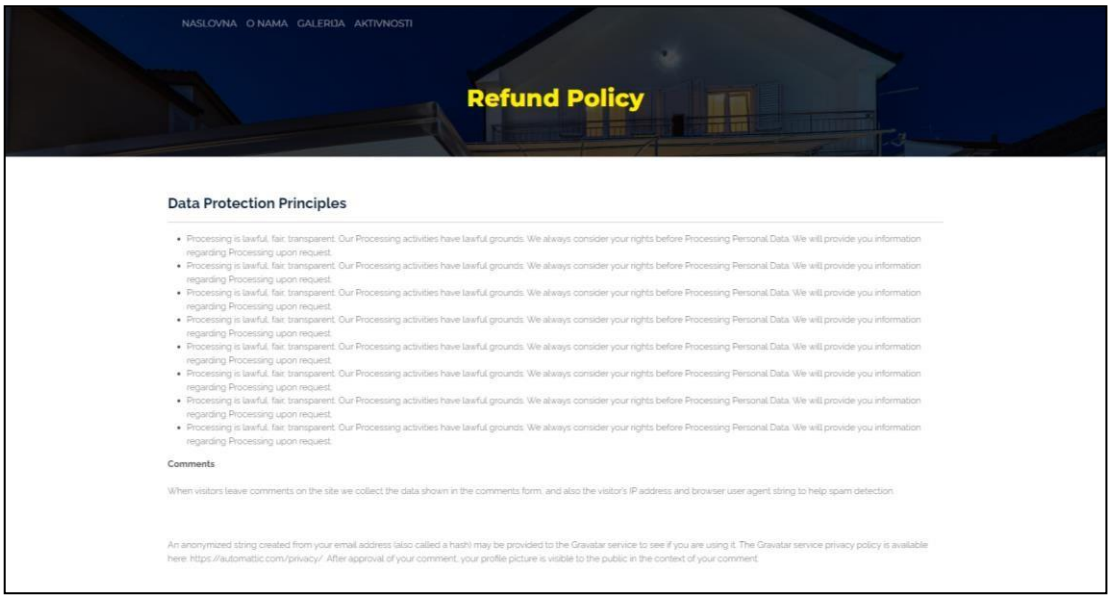
Slika 3.20 Pregled naslovne stranice

Ispod tog dijela naslovne stranice nalaze se isključivo elementi koje je korisnik samostalno dodao. Riječ je o videu, opisu videa te zaglavlju (slika 3.21). Također, na slici je vidljivo i da je implementiran način kako bi navigacijsko okno bilo vidljivo i kada se lista stranicom, a ne samo na vrhu stranice. Na dnu stranice vidljivi su i gumbi za preuzimanje odnosno uređivanje no o njima će više biti riječ u daljnjim dijelovima rada.



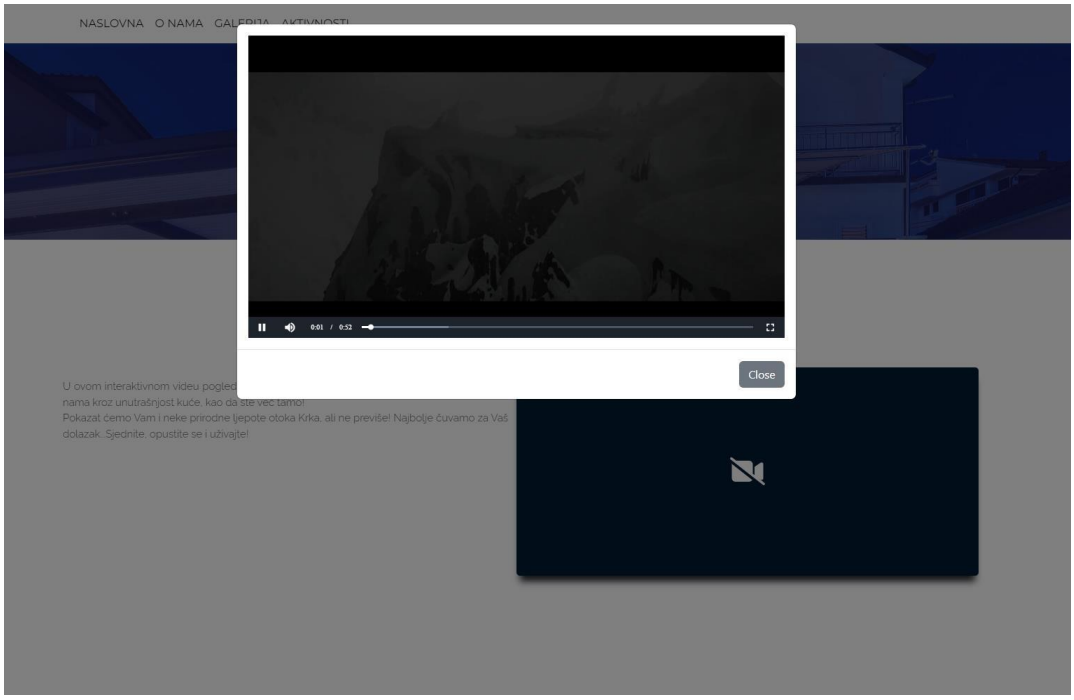
Slika 3.21 Pregled naslovne stranice (2)

Ispod videa riječ je o elementima koje je korisnik dodao u izborniku za sadržaj zaglavlja. Tu su dodane vlastite poveznice na društvene mreže, adresa, e-mail adresa te kontakt telefon (slika 3.21). Putem zaglavlja pristupa se i informacijama koje je korisnik unio te se nalaze u desnom donjem kutu na slici 3.21, a to su: *Refund Policy*, *Terms and Conditions* te *Privacy Policy* (slika 3.22).



