Web sustav za upravljanje ponudom i potražnjom turističkih smještajnih jedinica

Lulić, Rok

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Sveučilišni prijediplomski studij računarstva

Završni rad

WEB SUSTAV ZA UPRAVLJANJE PONUDOM I POTRAŽNJOM TURISTIČKIH SMJEŠTAJNIH JEDINICA

Rijeka, rujan 2023.

Rok Lulić
0069089883

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Sveučilišni prijediplomski studij računarstva

Završni rad

WEB SUSTAV ZA UPRAVLJANJE PONUDOM I POTRAŽNJOM TURISTIČKIH SMJEŠTAJNIH JEDINICA

Mentor: Izv. prof. dr. sc. Sandi Ljubić

Rijeka, rujan 2023.

Rok Lulić
0069089883

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA ZAVRŠNE ISPITE

Rijeka, 21. srpnja 2023.

Zavod: **Zavod za računarstvo**
Predmet: **Baze podataka**
Grana: **2.09.01 arhitektura računalnih sustava**

ZADATAK ZA ZAVRŠNI RAD

Pristupnik: **Rok Lulić (0069089883)**
Studij: Sveučilišni prijediplomski studij računarstva

Zadatak: **Web sustav za upravljanje ponudom i potražnjom turističkih smještajnih jedinica / A Web System for Tourist Accommodation Supply and Demand Management**

Opis zadatka:

U radu je potrebno implementirati Web sustav koji učinkovito spaja oglašavanje, pretraživanje i rezervaciju turističkih smještajnih jedinica. Sustav podrazumijeva dvije osnovne uloge za registrirane korisnike: iznajmljivač turističkog smještaja i turist. Po uzoru na postojeće slične sustave, definirati i implementirati odgovarajuće slučajeve korištenja za navedene uloge. Sustav treba, između ostaloga, podržavati višeparametarsko pretraživanje smještajnih jedinica te mogućnost komentiranja i ocjenjivanja. Posebnu pažnju valja usmjeriti na oblikovanje korisničkog sučelja, s ciljem implementacije rješenja koje je responzivno, intuitivno i jednostavno za koristiti.

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

Zadatak uručen pristupniku: 21. srpnja 2023.

Mentor:

Izv. prof. dr. sc. Sandi Ljubić

Predsjednik povjerenstva za završni ispit:

Prof. dr. sc. Miroslav Joler

Izjava o samostalnoj izradi rada

Izjavljujem da sam samostalno izradio ovaj rad.

Rijeka, rujan 2023.

_____

Rok Lulić

SADRŽAJ

POPIS SLIKA

POPIS ISJEČAKA KODA

1 UVOD

Glavni fokus ovog rada zasniva se na novim tehnologijama koje su drastično promijenile način razvoja web sustava, a naročito u industriji *online* rezervacija. Kroz primjer razvoja web sustava za upravljanje ponudom i potražnjom turističkog smještaja, predstavljeni su alati Next, Tailwind CSS i MongoDB koji se lako integriraju i jednostavno koriste.

Teško je procijeniti koliko je proces stvaranja web-baziranog sustava za kontrolu ponude i potražnje turističkih smještajnih jedinica pojednostavljen korištenjem navedenih tehnologija, no uloga ovog rada je ukazati na olakšanja u razvoju uzrokom korištenja tih tehnologija, kao i dati uvid u novi način razmišljanja pri razvoju web aplikacija.

2 POSTOJEĆI WEB SUSTAVI ZA PONUDU I POTRAŽNJU SMJEŠTAJNIH JEDINICA

Prije detaljnijih opisa funkcija implementiranog rješenja, važno je spomenuti postojeće web sustave koji su utrli put aplikacijama poput one predstavljene u ovom radu. Ove su platforme uspješno implementirale različite značajke vezane uz ponudu i potražnju smještajnih jedinica, služeći kao izvor inspiracije i postavljajući industrijske standarde.

Jedan od najznačajnijih primjera je Booking.com [1], svjetski lider u online rezervacijama smještaja. Booking.com nudi širok raspon mogućnosti smještaja, od hotela i apartmana do jedinstvenih boravaka, a sve je potkrijepljeno recenzijama i ocjenama korisnika. Ova je platforma revolucionirala način na koji ljudi rezerviraju smještaj, čineći je najboljim izborom za putnike diljem svijeta.

Airbnb [2] je još jedna istaknuta platforma u industriji dijeljenja smještaja. Za razliku od tradicionalnih hotela, Airbnb omogućuje pojedincima da putnicima iznajmljuju svoje domove ili slobodne sobe. Stvoren je novi model rezerviranja smještaja, stavljajući naglasak na personalizaciju i jedinstvena iskustva.

Expedia [3] je još jedna poznata platforma koja nudi širok raspon putničkih usluga, uključujući hotelske rezervacije. Expedijin sveobuhvatan pristup planiranju putovanja uključuje letove, hotele, najam automobila i više, služeći se putnicima koji traže udobnost i izbor.

Ove su platforme napravile značajan napredak u redefiniranju iskustva rezervacije smještaja, koristeći tehnologiju za povezivanje putnika s dostupnim opcijama smještaja. Iako svaka platforma ima svoje jedinstvene karakteristike i prednosti, sve one dijele zajednički cilj: pojednostaviti proces pronalaženja i rezervacije smještajnih jedinica.

U sljedećim odjeljcima demonstrirano je kako ovaj rad crpi inspiraciju u navedenim platformama dok uvodi svoje vlastite inovativne elemente. Testiranjem postojećih sustava, stječu se uvidi u evoluciju online rezervacija smještaja te se otkrivaju prilike za poboljšanje i diferencijaciju.

3 RAZVOJ WEB SUSTAVA ZA UPRAVLJANJE PONUDOM I POTRAŽNJOM TURISTIČKIH SMJEŠTAJNIH JEDINICA

U ovom opisuje se razvoj web rješenja za rješavanje zamršene dinamike upravljanja ponudom i potražnjom unutar industrije turističkog smještaja. Unutar ovog poglavlja istražit će se sljedeći ključni aspekti:

- **Tehnološki stog:** temeljne tehnologije koje podupiru implementirani web sustav, uz argumentaciju iza njihovog odabira i opisa kako doprinose ukupnoj funkcionalnosti.
- **Klijentska strana sustava:** fokus na korisničkom sučelju, uz raspravu o intuitivnom stvaranju vizualno privlačnih React komponenti te jednostavnosti postavljanja lokalnog web poslužitelja.
- **Poslužiteljska strana sustava:** Nadopunjavanje klijentske strane vitalna je uloga poslužitelja u upravljanju podacima, sigurnosti i komunikaciji. Opisuje se složenost postavljanja poslužitelja, integracija baze podataka, dizajn krajnje točke *API*-ja i shema baze podataka.

Ovo poglavlje pruža sveobuhvatan pregled tehničke osnove koja pokreće naš web sustav.

3.1 Tehnološki stog

Temelj svakog uspješnog projekta razvoja softvera leži u pažljivom odabiru i orkestraciji njegovih tehnoloških komponenti. Skup tehnologije odabran za predmetni web sustav temelj je na kojem se grade njegove mogućnosti, performanse i skalabilnost.

Ovo poglavlje opisuje odabrane alate i tehnologije, kao i razloge za njihovog odabira. Pomno ćemo ispitati obrazloženje i razmatranja koja su vodila odabir svake tehnologije, s ciljem pružanja sveobuhvatnog razumijevanja sinergija i koristi koje donose projektu u cjelini.

Kroz ovaj poglavlje pružaju se uvidi u strateške odluke koje su pridonijele robusnosti, učinkovitosti i prilagodljivosti web sustava. Također ćemo raspravljati o tome kako ove tehnologije rješavaju specifične potrebe upravljanja turističkim smještajem u današnjem dinamičnom digitalnom okruženju.

.3.1.1 React.js

Razvijen od strane Facebooka (sada Meta) 2013., React.js [4] je JavaScript biblioteka za stvaranje korisničkih sučelja u modernim web i mobilnim aplikacijama. Ova se biblioteka može pohvaliti arhitekturom orijentiranom na komponente koja potiče ponovnu upotrebu koda i modularnost. Iskorištava virtualni *DOM (Document Object Model)* za pojednostavljenje ažuriranja korisničkog sučelja, poboljšavajući performanse čak i u velikim aplikacijama.

Jedna od značajki Reacta je JSX, JavaScript-HTML fuzija koja poboljšava čitljivost koda i podržava dinamičke podatke i funkcionalnost. React- se lako integrira s raznim alatima i bibliotekama, što ga čini prikladnim za male projekte, ali i za složenije sustave.

React osnažuje komponente s dinamičkim unutarnjim stanjima, omogućujući automatska ažuriranja korisničkog sučelja nakon promjena stanja, potičući razvoj responzivnih aplikacija. To je prikazano u značajkama kao što su preklapanje tema (npr. svijetli ili tamni način). Nadalje, React se često bira za izradu *SPA (Single Page Applications)*, gdje se inicijalno učitava samo HTML, isporučujući dinamički sadržaj bez ponovnog učitavanja cijele stranice za poboljšano korisničko iskustvo.