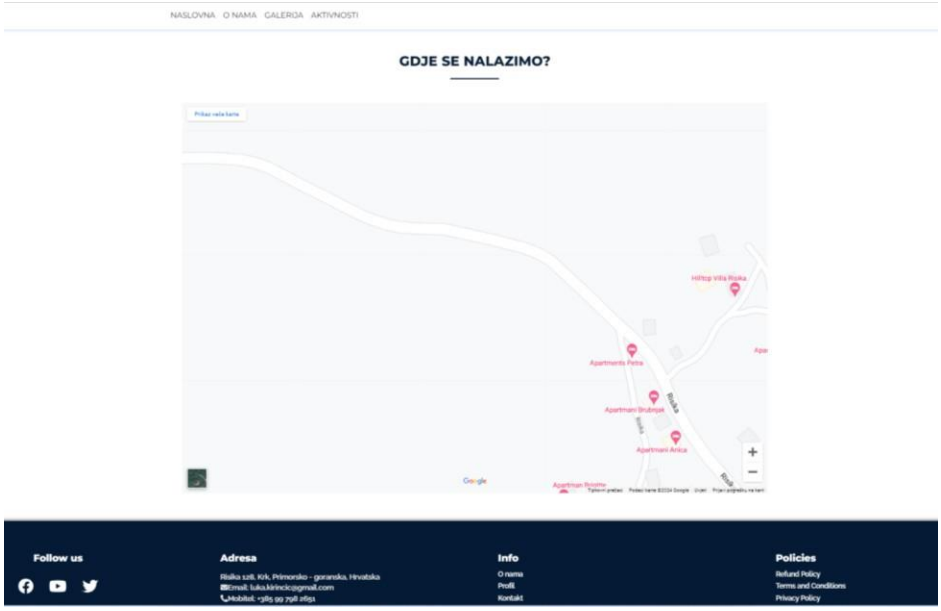
Slika 3.22 Refund Policy stranica

Korisnik je sam odabrao opis videa prikladan tome što se nalazi na videu, a u istom redu nalazi se i video za koji je korisnik priložio pripadajuću vezu stoga je video i moguće pokrenuti na web stranici (slika 3.23).



Slika 3.23 Prozor videa

Na stranici *O nama* nalazi se lokacija na karti te zaglavlje isto kao i na naslovnoj stranici (slika 3.24).



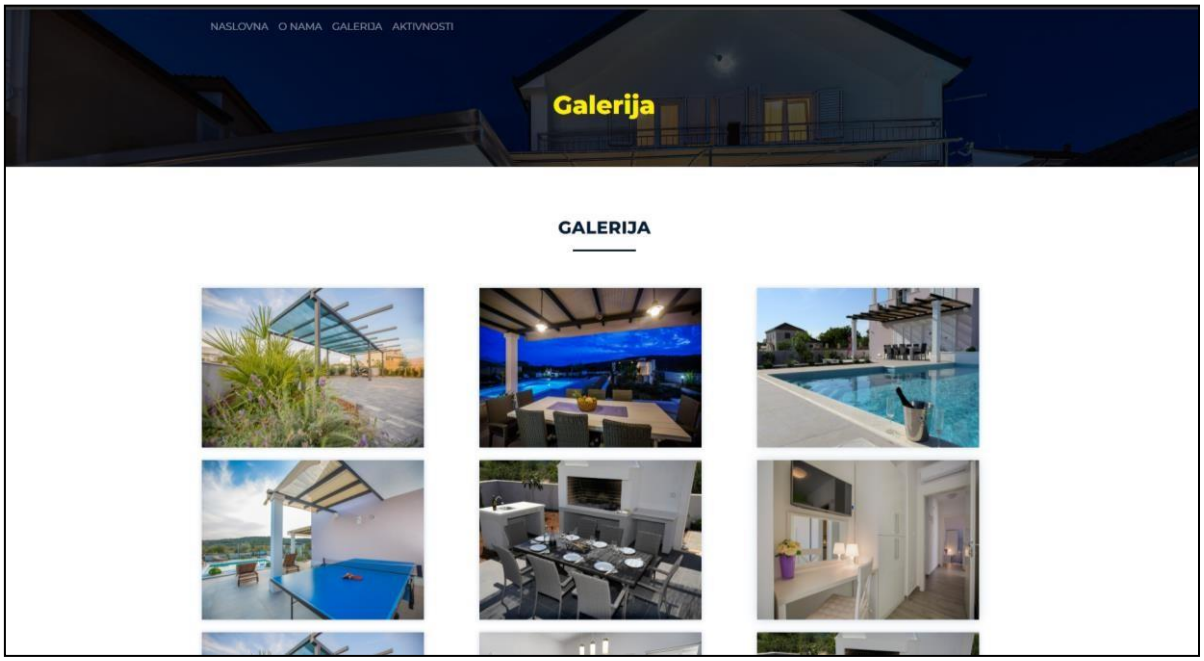
Slika 3.24 Pregled O nama stranice

Kako bi Google Karte bile implementirane, može se koristiti i *React* paket, no onda u preuzetoj *.html* datoteci karta ne bi bila vidljiva. Kako bi i to bilo omogućeno potrebno je koristiti *iframe* element u kodu vidljiv na slici 3.25. [10]

```
<iframe
    src="https://www.google.com/maps/embed?pb=!1m14!1m12!1m3!1d582.7293294989942!2d14.6424640347
    style={style}
    loading="lazy"
    title="Google Map"
></iframe>
```

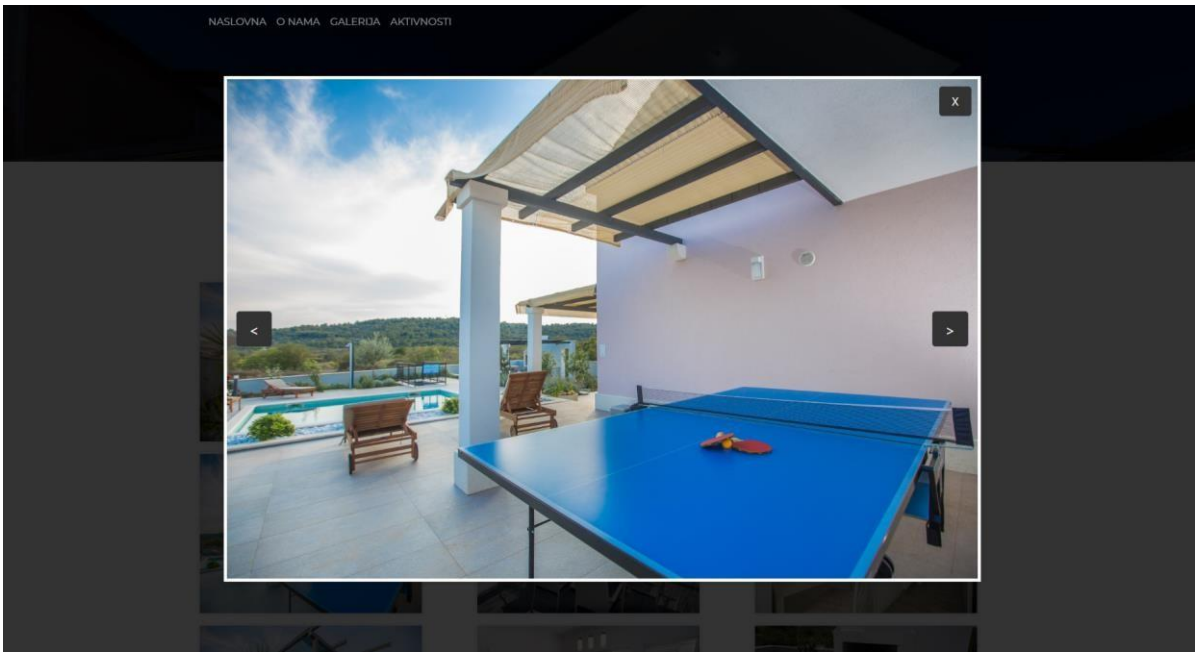
Slika 3.25 Korištenje iframe elementa unutar .html datoteke

Iduća stranica je Galerija te se na toj stranici nalaze sve slike te zaglavlje. Na ovoj stranici vidljive su sve slike koje je dodao korisnik i to upravo onim redoslijedom kojim su dodane (slika 3.26). Kada je riječ o zaglavlju, istoj je dizajna i sadržaja kao i na prethodnim stranicama.



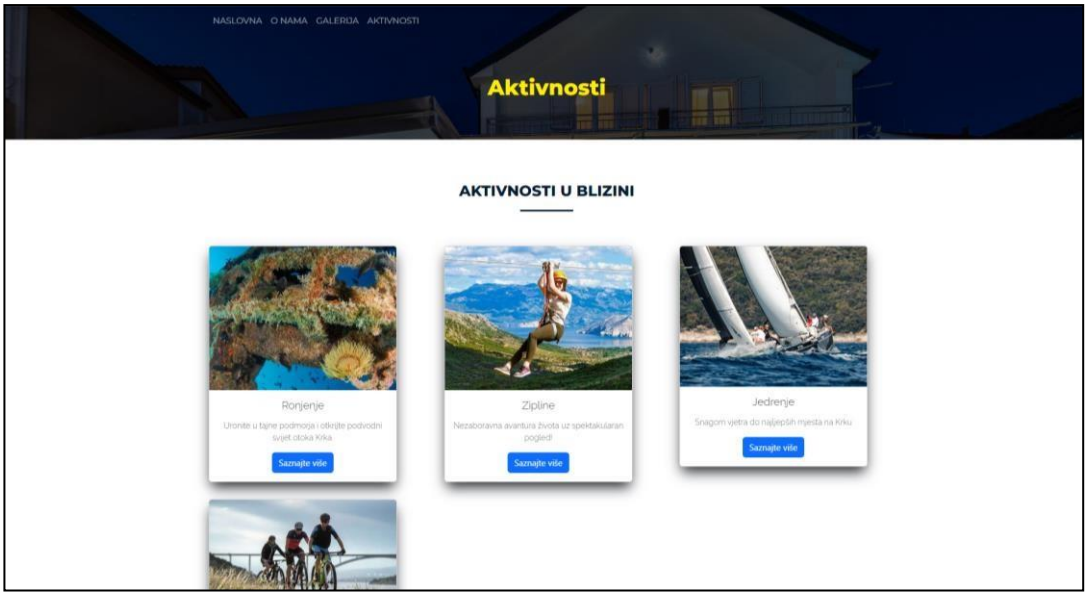
Slika 3.26 Pregled galerije

Kako bi pregled slika bio kvalitetniji i pregledniji, u kodu je implementirana i *Lightbox Javascript* datoteka te je izgled toga vidljiv na slici 3.27. Slike je moguće pregledavati i unutar *Lightboxa* pomoću strelica, a *Lightbox* se napušta klikom na *X* u gornjem desnom kutu. [11] [12]



Slika 3.27 Lightbox

Iduća, a ujedno i posljednja stranica je stranica *Aktivnosti*. Kao što je već spomenuto u prethodnom dijelu rada, ova stranica je namijenjena svim aktivnostima kojima je moguće pristupiti u blizini objekta za iznajmljivanje. Na stranici su vidljive sve aktivnosti koje je korisnik dodao te zaglavlje ekvivalentno onome na prethodnim stranicama (slika 3.28). Aktivnosti su prikazane u obliku kartica, a korisnik je samostalno dodao sliku, naslov, podnaslov te vezu za detaljnije informacije. Kada se na kartici klikne na gumb *Saznajte više* otvara se upravo web stranica za koju je korisnik učitao vezu.



Slika 3.28 Pregled Aktivnosti

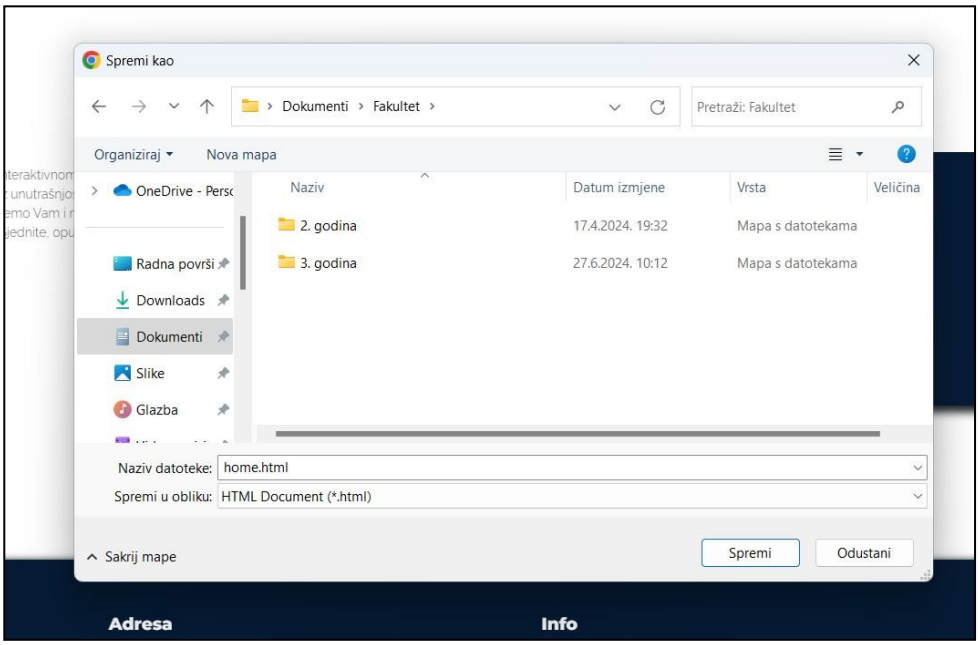
3.10 Preuzimanje web stranica

Na dnu svake od spomenutih stranica nalaze se 2 gumba: *Preuzmi web stranicu* i *Uredi web stranicu* (slika 3.29). Ukoliko korisnik nije zadovoljan sadržajem kojim je dodao, moguće je vratiti se nazad na stranicu putem koje se dodaje sav sadržaj te tamo unijeti željene izmjene.