Ono što React izdvaja je njegova ekspanzivna zajednica programera koji održavaju robustan ekosustav biblioteka i alata.

3.1.2 Next.js

Next.js [5] , najistaknutiji React okvir koji preporučuje sam React, predstavlja vodeći izbor u području web razvoja. Uvodi promjenu paradigme u komponente programiranja generirane na strani poslužitelja, čineći prikazivanje na strani poslužitelja (*SSR – Server Side Rendering*) pristupačnijim i učinkovitijim putem usmjeravanja temeljenog na datotekama.

Uz Next razvoj se eksponencijalno ubrzava zahvaljujući skupu alata bogatom značajkama koji unaprjeđuju temelj React. Okvir se posebno ističe u poboljšanju *SEO*-a (*Search Engine Optimization*), performansi i brzine, što se prvenstveno pripisuje njegovim *SSR* sposobnostima. Ova optimizacija osigurava brzo renderiranje sadržaja.

Next.js uvodi *HMR* (*Hot Module Reloading*), značajku koja mijenja sami razvoj web aplikacija, omogućujući neometanu primjenu izmjena modula, ažurirajući kod u pregledniku bez potrebe

ponovnog učitavanja cijele stranice. Ova fluidnost u izmjenama koda drastično poboljšava razvojno iskustvo.

Osim *SSR*-a, Next.js također se ističe u generiranju statičkih stranica (*SSG – Static Site Generation*), omogućujući stvaranje statičkih stranica koje se odmah poslužuju korisnicima putem mreža za isporuku sadržaja (*CDN – Content Delivery Network*). Ova se mogućnost pokazala neprocjenjivom za statične stranice poput blogova, budući da osigurava brže vrijeme učitavanja zbog blizine krajnjih korisnika.

Next-ove robusne mogućnosti usmjeravanja i *API* rute pojednostavljuju razvoj dinamičnih web stranica bogatih sadržajem, izdvajajući ga kao vodeći izbor u modernom web razvoju.

3.1.3 Tailwind CSS

Tailwind CSS [6], ultimativni CSS okvir, odlikuje se svojim pristupom usmjerenim na korisnika i uslužnim programima za stvaranje responzivnih i vizualno privlačnih korisničkih sučelja. Tailwind, za razliku od tradicionalnih CSS okvira, ubrzava dizajn korisničkog sučelja pružajući mnoge unaprijed definirane klase, omogućujući programerima da brzo stiliziraju komponente s preciznošću i dosljednošću.

Tailwind-ova opsežna biblioteka korisnih klasa, koja pokriva tipografiju, razmake, sheme boja i rasporede, jedna je od njegovih ključnih prednosti. Tailwind također promiče prilagodbu, dopuštajući programerima da prilagode svoj dizajn u širokom rasponu veličina zaslona i uređaja.

3.1.4 MongoDB

Popularna NoSQL baza podataka MongoDB [7], ističe se svojom prilagodljivošću i fleksibilnošću. Ova je fleksibilnost vrlo korisna, osobito u situacijama u kojima se vrste podataka i formati mogu mijenjati tijekom vremena. Zbog svoje prirode bez sheme omogućuje spremanje podataka bez određene strukture, što ga čini vrlo prilagodljivim promjenjivim zahtjevima za podacima i adekvatnim rješenjem za razne primjene.

Nadalje, MongoDB pruža horizontalnu skalabilnost, omogućujući aplikacijama učinkovito rukovanje rastućim količinama podataka i prometa. Njegov format podataka temeljen na dokumentima, koji koristi dokumente slične JSON-u, olakšava pohranjivanje i dohvaćanje

podataka, što ga čini najboljim izborom za programere koji rade na projektima s kompliciranim povezivanjem podataka ili se stalno mijenjaju zahtjevi za podacima.

3.1.5 Razlozi odabira navedenih tehnologija

Odabir spomenutih tehnologija svodi se na nekoliko ključnih razmatranja:

- **Učinkovitost i performanse**: React.js i Next.js olakšavaju stvaranje visoko responzivnih i učinkovitih korisničkih sučelja. SSR u Next-u poboljšava SEO i optimizira učitavanje stranica, što je ključno za platformu za rezervaciju smještaja.
- **Skalabilnost**: Arhitektura React.js i Next.js dopušta skalabilni razvoj, prilagođavajući potencijalna buduća proširenja i poboljšanja značajki.
- **Korisničko iskustvo**: Tailwind CSS omogućuje brzi razvoj korisnički prihvatljivih i vizualno privlačnih sučelja, pridonoseći pozitivnom korisničkom iskustvu.
- **Upravljanje podacima**: NoSQL priroda MongoDB-a pruža fleksibilnost za rukovanje različitim podacima o uslugama poslužitelja, istovremeno osiguravajući učinkovito dohvaćanje i upravljanje podacima.

Izabrani tehnološki stog služi kao čvrst temelj, usklađuje se sa zahtjevima web poslužitelja i pozicionira ga za uspjeh u konkurentskoj domeni upravljanja turističkim smještajem.

3.2 Klijentska strana sustava

U ovom poglavlju opisano je kako je postavljena klijentska strana implementiranog sustava. Prije svega, potrebno je obaviti instalaciju Next.js razvojnog okvira, koja se može preuzeti s izvorne web stranice [5].

Za stvaranje dinamičnih komponenti potrebno je dodati „/components“ direktorij unutar „/src“ direktorija repozitorija sustava, te u njemu stvoriti .jsx datoteku (kodni isječak 3.1). Unutar bilo koje vrste HTML oznake možemo dodati inačicu *className=" "*, čijim se dodavanjem poziva na korištenje Tailwind CSS-a.

```jsx
import React from "react";
const Komponenta = async () => {
  return <div className="text-body-18 text-stone-700">Hello World!</div>;
};

export default Komponenta;
```

Kodni isječak 3.1: *Primjer React.jsx komponente*

Postavljanje stranice iscrtane u pregledniku postiže se već samo stvaranjem jedne page.jsx datoteke, u kojoj se pozivaju komponente sustava (kodni isječak 3.2), te pokretanjem komande:

```
PS C:\Users\lulic\Desktop\zavrsni> npm run dev
```

u terminalu.

```jsx
import "../app/globals.css";
import CardSection from "../components/Hotel/CardSection";
import FeaturedDestinations from
"../components/FeaturedDestinations/FeaturedDestinations";
import TopTour from "../components/TopTour/TopTour";
import SearchField from "../components/SearchField";

export default function Home() {
  return (
    <>
      <SearchField />
      <CardSection />
      <FeaturedDestinations />
      <TopTour />
    </>
  );
}
```

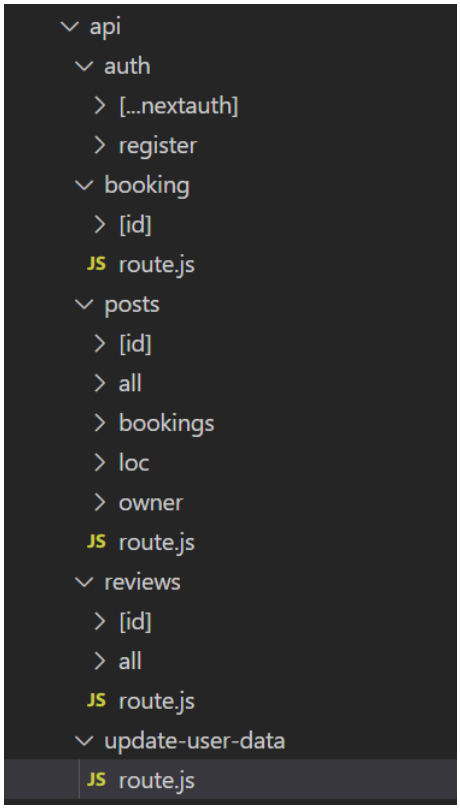
Kodni isječak 3.2: *Primjer page.jsx datoteke, u /src/app direktoriju repozitorija*

3.3 Poslužiteljska strana sustava

U ovom poglavlje opisuje kako je postavljena poslužiteljska strana implementiranog sustava. Programeri mogu jednostavno stvoriti *API*-je uz pomoć snažne *Next.js API* funkcionalnosti okvira. Mogu se uspostaviti krajnje točke *API*-ja neposredno uz kod sučelja koristeći Next *API* rute, uklanjajući zahtjev za zasebnim poslužiteljem. Rute *API*-ja automatski se optimiziraju i objavljuju na platformu ovom metodom, što usmjerava razvojni proces i jamči vrhunske performanse. *Next API* koristan je alat za stvaranje dinamičnih i responzivnih *online* aplikacija, bilo da je riječ o

stvaranju *RESTful API*-ja ili o upravljanju logikom na strani poslužitelja. Nudi jednostavan i učinkovit način rukovanja pozadinskom funkcionalnošću unutar Next.js aplikacije.

Na slici 3.2, prikazano su *API* rute ovog sustava. Primjer jedne *API* rute sustava i poziva iste, demonstrirani su u kodnim isječcima 3.3 i 3.4.