Slika 3.29 Preuzmi i Uredi gumbi

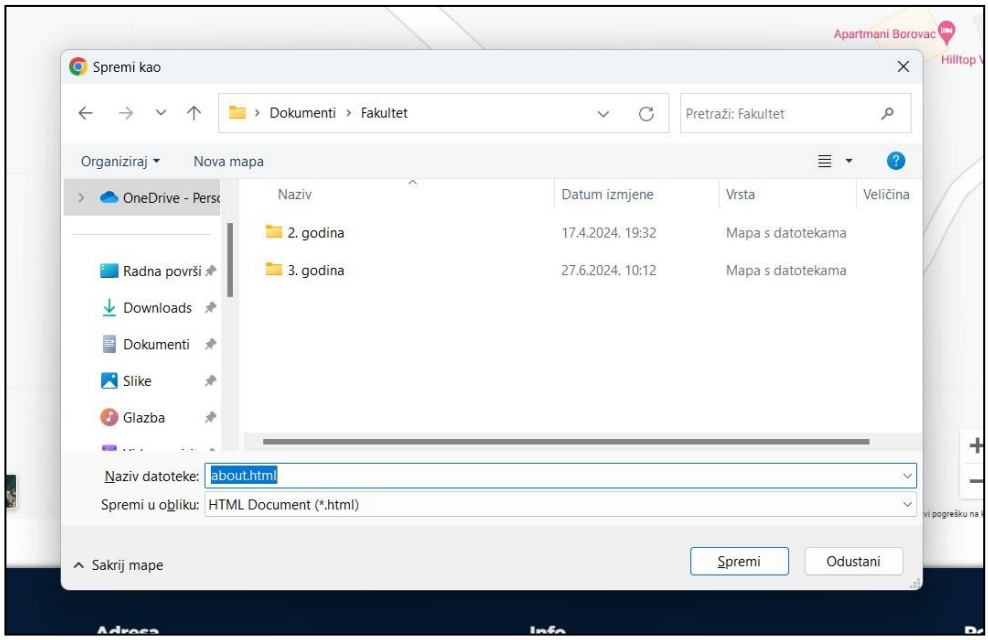
Nije bitan redoslijed kojim se preuzimaju *.html* datoteke, već je bitno samo da se nakon preuzimanja sve datoteke spreme u istu mapu, kako bi i u *.html* datotekama bilo moguće koristiti navigacijsko okno za pristup drugim *.html* datotekama.

Kada se preuzima naslovna stranica, otvara se prozor za preuzimanje te se uvijek preuzima datoteka naziva *home.html,* kao što je vidljivo na slici 3.30.



Slika 3.30 Preuzimanje Naslovne stranice

Nakon preuzimanja naslovne stranice, klikom na *O nama* u navigacijskom oknu otvara se istoimena stranica te je moguće i nju preuzeti. Ona se preuzima pod nazivom *about.html* (slika 3.31).



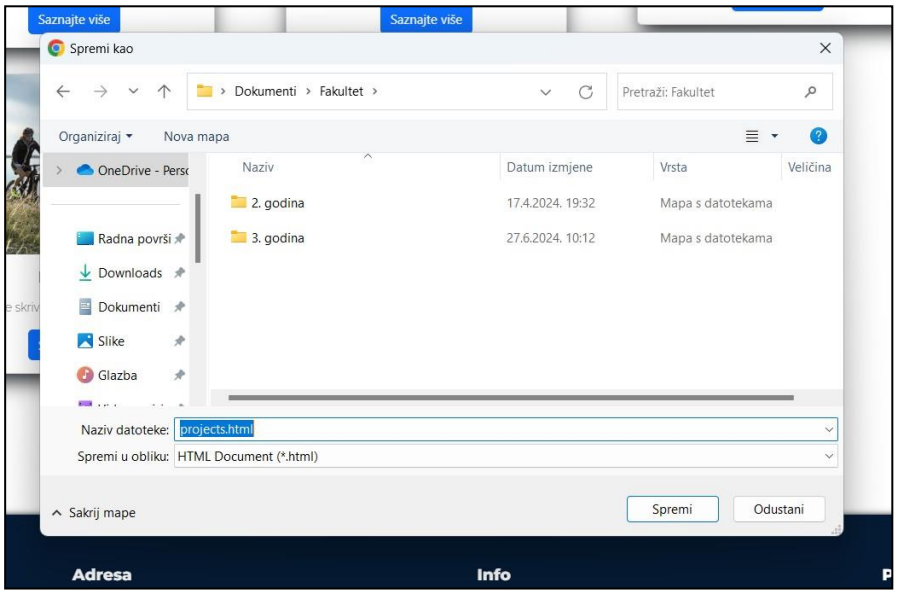
Slika 3.31 Preuzimanje stranice O nama

Kada se preuzima stranica *Galerija*, ne preuzima se samo *.html* datoteka već se preuzima *.zip* datoteka iz razloga što se u mapi osim *.html* datoteke nalaze i slike vidljive na web stranici. Kako bi i te slike bile vidljive kada se *.html* datoteka otvori na bilo kojem računalu, važno je da u istoj mapi s *.html* datotekom bude i mapa sa slikama (slika 3.32).

Slika 3.32 Sadržaj zip datoteke

U mapi *asset* nalazi se i *lightbox.js* datoteka kako bi *lightbox* bilo moguće koristiti i u preuzetoj *.html* datoteci.