Slika 3.1: *API rute ovog sustava*

```
import { db } from "@/lib/database"; //Veza na bazu podataka i njene modele
import { NextResponse } from "next/server";

export const GET = async (request, { params }) => {
  const { id } = params;
  try {
    //Dohvaćanje podataka iz baze
    const reveiw = db.Review.findById(id).populate("accomodation");

    return NextResponse.json(reveiw, { status: 200 });
  } catch (err) {
    return new NextResponse(err.message, {
      status: 500,
    });
  }
};
```

Kodni isječak 3.3: *Primjer API rute*

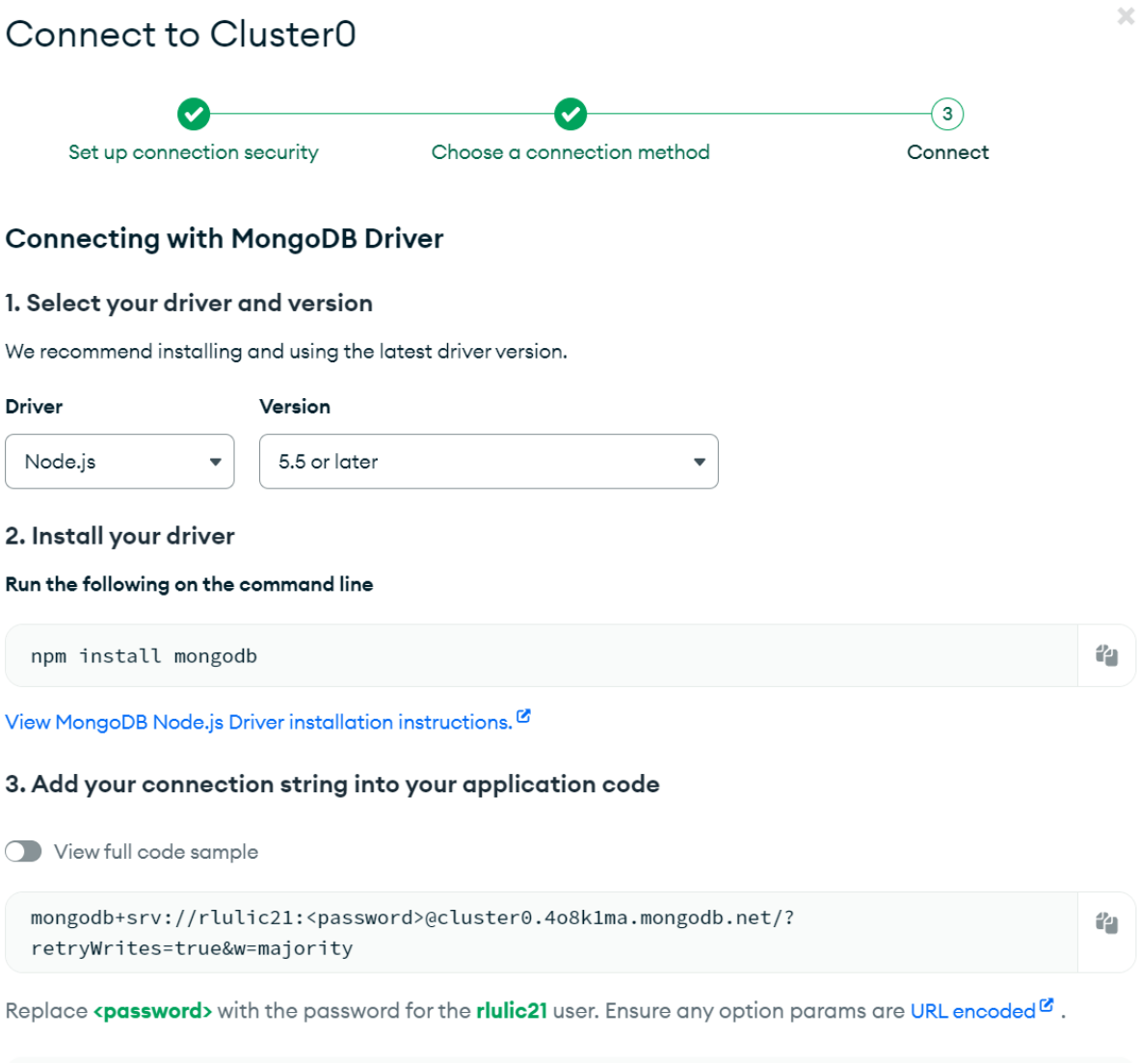
```
export const fetchPosts = async () => {
  try {
  //Dohvaćanje API rute za prikupljanje recenzija
    const response = await fetch("/API/reviews/all");
    if (!response.ok) {
      throw new Error("Network response was not ok");
    }
    const data = await response.json();
    return data;
  } catch (error) {
    throw new Error(`Error fetching posts: ${error.message}`);
  }
};
```

Kodni isječak 3.4: *Primjer poziva API rute*

Uspostavljanje konekcije na MongoDB izgleda ovako:

Veza s MongoDB bazom, slika 3.3, uspostavlja se putem MongoDB Atlas-a – *fully-managed* (u potpunosti upravljiva) baza podataka u oblaku, koja obrađuje svu složenost postavljanja, upravljanja i ispravljanja implementacija na poslužiteljskoj strani sustava.

Unutar repozitorija sustava potrebno je samo postaviti konekcijski *string* u *.env* lokalnu datoteku, kodni isječak 3.5, te u *database.js* datoteci uspostaviti samu vezu s MongoDB Atlas-om (kodni isječak 3.6). U kodnom isječku 3.7 demonstriran je primjer modela podataka, koji se kreira u *database.js* datoteci.



Slika 3.2: *Postupak uspostavljanja MongoDB baze u projektu [8]*

```
MONGO =
mongodb+srv://rlulic21:rlulic21@cluster0.4o8k1ma.mongodb.net/?retryWrites=true&w=
majority
```

Kodni isječak 3.5: *.env lokalna datoteka*

```
mongoose.connect(process.env.MONGO ?? "");
mongoose.set("debug", false);
mongoose.Promise = global.Promise;
```

Kodni isječak 3.6: *Konekcija na MongoDB Atlas pomoću mongoose knjižnice*

```
function userModel() {
  const schema = new Schema(
    {
      email: { type: String, unique: true, required: true },
      name: { type: String, required: false },
      image: { type: String, required: false },
      password: { type: String, required: false },
      country: { type: String, required: false },
      birthday: { type: String, required: false },
      surname: { type: String, required: false },
    },
    {
      timestamps: true,
    }
  );

  return mongoose.models.User || mongoose.model("User", schema);
}
```

Kodni isječak 3.7: *Primjer modela podataka*

4 SLUČAJEVI KORIŠTENJA IMPLEMENTIRANOG RJEŠENJA

U ovom poglavlju su predstavljeni svi glavni slučajevi korištenja razvijene platforme, te kako su oni izvedeni na klijentskoj i poslužiteljskoj strani sustava.

4.1 Registracija i prijava u sustav

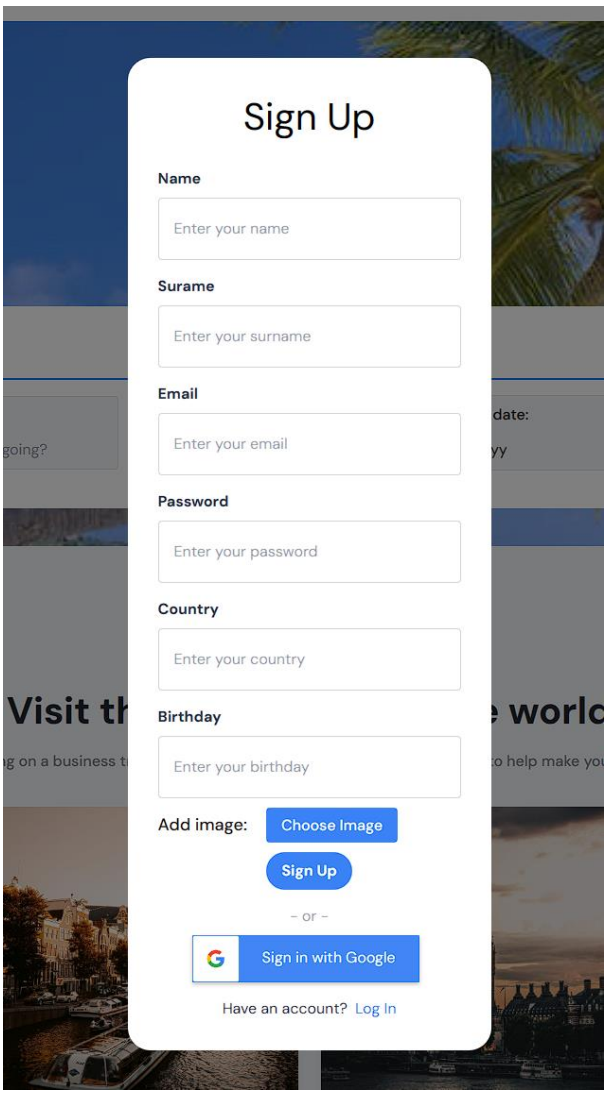
Registracija korisnika i prijava bitne su komponente razvijene platforme koje omogućavaju korisnicima stvaranje računa i pristup značajkama i funkcijama koje sustav nudi. U ovom odjeljku se detaljno raspravlja o procesima registracije i prijave.