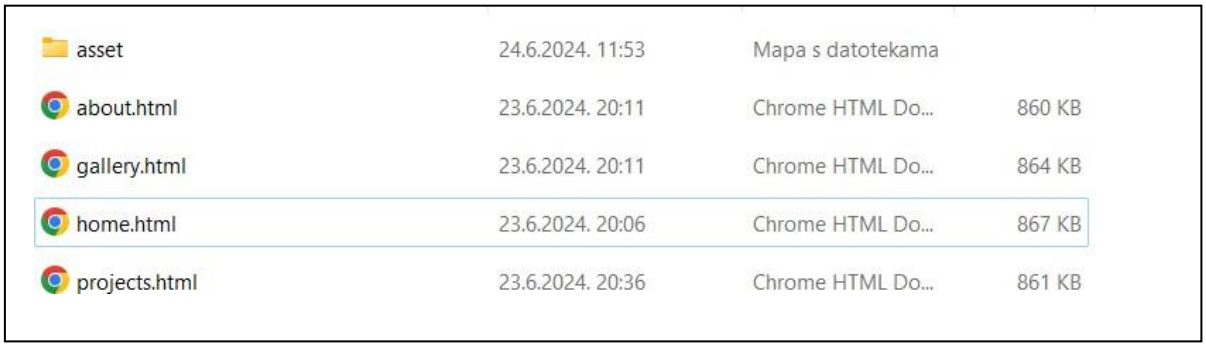
U konačnici još je potrebno preuzeti i stranicu Aktivnosti. To je moguće kada na istoimenoj stranici kliknemo gumb *Preuzmi web stranicu* koji se nalazi na dnu stranice. Tada se preuzima *projects.html* datoteka (slika 3.33).



Slika 3.33 Preuzimanje stranice Aktivnosti

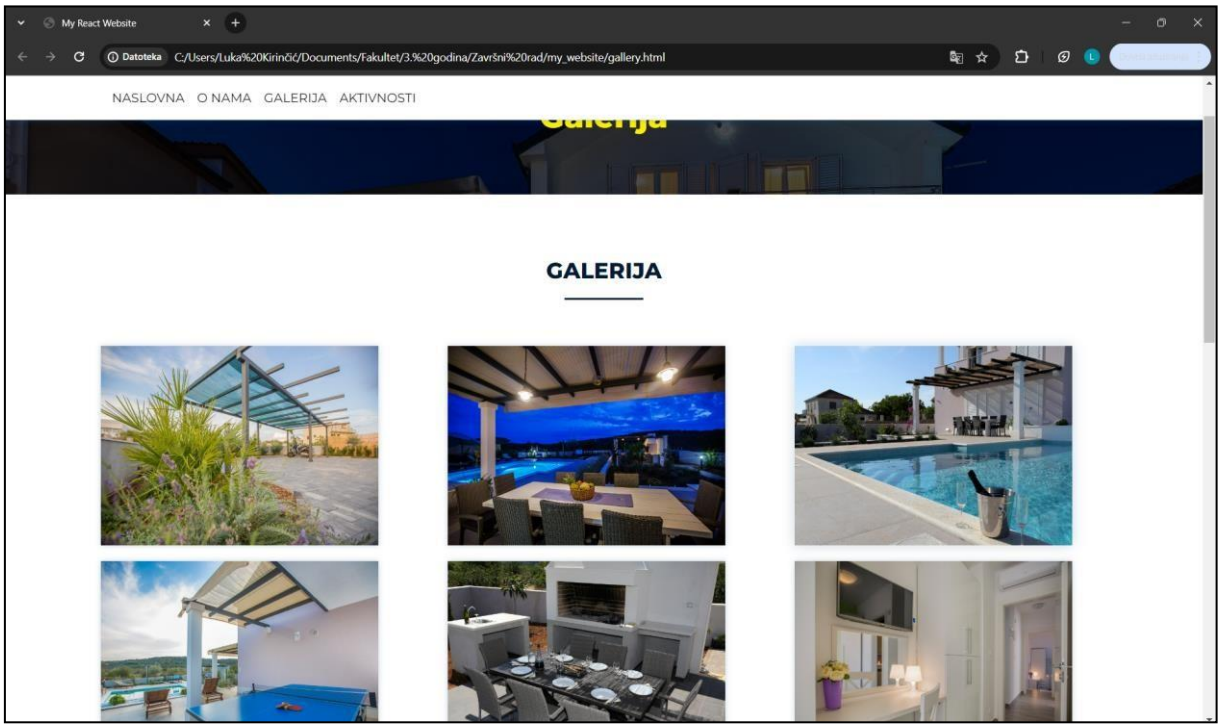
Kada su preuzete sve stranice u *.html* formatu, potrebno ih je smjestiti u zajedničku mapu. S obzirom da je galerija preuzeta u *.zip* formatu, najjednostavnije rješenje bi bilo raspakirati tu mapu, a preostale *home.html, about.html* te *projects.html* datoteke premjestiti unutar te mape. U kodu je implementirano da se ta mapa preuzeta u *.zip* formatu zove *my_website* te u protivnom neće biti moguće učitati slike na nekom

drugom računalu. Na taj način omogućilo bi se i korištenje navigacijskog okna za prebacivanje s jedne stranice na drugu. Sadržaj mape vidljiv je na slici 3.34.



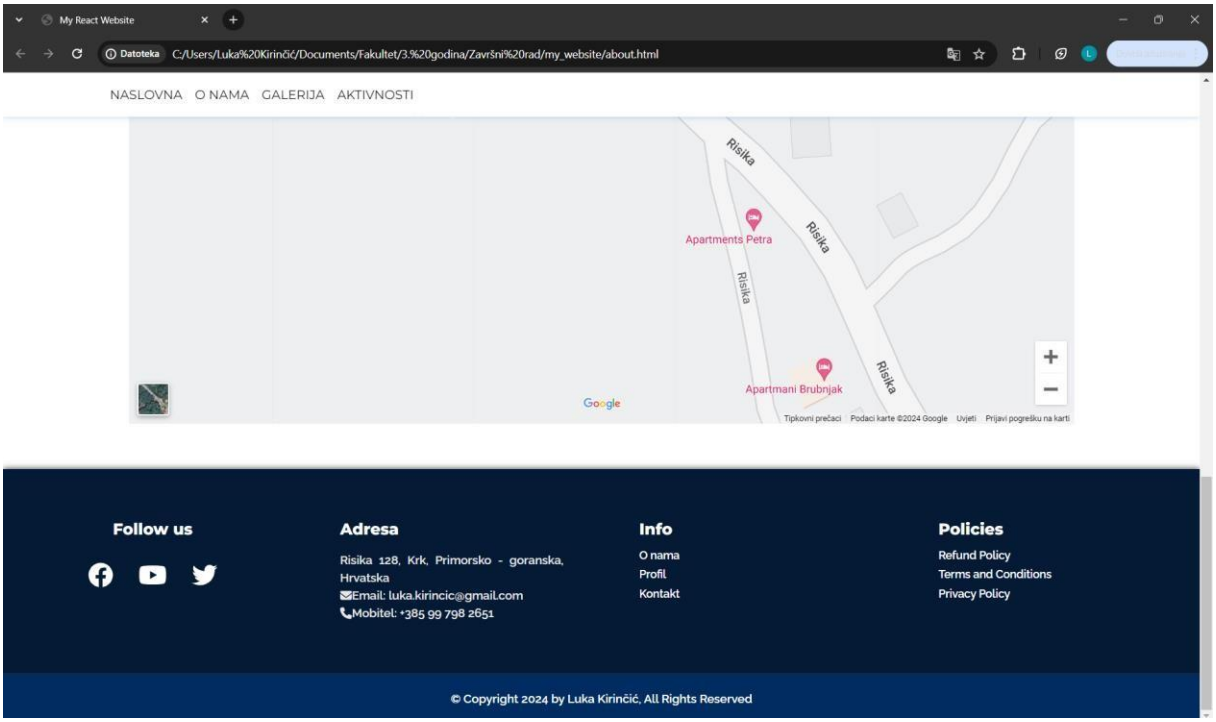
Slika 3.34 Sadržaj my_website mape

Kako bi korisnik bio siguran da je preuzeta *.html* datoteka jednaka upravo onoj web stranici može vrlo jednostavno otvoriti bilo koju od njih. Primjerice, usporedimo li *gallery.html* vidljiv na slici 3.35 sa slikom 3.26, jasno se da zaključiti kako je sav sadržaj ispravno preuzet te je uspješno generirana korisnička vlastita web stranica.



Slika 3.35 Pregled gallery.html

Jedina razlika u odnosu na web stranicu je to što u *.html* datotekama nisu vidljivi gumbi za preuzimanje i uređivanje iz razloga što je ta funkcionalnost tada nepotrebna, stoga je posljednji odlomak na web stranici isključen iz preuzimanja i prikazivanja u *.html* datoteci. Tu razliku moguće je uočiti uspoređujući dno bilo koje web stranice i njezine preuzete *.html* datoteke. Na slici 3.36 vidljivo je zaglavlje *about.html* datoteke te ga je moguće usporediti sa slikom 3.24 i uočiti navedenu razliku.



Slika 3.36 Pregled about.html datoteke

4 RAZVOJ I IMPLEMENTACIJA SPECIFIČKIH APLIKACIJSKIH FUNKCIONALNOSTI

Kako se ovaj rad odnosi upravo na preuzimanje web stranica, fokus ovog poglavlja biti će na objašnjavanju tog procesa preuzimanja te na preuzimanju slika s obzirom da je riječ o dvije funkcionalnosti koje su temelj ovog rada i bez kojih bi aplikacija bila u potpunosti nefunkcionalna.

4.1 Preuzimanje web stranice

Gumb za preuzimanje i njegove funkcionalnosti implementirani su kao odvojena komponenta naziva *Download.jsx* te su uključeni na sve postojeće stranice.

Na početku je potrebno postaviti inicijalno stanje gumba za preuzimanje na *false* kako bi gumb bio vidljiv te kako se ne bi dohvaćala trenutna ruta. Zatim se postavlja funkcija *„handleDownload“* u kojoj se stanje gumba mijenja na *true* kada se klikne gumb te se u tom trenutku dohvaća trenutna ruta i postavlja se inicijalni naziv datoteke za preuzimanje (slika 4.1).

```
class DownloadButton extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      clicked: false, // Inicijalno stanje button-a
    };
  }

  handleDownload = async () => {
    // Postavljanje stanja button-a na true kada se klikne
    this.setState({ clicked: true }, async () => {
      const route = window.location.pathname; // trenutna ruta
      let fileName = "my_website.html";

      //naziv file-a s obzirom na rutu gdje se nalazimo
      switch (route) {
        case "/":
          fileName = "home.html";
          break;
        case "/gallery":
          fileName = "gallery.html";
          break;
        case "/about":
          fileName = "about.html";
          break;
        case "/projects":
          fileName = "projects.html";
          break;
        default:
          break;
      }
```

Slika 4.1 Komponenta za preuzimanje

Inicijalni naziv datoteke se ne koristi već se mijenja ovisno o ruti na kojoj se korisnik nalazi kako bi nakon preuzimanja korisniku bilo jasno o kojoj je *.html* datoteci riječ. Sve opisano također je vidljivo u kodu na slici 4.1.

U idućim linijama nalazi se html sadržaj koji se želi preuzeti (slika 4.2). Prvo je potrebno uključiti datoteke koje su korištenje za stiliziranje te kompletan html sadržaj s web stranice. Nakon toga dodaju se sve željene skripte i *event liseneri* koji će se učitati kada se sav DOM i HTML sadržaj učita i parsira. U ovom slučaju se koriste JavaScript skripta za *lightbox* te svi *<a>* odnosno *<img>* elementi koji imaju *data-url* postavljen na *href*, odnosno *src*.

```
const htmlContent = `
  <!DOCTYPE html>
  <html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>My React Website</title>
    <link rel="stylesheet" href="./asset/css/custom.css">
    <link rel="stylesheet" href="./asset/css/responsive.css">
    <link rel="stylesheet" href="./asset/css/bootstrap.min.css">
    <link rel="stylesheet" href="./asset/css/lightbox.css">
  </head>
  <body>
  ${document.documentElement.innerHTML}
    <script src="./asset/js/lightbox.js"></script>
  </body>
  <script>
    document.addEventListener('DOMContentLoaded', function() {
        document.querySelectorAll('a[data-url]').forEach(function(element) {
            element.href = element.getAttribute('data-url');
        });
        document.querySelectorAll('img[data-url]').forEach(function(element) {
          element.src = element.getAttribute('data-url');
      });
    });
  </script>
  </html>
`;
```