4.1.1 Registracija

Registracijska komponenta omogućuje korisnicima stvaranje novih računa na platformi. Procesom registracije prikupljaju se bitni podaci o korisniku i sigurno se pohranjuju u bazu podataka. U nastavku je pregled procesa registracije, uključujući relevantni isječak koda; gdje je kod objašnjen komentari u formi // *Komentar*:

Komponenta za registraciju, slika 4.1, upravlja registracijom korisnika i prikazuje obrazac za registraciju. Ovaj obrazac prikuplja sljedeće korisničke podatke:

- Ime
- Prezime
- E-mail
- Lozinka
- Zemlja
- Rođendan
- Slika profila



Slika 4.1: *Komponenta za registraciju korisnika*

Korisnici šalju ove informacije kako bi stvorili svoje račune. Komponenta je integrirana s NextAuth, koji pruža strategije provjere autentičnosti za e-poštu i lozinku. Ako korisnik već postoji, bit će prijavljen; inače će se stvoriti novi račun.

Kodni isječak 4.1 prikazuje proces registracije koji se pokreće klikom na „Sign Up“ gumb komponente. U kodnom isječku 4.2 je demonstrirano kako radi *signIn()* funkcija, kojom se kreira korisnik pozivom modela korisnika iz *database.js* datoteke (kodni isječak 4.3).

```
1  import { signIn } from "next-auth/react";
2
3  const Signup = () => {
4    // Funkcija koja se izvršava prilikom slanja registracijske forme, tj. klikom
5  na „Sign Up“
6    const onSubmit = async (e) => {
7      e.preventDefault();
```

```
    setLoading(true);
    try {
      await signIn("credentials", {
        //podaci koji se šalju u signIn funkciju session.js datoteke
        name: formValues.name,
        surname: formValues.surname,
        email: formValues.email,
        password: formValues.password,
        country: formValues.country,
        birthday: formValues.birthday,
        image: formValues.image,
        redirect: false,
        callbackUrl: "/",
      });
      //. . .
  };
```

Kodni isječak 4.1: *Dio koda registracijske komponente*

```
async signIn({ user }) {
  try {
    if (!user.email || !user.password) return false;
//Učitaj korisnika ako postoji, kreiraj ga ako ne postoji
const userExists = await db.User.findOne({
email: user.email,
password: user.password,
});

    if (!userExists) {
      await db.User.create({
        name: user?.name,
        surname: user?.surname,
        email: user?.email,
        image: user?.image,
        password: user?.password,
        country: user?.country,
        birthday: user?.birthday,
        image: user?.image,
      });
    }

    return true;
  } catch (error) {
    console.log("Error checking if user exists: ", error.message);
    return false;
  }
},
```

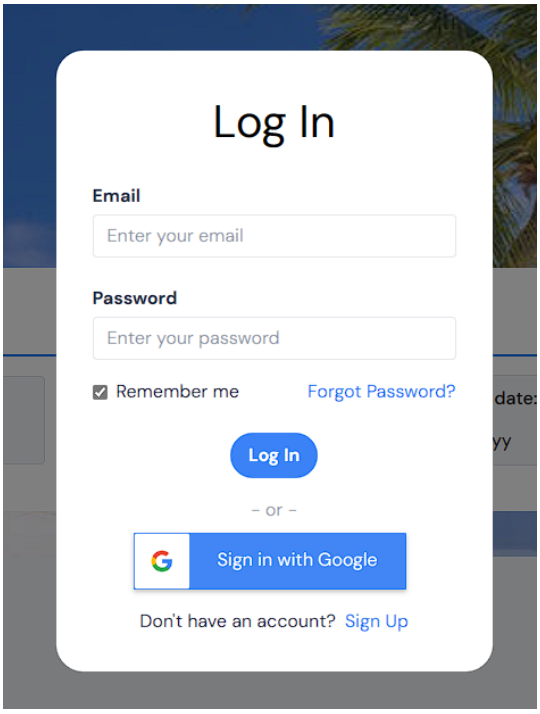
Kodni isječak 4.2: *Dio koda session.js datoteke*

```
function userModel() {
  const schema = new Schema(
    {
      email: { type: String, unique: true, required: true },
      name: { type: String, required: false },
      image: { type: String, required: false },
      password: { type: String, required: false },
      country: { type: String, required: false },
      birthday: { type: String, required: false },
      surname: { type: String, required: false },
    },
    {
      timestamps: true,
    }
  );

  return mongoose.models.User || mongoose.model("User", schema);
}
```

Kodni isječak 4.3: *Dio koda database.js datoteke, model korisnika*

4.1.2 Prijava

Registrirani korisnici mogu se prijaviti na svoje račune kako bi pristupili značajkama platforme. Proces prijave zahtijeva od korisnika unos e-pošte i lozinke. Komponenta za prijavu, slika 4.2, upravlja prijavom korisnika u sustav i prikazuje obrazac za prijavu.



Slika 4.2: *Komponenta za prijavu korisnika*

U kodnom isječku 4.4 objašnjen je postupak prijave korisnika.

```
import { signIn } from "next-auth/react";
const Login = () => {
     // Funkcija koja se izvršava prilikom slanja forme za prijavu, tj. klikom na
„Log in“
  const handleLogin = async (e) => {
    e.preventDefault();
  //podaci koji se šalju u signIn funkciju session.js datoteke
    try {
      await signIn("credentials", { email, password, redirect: false,
callbackUrl: '/' });
      } //. . .
};
```

Kodni isječak 4.4: *Dio koda komponente za prijavu*

Komponenta za prijavu također je integrirana s NextAuth, pružajući sigurnu autentifikaciju za korisnike.

4.1.3 Prijava putem Google korisničkog računa

Uz tradicionalnu prijavu putem e-pošte i lozinke, implementirana platforma korisnicima nudi pogodnost prijave s njihovim Google računima. Ova značajka koristi Google OAuth autentifikaciju, olakšavajući korisnicima s Google računima pristup platformi. U nastavku je pregled procesa prijave na Google.

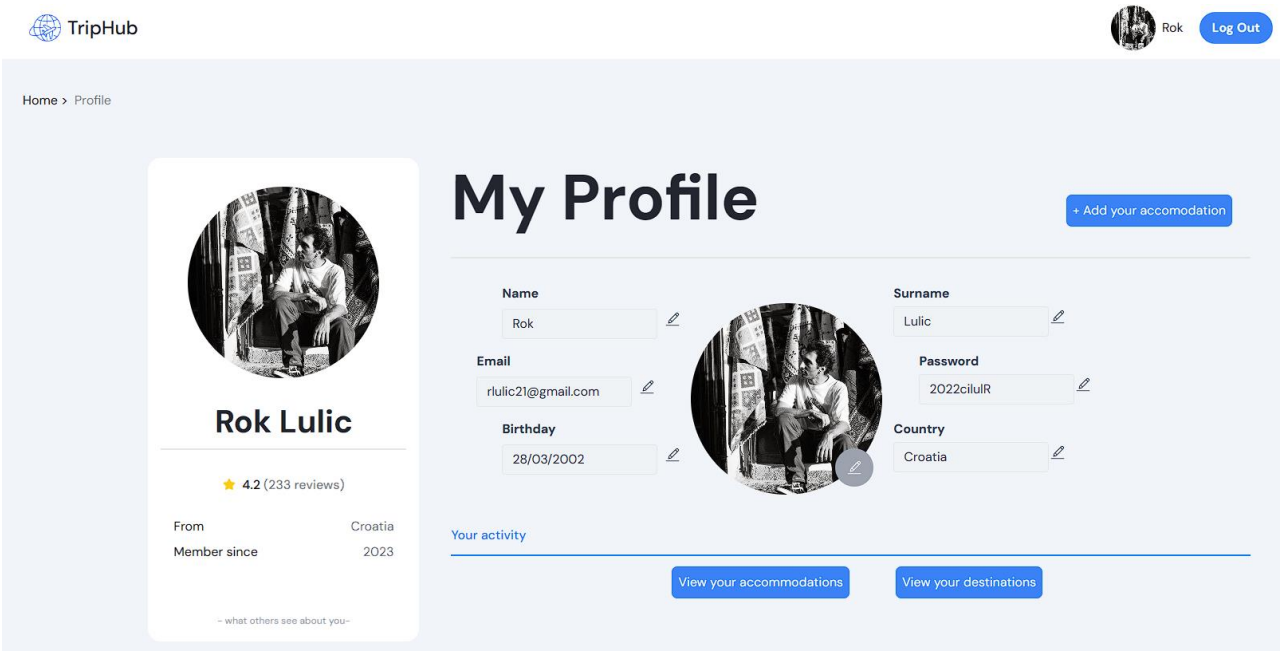
Korisnici se mogu početi prijavljivati s Google-om klikom na gumb "Sign in with Google" vidljivim na slikama 4.1.1 i 4.1.2. Ova radnja pokreće tok Google OAuth, kojim se korisnici preusmjeravaju na Google-ovu stranicu za provjeru autentičnosti. Nakon uspješne autentifikacije na Googleu, korisnici se preusmjeravaju natrag na platformu, gdje dobivaju pristup svojim računima.

Google-ova integracija prijave povećava praktičnost i sigurnost korisnika korištenjem Google-ovog robusnog sustava autentifikacije.

4.2 Pregled korisničkog računa i upravljanje podacima

Upravljanje korisničkim računom bitan je aspekt razvijene platforme koji korisnicima omogućuje pregled i upravljanje osobnim podacima. Ovaj odjeljak istražuje funkcije povezane s upravljanjem korisničkim računom, uključujući ažuriranja profila i upravljanje podacima.

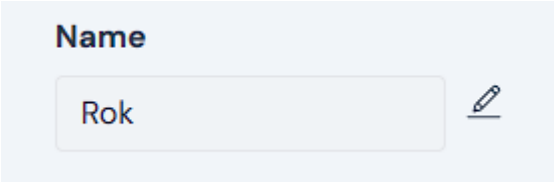
Tek nakon prijave u sustav, korisniku se omogućuje pregled stranice profila (slika 4.3).



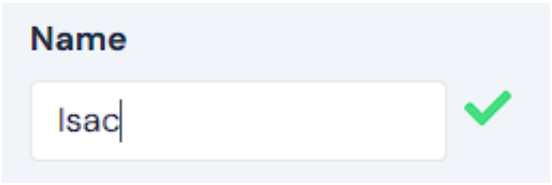
Slika 4.3: *Stranica korisničkog profila*

4.2.1 Ažuriranje podataka računa

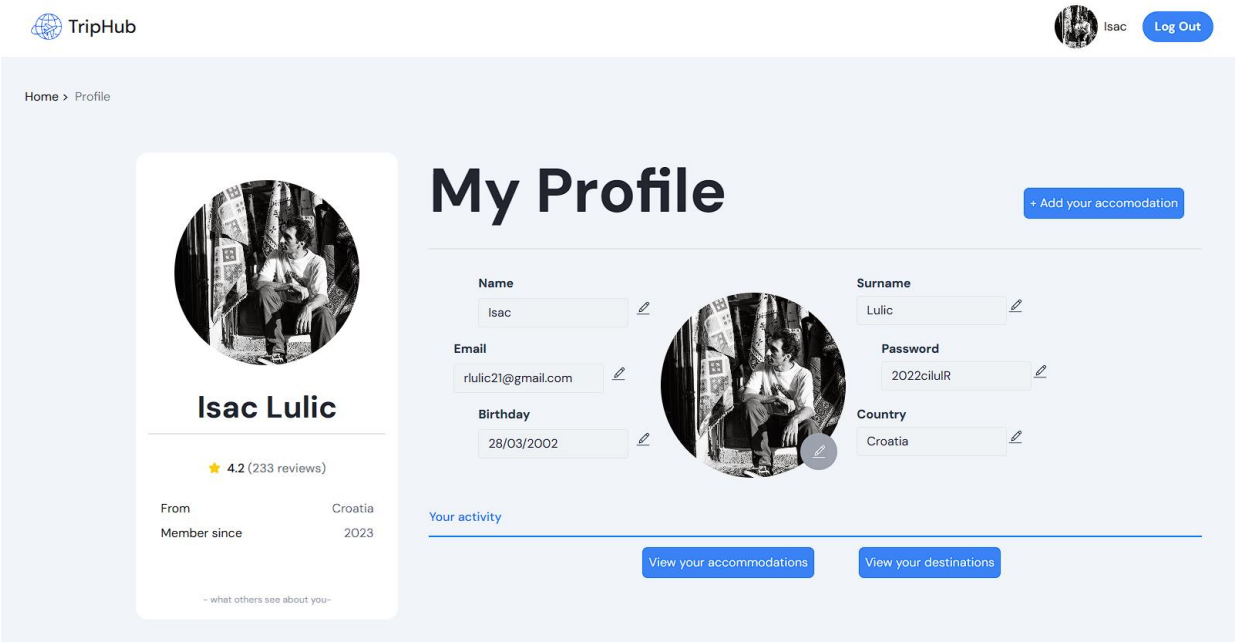
Korisnici bi trebali moći ažurirati svoje podatke o profilu u bilo kojem trenutku. To uključuje promjenu detalja kao što su ime, prezime, e-mail, lozinka, država, rođendan i profilna slika. Klikom na ikonu olovke, slika 4.4, korisniku se omogućava promjena podataka profila. Taj proces završava klikom na ikonu kvačice, slika 4.5; nakon kojeg osvježavanjem stranice vide ažurirani podaci, slika 4.6.



Slika 4.4: *Polje imena korisnika prije procesa ažuriranja*



Slika 4.5: *Polje imena korisnika tijekom procesa ažuriranja*



Slika 4.6: *Ažurirana stranica profila*

Proces ažuriranja podataka se odvija tako što pri spremanju novih podataka, klikom na kvačicu, *DataS.jsx* komponenta šalje nove podatke *API* ruti za ažuriranje podataka (kodni isječak 4.5). *API* ruta, kodni isječak 4.6, dohvaća ulogiranog korisnika i ažurira mu korisničke podatke.

```
// Funkcija koja se izvršava prilikom klika na zelenu kvačicu (Slika 4.2.3)
const handleSave = async (setStateFunc, fieldName, stateValueToUpdate) => {
  try {
    setStateFunc(false);
    const updatedUserData = { [fieldName]: stateValueToUpdate };
//pozivanje na API rutu za ažuriranje podataka
    const response = await fetch("/API/update-user-data", {
      method: "PATCH",
      headers: {
        "Content-Type": "application/json",
```

```
      },
      body: JSON.stringify({
        userId: session.user._id,
        updatedUserData,
      }),
    });

    const data = await response.json();
  } catch (error) {
    console.error("Error updating user data:", error);
  }
};
```

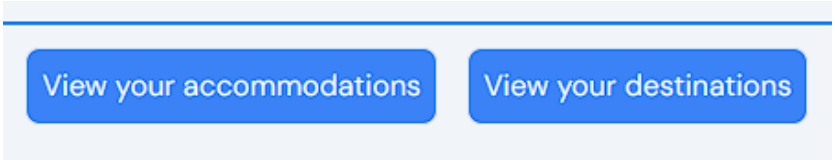
Kodni isječak 4.5: *Dio koda DataS.jsx komponente*

```
import { db } from "@/lib/database";
import { NextResponse } from "next/server";

export const PATCH = async (request) => {
  try {
    const { userId, updatedUserData } = await request.json();

    // Ažuriranje podataka korisnika prema njegovom _id
    await db.User.updateOne(
      { _id: userId },
      {
        $set: updatedUserData,
      }
    );

    return NextResponse.json({ messag: "success" }, { status: 200 });
  } catch (error) {
    console.error("Error updating user data:", error);
    return NextResponse.json({ messag: "no success" }, { status: 500 });
  }
};
```

Kodni isječak 4.6: *API ruta za ažuriranje podataka*

4.2.2 Upravljanje podacima

Upravljanje podacima omogućuje korisnicima pregled i upravljanje podacima povezanim s njihovim računima. To uključuje pregledavanje i brisanje/ažuriranje objavljenog sadržaja (smještajnih jedinica), te povijest rezervacija prijavljenog korisnika, slika 4.7.