Slika 4.2 Komponenta za preuzimanje

U nastavku, ukoliko je riječ o *gallery.html* datoteci, onda se generira *.zip* datoteka te se kreira nova instanca *JSZip* klase koja će se koristiti za generiranje ZIP datoteke. Osim *gallery.html* datoteke, pronalazi sve slike na stranici i kreira mapu u koju će se spremati slike. Kada su sve slike preuzete, generira se ZIP arhiva kao *blob* i pokreće se preuzimanje ZIP datoteke koristeći *saveAs* metodu. Ukoliko nije riječ o *gallery.html*,

kreira se *blob* objekt te se pokreće preuzimanje samo odgovarajuće *html* datoteke (slika 4.3).

```
if (fileName === "gallery.html") {
  const zip = new JSZip();
  zip.file(fileName, htmlContent);

  const imgElements = document.querySelectorAll("img");
  const imagesFolder = zip.folder("asset/images");

  const imagePromises = [];

  zip.file("asset/js/lightbox.js", await this.fetchFile("./asset/js/lightbox.js"));
  zip.file("asset/css/lightbox.css", await this.fetchFile("./asset/css/lightbox.css"));

  for (const imageElement of imgElements) {
    const imageId = imageElement.src.split("/").pop();
    const imgPromise = this.fetchImage(
      `http://127.0.0.1:8000/api/upload/service/${imageId}`,
    ).then((imgFile) => imagesFolder.file(imageId, imgFile));
    imagePromises.push(imgPromise);
  }

  await Promise.all(imagePromises);

  zip.generateAsync({ type: "blob" }).then((content) => {
    saveAs(content, "my_website.zip");
  });
}

else {
  const blob = new Blob([htmlContent], { type: "text/html;charset=utf-8" });
  saveAs(blob, fileName);
}
});
};
```

Slika 4.3 Komponenta za preuzimanje

U konačnici, stranica se preuzima klikom na gumb Preuzmi web stranicu te se tada poziva metoda *„handleDownload“*. Taj dio je obuhvaćen *if* petljom te se prikazuje samo i isključivo dok gumb za preuzimanje nije kliknut.

Kao što je vidljivo na prethodnoj slici, kada se preuzima *.zip* datoteka, potrebno je i dohvatiti datoteke *lightbox.js* i *lightbox.css* kako bi se omogućilo korištenje *lightboxa* i u preuzetoj *.html* datoteci. Te datoteke se dohvaćaju pozivanjem funkcije *„fetchFile“* koja je vidljiva na slici 4.4.

Slika 4.4 Funkcija za dohvaćanje datoteka

4.2 Preuzimanje slika

Iduća funkcionalnost koja je također jedna od temeljnih za funkcioniranje ovog projekta je preuzimanje slika zajedno s *.html* datotekom. Kako bi slike uopće bile učitane s HTML sadržajem, potrebna je njihova apsolutna putanja. No kako bi te slike bile vidljive na bilo kojem uređaju, potrebno ih je preuzeti i spremiti u mapu zajedno s *.html* datotekom. [13] Za potrebe ovog rada slike se preuzimaju zajedno s *gallery.html* datotekom koja i sadrži najviše slika.

Na poslužiteljskoj strani, implementirana je metoda *„getImage“* vidljiva na slici 4.5. Na početku se određuje putanja do slike spajanjem direktorija *upload/service/* s nazivom slike *$imageId* koristeći funkciju *„public_path“*, koja vraća apsolutnu putanju do direktorija *public*. Zatim se provjerava postoji li datoteka na određenoj putanji koristeći *File::exists*. Ako datoteka ne postoji, vraća se JSON odgovor s porukom o grešci i statusnim kodom 404 (*Not Found*). Ako datoteka postoji, njen sadržaj se učitava koristeći *File::get*. MIME tip (npr. *image/jpeg*, *image/png*) se određuje pomoću *File::mimeType*. U konačnici, sadržaj slike se vraća kao HTTP odgovor s statusnim kodom 200 (OK) i odgovarajućim *Content-Type* zaglavljem kako bi preglednik znao kako prikazati sliku.

```php
public function getImage($imageId)
{
    $imagePath = public_path('upload/service/' . $imageId);

    if (!File::exists($imagePath)) {
        return response()->json(['error' => 'Image not found'], 404);
    }

    $image = File::get($imagePath);

    $mimeType = File::mimeType($imagePath);

    return Response::make($image, 200, ['Content-Type' => $mimeType]);
}
```

Slika 4.5 Dohvaćanje slike

Također, na poslužiteljskoj strani kako bi se omogućila komunikacija s klijentskom stranom, potrebno je definirati API rutu za dohvaćanje slika vidljivu na slici 4.6. [3]

```php
Route::get('/upload/service/{imageId}', [ServiceController::class, 'getImage']);
```

Slika 4.6 API ruta

Na klijentskoj strani, implementira se funkcija *„fetchImage“* za dohvaćanje slika s određenih URL-ova (slika 4.7). Na početku se koristi *fetch* API za slanje HTTP zahtjeva na navedeni *imageUrl*. Budući da je *fetch* asinkrona funkcija, koristi se *await* kako bi se čekalo da zahtjev završi. Ako je odgovor uspješan (*response.ok* je *true*), odgovor se pretvara u *Blob* koji predstavlja binarne podatke slike. Ako zahtjev nije uspješan, ispisuje se greška u konzolu.

```javascript
async fetchImage(imageUrl) {
  const response = await fetch(imageUrl);
  if (response.ok) {
    return await response.blob();
  } else {
    console.error();
  }
}
```

Slika 4.7 Funkcija za dohvaćanje slika

U konačnici, stranica se preuzima klikom na gumb Preuzmi web stranicu te se tada poziva metoda *„handleDownload“*. Taj dio je obuhvaćen *if* petljom te se prikazuje samo i isključivo dok gumb za preuzimanje nije kliknut.