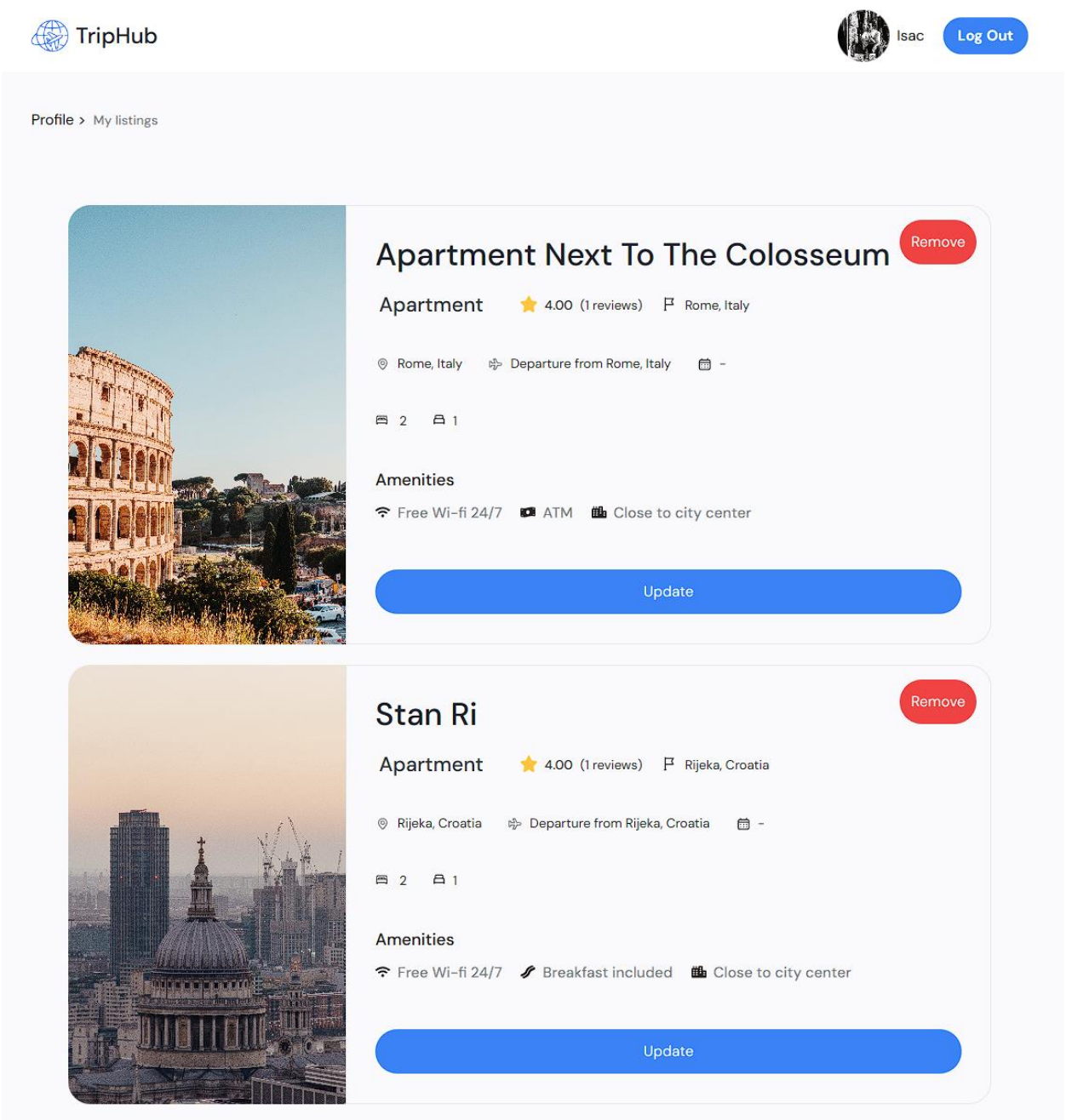


Slika 4.7: *Gumbi za pregled objavljenih/rezerviranih smještajnih jedinica*

Klikom na gumb „View your accommodations“, kodni isječak 4.7, korisnika se šalje na stranicu */hotel-list/my* (slika 4.8). Dolaskom na */hotel-list/my* poziva se *API* ruta za dohvaćanje smještaja čiji je vlasnik prijavljeni korisnik, kodni isječak 4.8.

```
//Klikom na gumb, korisnik se šalje na stranicu /hotel-list/my
<Link href="/hotel-list/my">
                <button className="ml-2 mr-10 inline rounded-lg border bg-blue-
500 p-2 align-super text-white md:ml-4">
                  View your accommodations
                </button>
</Link>
```

Kodni isječak 4.7: *Dio koda Profile.jsx komponente (1)*



Slika 4.8: *Stranica za pregled objavljenih smještajnih jedinica*

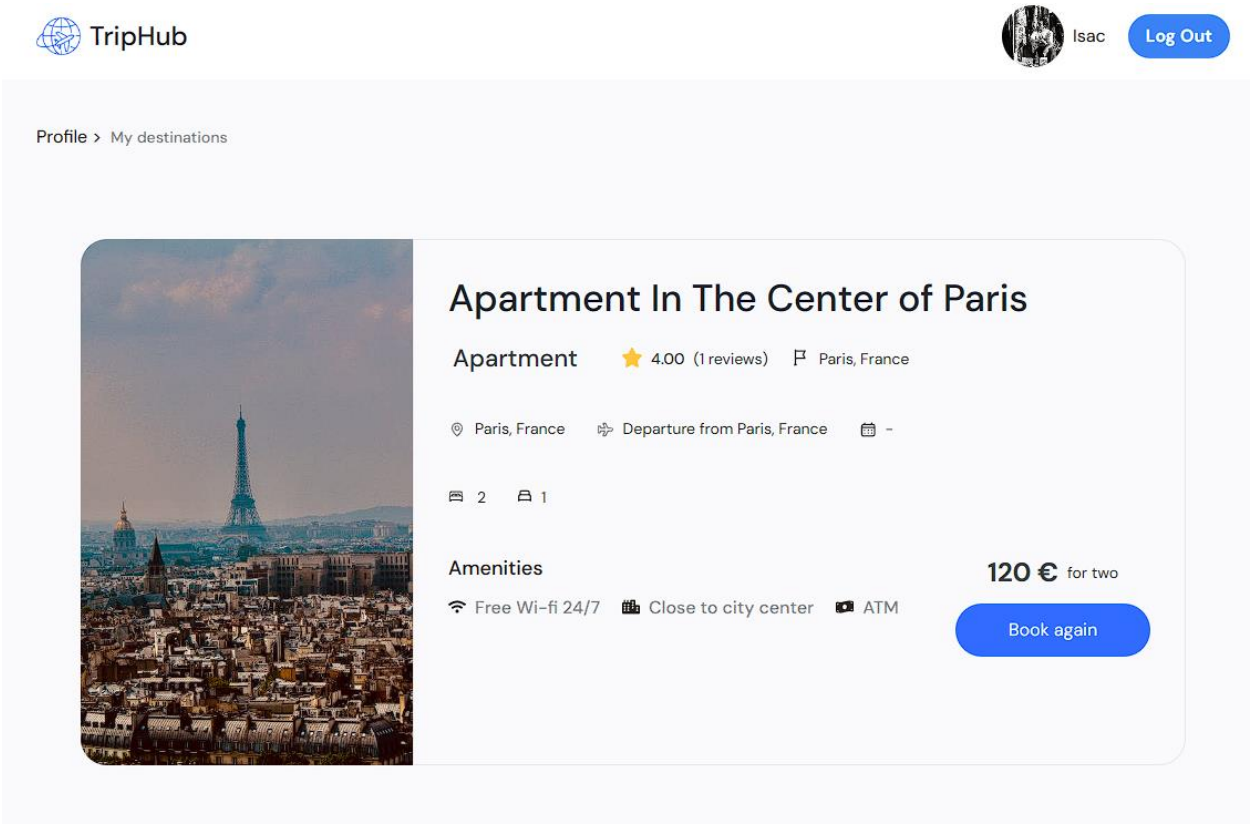
```
import { db } from "@/lib/database";
import { authOptions } from "@/lib/session";
import { getServerSession } from "next-auth";
import { NextResponse } from "next/server";

export const GET = async (request) => {
  //Prikupljanje podataka o prijavljenom korisniku
  const session = await getServerSession(authOptions);

  if (!session)
    return NextResponse.json({ message: "Not signed in" }, { status: 401 });

  const { _id } = session.user;

  try {
  //Dohvaćanje svih jedinica čiji je vlasnik prijavljeni korisnik
    const accommodations = await db.Post.find({
      owner: _id,
    }).populate("owner");

    if (!accommodations || accommodations.length === 0) {
      return new NextResponse("No accommodations found", {
        status: 404,
      });
    }

    return NextResponse.json(accommodations, { status: 200 });
  } catch (err) {
    return new NextResponse(err.message, {
      status: 500,
    });
  }
};
```

Kodni isječak 4.8: *API ruta za dohvaćanje smještajnih jedinica prijavljenog korisnika*

Sukladno tome, klikom na gumb „View your destinations“, kodni isječak 4.9, korisnika se šalje na stranicu */hotel-list/my-destinations* (slika 4.9). Dolaskom na */hotel-list/my-destinations* poziva se *API* ruta za dohvaćanje smještaja koje je prijavljeni korisnik prethodni posjetio, kodni isječak 4.10.

```
//Klikom na gumb, korisnik se šalje na stranicu /hotel-list/my-bookings
<Link href="/hotel-list/my-bookings">
                <button className="ml-2 inline rounded-lg border bg-blue-500 p-2
align-super text-white md:ml-4">
                  View your destinations
                </button>
</Link>
```

Kodni isječak 4.9: *Dio koda Profile.jsx komponente (2)*



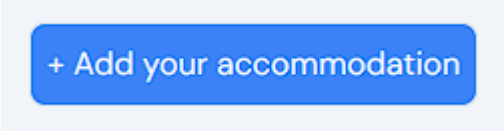
Slika 4.9: *Stranica za* p*regled posjećenih smještajnih jedinica*

```
import { db } from "@/lib/database";
import { authOptions } from "@/lib/session";
import { getServerSession } from "next-auth";
import { NextResponse } from "next/server";

export const GET = async (request) => {
   //Prikupljanje podataka o prijavljenom korisniku
  const session = await getServerSession(authOptions);
  console.log("Here");

  if (!session)
    return new NextResponse({ message: "Not signed in" }, { status: 401 });

  const { _id } = session.user;

  try {
    //Dohvaćanje svih rezervacija koje je napravioprijavljeni korisnik
    const books = await db.Book.find({ owner: _id });

    if (!books || books.length === 0) {
      return new NextResponse("No books found for the user", {
        status: 404,
      });
    }

    //Izvlačenje id oznaka smještajnih jedinica iz rezervacija
    const listingIds = books.map((book) => book.accomodation);

    //Dohvaćanje smještajnih jedinica
    const accommodations = await db.Post.find({
      _id: { $in: listingIds },
    }).populate("owner");

    if (!accommodations || accommodations.length === 0) {
      return new NextResponse("No accommodations found", {
        status: 404,
      });
    }

    return NextResponse.json(accommodations, { status: 200 });
  } catch (err) {
    return new NextResponse(err.message, {
      status: 500,
    });
  }
};
```

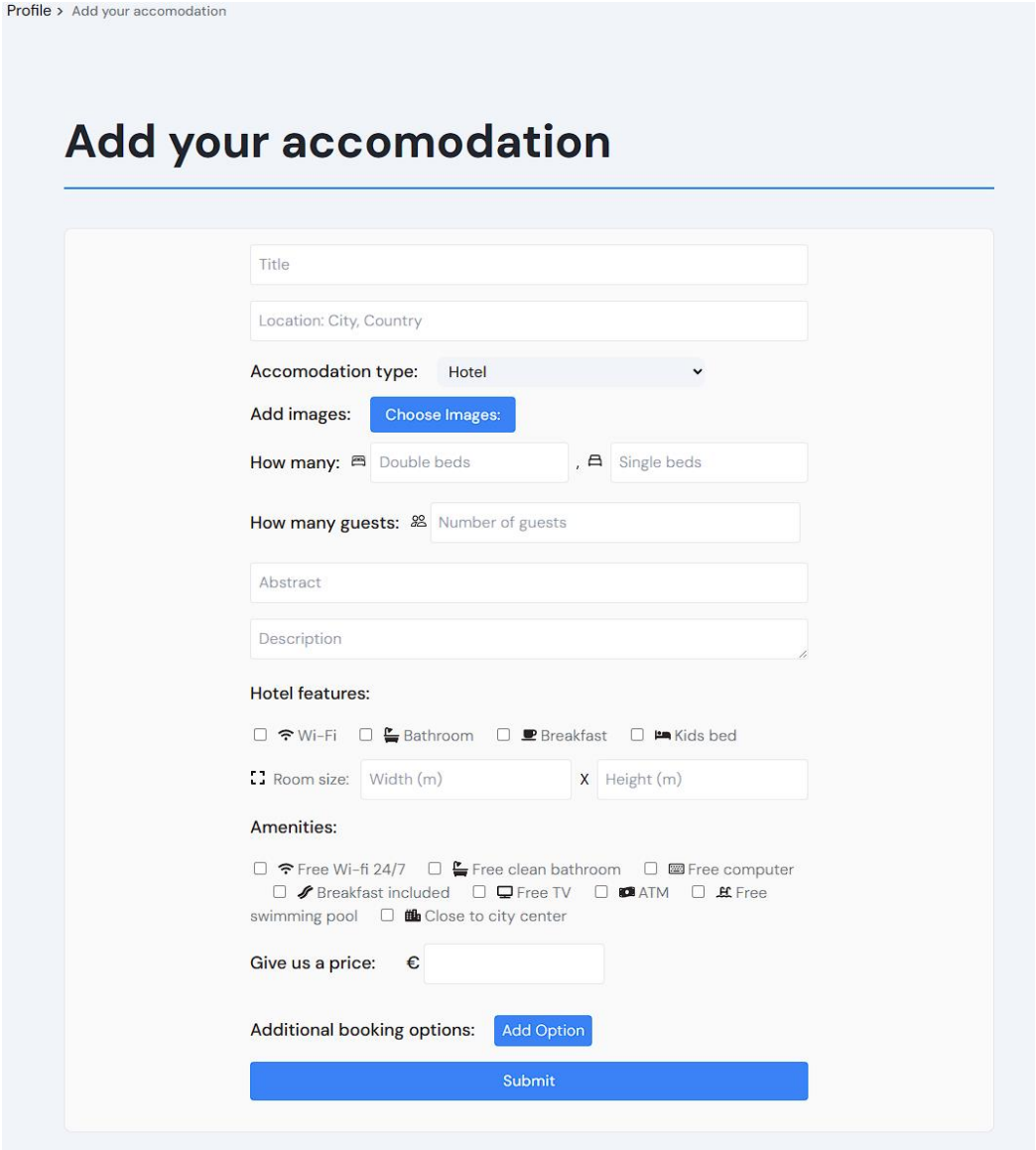
Kodni isječak 4.10: *API ruta za dohvaćanje posjećenih smještajnih jedinica prijavljenog korisnika*

4.3 Dodavanje vlastitih smještajnih jedinica

Proces dodavanja vlastitih smještajnih jedinica moguće je pokrenuti tek nakon što je korisnik prijavljen u sustav. Na stranice profila potrebno je kliknuti na gumb „Add your accommodation“, slika 4.11, te će se korisnika odvesti na stranicu za dodavanje smještajnih jedinica, slika 4.12. U nastavku je pregled procesa upravljanja podacima.



Slika 4.10: *Gumb na stranici profila, za pokretanje procesa stvaranja smještajnih jedinica*



Slika 4.11: *Stranica za dodavanje vlastitih smještajnih jedinica*

Na stranici za dodavanje smještajnih jedinica nalazi se obrazac putem kojega se korisniku nudi unos brojnih informacija o samom smještaju, a neki od njih su: lokacija, vrsta smještaja, broj kreveta, opis,... Klikom na „Submit“, kodni isječak 4.11, obrazac se šalje *API* ruti za stvaranje smještajnih jedinica (kodni isječak 4.12), koja potom kreira smještaj prema modelu definiranom u *database.js datoteci* (kodni isječak 4.13).

```
//Funkcija koja se pokreće klikom na „Submit“ gumb u obrazcu
const handleSubmit = async (e) => {
    e.preventDefault();
    try {
//Pozivanje „POST“ metode putem API rute za kreaciju smještajnih jedinica
      const response = await fetch("http://localhost:3000/api/posts", {
        method: "POST",
        headers: {
          "Content-Type": "application/json",
        },
        body: JSON.stringify({
          title,
          abstract,
          location,
          images,
          description,
          features,
          amenities,
          roomWidth,
          roomHeight,
          price,
          type,
          guests,
          beds,
          singleBeds,
          bookOption,
          bookOPrice,
        }),
      });
//. . .
```

Kodni isječak 4.11: *Dio koda Creator.jsx komponente*

```
import { db } from "@/lib/database";
import { authOptions } from "@/lib/session";
import { getServerSession } from "next-auth";
import { NextResponse } from "next/server";

export const POST = async (request) => {
  //Prikupljanje podataka o prijavljenom korisniku
  const session = await getServerSession(authOptions);

  if (!session)
    return NextResponse.json({ message: "Not signed in" }, { status: 401 });

  const { _id } = session.user;

  const {
    title,
    abstract,
    location,
    images,
    description,
    features,
    amenities,
    roomWidth,
    roomHeight,
    price,
    type,
    beds,
    guests,
    singleBeds,
    bookOption,
    bookOPrice,
  } = await request.json();

  //Poziv na model smještajnih jedinica u bazi podataka database.js
  const newPost = new db.Post({

    title,
    abstract,
    location,
    description,
    features,
    amenities,
    roomWidth,
    roomHeight,
    price,
    images,
    type,
    beds,
    guests,
    singleBeds,
    bookOption,
```

```
    bookOPrice,
    owner: _id,
  });

  try {
    await newPost.save();
    return new NextResponse("Post has been created!", {
      status: 201,
    });
  } catch (err) {
    return NextResponse.json(
      { message: err },
      {
        status: 500,
      }
    );
  }
};
```

Kodni isječak 4.12: *API ruta za stvaranje smještaja*

```
function postModel() {
  const schema = new Schema(
    {
      title: { type: String, unique: true, required: true },
      abstract: { type: String, required: true },
      location: { type: String, required: true },
      description: { type: String, required: true },
      features: { type: [String], required: true },
      amenities: { type: [String], required: true },
      roomWidth: { type: String, required: true },
      roomHeight: { type: String, required: true },
      price: { type: Number, required: true },
      type: { type: String, required: true },
      beds: { type: String, required: true },
      singleBeds: { type: String, required: true },
      images: { type: [String], required: true },
      owner: { type: Schema.Types.ObjectId, ref: "User", required: true },
    },
    {
      timestamps: true,
    }
  );

  return mongoose.models.Post || mongoose.model("Post", schema);
}
```