Također, na klijentskoj strani funkcija *„fetchImage“* se poziva za dohvaćanje slike putem API poziva definiranog na poslužiteljskoj strani (slika 4.8).

```
for (const imageElement of imgElements) {
  const imageId = imageElement.src.split("/").pop();
  const imgPromise = this.fetchImage(
    `http://127.0.0.1:8000/api/upload/service/${imageId}`,
  ).then((imgFile) => imagesFolder.file(imageId, imgFile));
  imagePromises.push(imgPromise);
}
```

Slika 4.8 Dohvaćanje putem API poziva

5 ZAKLJUČAK

Projekt izrade web stranice za automatsko generiranje web stranica za apartmane predstavlja uspješnu integraciju Reacta za klijentsku stranu i Laravela za poslužiteljsku stranu, pružajući korisnicima moderno iskustvo pregleda smještaja. React, s naglaskom na arhitekturi zasnovanoj na komponentama, omogućuje modularan i efikasan razvoj korisničkog sučelja. Korištenje *React Bootstrap* knjižnice pridonijelo je postizanju responzivnog dizajna, dok su ključne komponente poput *Container*, *Row* i *Col* omogućile organiziranje elemenata na stranici na sustavan način. Laravel, kao *backend framework*, pruža stabilnu infrastrukturu za upravljanje podacima, rute i kontrolere.

Unatoč postignućima i funkcionalnostima, uočeni su određeni aspekti koji zahtijevaju dodatnu pažnju i nadogradnju kako bi se poboljšalo ukupno korisničko iskustvo. Poželjno bi bilo kada bi se sve web stranice mogle preuzimati zajedno te bi se time korisniku još dodatno olakšalo korištenje ove aplikacije. Najveći problem u tome predstavlja dohvaćanje React sadržaja s ostalih ruta te realizacija toga izlazi iz uobičajene domene znanja programera. [14] Također, u okviru ovog projekta je okvirno pokazan princip na kojem ova aplikacija radi, no nisu svi elementi dostupni korisniku za dodavanje već su statički dodani, ali tu se ipak radi o itekako jednostavnijoj korekciji u odnosu na prethodno spomenuti nedostatak.

Nadogradnjom tih aspekata, moguće je dodatno unaprijediti performanse i korisničko iskustvo. Ukupno gledano, projekt predstavlja uspješnu realizaciju web stranice, a nadogradnja može doprinijeti daljnjem poboljšanju funkcionalnosti i interaktivnost.

LITERATURA

INTERNET IZVOR

[1] »React instalacija verzija 18,« [Mrežno]. Available: https://react.dev/learn/installation. [Pokušaj pristupa 25. 3. 2024.].

[2] »Instalacija Laravela verzija 10,« [Mrežno]. Available: https://laravel.com/docs/10.x/installation. [Pokušaj pristupa 14. 4. 2024.].

[3] M. Gulić, *Predavanje prezentacija - React + Laravel,* Rijeka, 2024.

[4] »React Bootstrap 1.5.2.,« [Mrežno]. Available: https://react-bootstrap.netlify.app/docs/components/accordion. [Pokušaj pristupa 25. 3. 2024.].

[5] »React Router Dom 5.2.0,« [Mrežno]. Available: https://v5.reactrouter.com/web/guides/quick-start. [Pokušaj pristupa 11. 4. 2024.].

[6] M. Gulić, *Predavanje prezentacija - Laravel dodatne funkcionalnosti,* Rijeka, 2024.

[7] M. Gulić, *Predavanje prezentacija - Laravel instalacija i CRUD,* Rijeka, 2024.

[8] »Laravel JetStream,« [Mrežno]. Available: https://jetstream.laravel.com/installation.html. [Pokušaj pristupa 30. 4. 2024.].

[9] »CSS,« [Mrežno]. Available: https://www.w3schools.com/css/. [Pokušaj pristupa 20. 4. 2024.].

[10] »Google maps implementacija,« [Mrežno]. Available: https://www.youtube.com/watch?v=ZbBaOKyqlUA&ab_channel=PRAROZ. [Pokušaj pristupa 5. 5. 2024.].

[11] »Lightbox,« [Mrežno]. Available: https://www.w3schools.com/howto/howto_js_lightbox.asp. [Pokušaj pristupa 14. 5. 2024.].

[12] »Lightbox,« [Mrežno]. Available: https://www.freecodecamp.org/news/how-to-create-a-lightbox-using-html-css-and-javascript/. [Pokušaj pristupa 14. 5. 2024.].

[13] »Stack Overflow,« [Mrežno]. Available: https://stackoverflow.com/questions/43871637/no-access-control-allow-origin-header-is-present-on-the-requested-resource-whe. [Pokušaj pristupa 25. 5. 2024.].

[14] »StackOverflow,« [Mrežno]. Available: https://stackoverflow.com/questions/34870711/download-a-file-at-different-location-

using-html5. [Pokušaj pristupa 11. 5. 2024.].

[15] »FontAwesome,« [Mrežno]. Available: https://fontawesome.com/icons. [Pokušaj pristupa 11. 4. 2024.].

[16] »React Slick 0.28.1,« [Mrežno]. Available: https://www.npmjs.com/package/react-slick,. [Pokušaj pristupa 11. 4. 2024.].

[17] »Aktivnosti,« [Mrežno]. Available: https://www.krk-outdoor.hr/. [Pokušaj pristupa 17. 4. 2024.].

POPIS SLIKA

SAŽETAK

U ovom radu razvijena je web aplikacija koja omogućuje registriranim korisnicima samostalno generiranje web stranica za kuće ili apartmane koje iznajmljuju. Korisnicima je omogućeno postavljanje početne slike, teksta, podataka o apartmanu, dodavanje slika te kontaktnih informacija. Korisnici se mogu registrirati i prijaviti na platformu radi pristupa alatima za generiranje web stranica. Nakon uređivanja, korisnici mogu pregledati svoje web stranice u stvarnom vremenu i preuzeti gotov HTML/CSS/JavaScript kod na svoje računalo. Za razvoj poslužiteljskog dijela web aplikacije korišten je Laravel radni okvir uz MySQL sustav za upravljanje bazama podataka. Za razvoj klijentskog dijela aplikacije korištena je React JavaScript knjižica za razvoj korisničkog sučelja uz učinkovito renderiranje aplikacije na uređajima s različitim veličinama zaslona.

Ključne riječi: web, aplikacija, preuzimanje, uređivanje, React, Laravel, korisnik

ABSTRACT

In this work, a web application was developed that allows registered users to independently generate web pages for houses or apartments they rent. Users can set the initial image, text, information about the apartment, add images and contact information. Users can register and login to the platform to access the website generation tools. After editing, users can preview their web pages in real time and download the finished HTML/CSS/JavaScript code to their computer. For the development of the server part of the web application, the Laravel framework was used along with the MySQL database management system. To develop the client part of the application, the React JavaScript library was used to develop the user interface with efficient rendering of the application on devices with different screen sizes.

Keywords: web, application, downloading, editing, React, Laravel, user