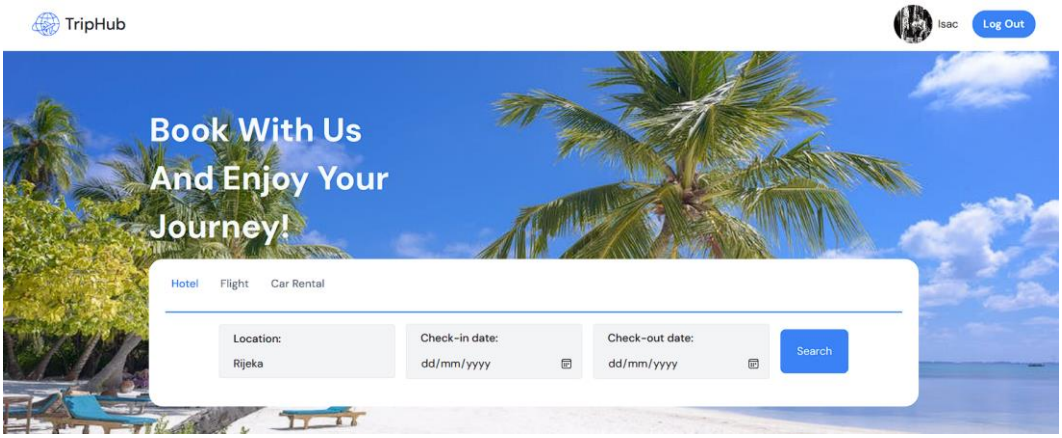
Kodni isječak 4.13: *Dio koda database.js datoteke, model smještajne jedinice*

4.4 Pregled ponude

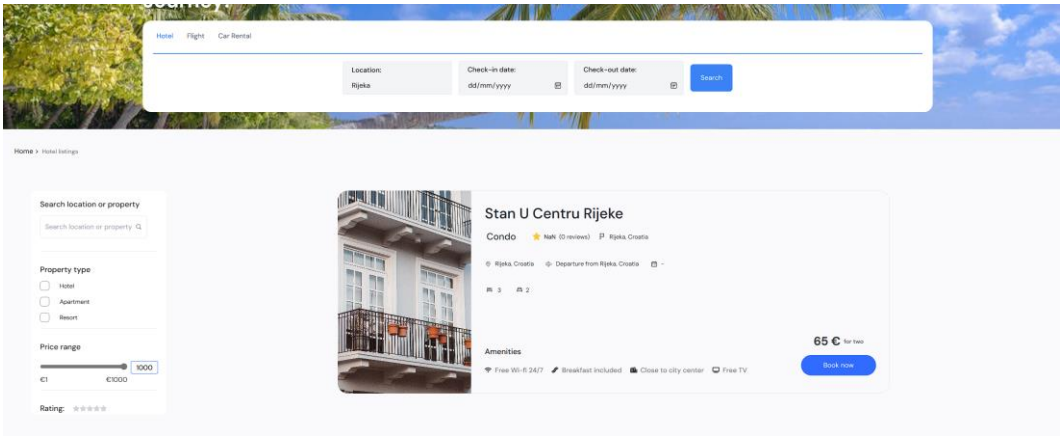
U ovom poglavlju prikazani su različiti načini na koje korisnici istražuju mogućnosti smještaja dostupnih unutar sustava. Cilj je razumjeti preferencije korisnika, dizajnirati sučelja usmjerena na korisnika i uzeti u obzir pristupačnost, osiguravajući da interakcija svakog korisnika s ponudom smještaja bude ugodno iskustvo.

4.4.1 Pretraživačka komponenta

Pretraživačka komponenta, prikazana na slici 4.13, omogućava unos ciljane destinacije u polje s naslovom „Location“ unese svoje odredište, a sustav nakon odgovarajuće pretrage prikazuje sve smještajne jedinice koje sadržavaju unesenu lokaciju, slika 4.4.2.



Slika 4.12: *Pretraživačka komponenta*



Slika 4.13: *Rezultat pretraživanja*

To se proces odvija tako što se klikom na „Search“, kodni isječak 4.14, korisnika šalje na stranicu /hotel-list/unesena_lokacija, gdje je pozvana *fetchPosts()* (kodni isječak 4.15) funkcija koja dohvaća sve smještajne jedinici s podudarnim parametrom lokacije. *fetchPosts()* funkcija i *API* ruta za prikaz smještaja po lokaciji vidljivi su u kodnim isječcima 4.16 i 4.17.

```
//Kod koji se izvrašava klikom na „Search“ gumb, šalje korisnika na stranicu
/hotel-list/usesena_lokacija
<Link href={`/hotel-list/${location}`}>
                <button
                  type="submit"
                  className="text-lg mb-4 h-16 flex-grow-0 rounded-lg bg-blue-500
px-6 py-3 text-white"
                >
                  Search
                </button>
</Link>
```

Kodni isječak 4.14: *Dio koda pretraživačke komponente SearchField.jsx*

```
//. . .
export default function ListPage({ params }) {
  const location = params.location;

  const [isFilterVisible, setFilterVisible] = useState(false);
  const [accommodations, setAccommodations] = useState([]);

  useEffect(() => {
    //Konekcija na server parametrom lokacije
    fetchPosts(location)
      .then((data) => {
        setAccommodations(data);
      })
      .catch((error) => {
        console.error("Error fetching accommodations:", error);
      });
  }, [location]);

  const handleDivClick = () => {
    setFilterVisible(!isFilterVisible);
  };

//. . .
};
```

Kodni isječak 4.15: *Dio koda page.jsx datoteke za /hotel-list/unesena_lokacija stranicu*

```
export const fetchPosts = async (loc) => {
  try {
    const response = await fetch(`/API/posts/loc/${loc}`);
    if (!response.ok) {
      throw new Error("Network response was not ok");
    }
    const data = await response.json();
    return data;
  } catch (error) {
    throw new Error(`Error fetching posts: ${error.message}`);
  }
};
```

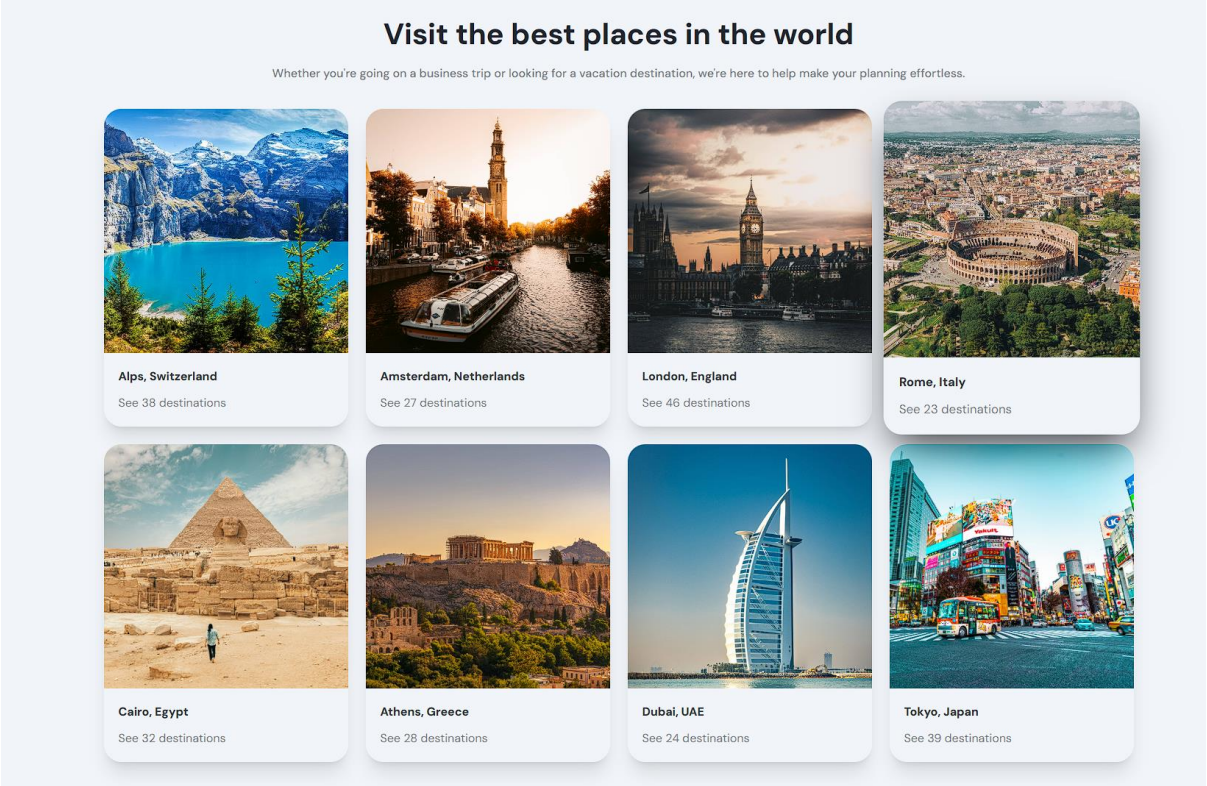
Kodni isječak 4.16: *Funkcija za dohvaćanje API rute za izlistaj smještaja po lokaciji*

```
import { db } from "@/lib/database";
import { NextResponse } from "next/server";

export const GET = async (request, { params }) => {
  const { loc } = params;

  try {
    //Dohvaćanje smještaja sa servera parametrom lokacije
    const accommodations = await db.Post.find({
      location: { $regex: loc, $options: "i" },
    }).populate("owner");

    if (!accommodations || accommodations.length === 0) {
      return new NextResponse("No accommodations found", {
        status: 404,
      });
    }

    return NextResponse.json(accommodations, { status: 200 });
  } catch (err) {
    return new NextResponse(err.message, {
      status: 500,
    });
  }
};
```

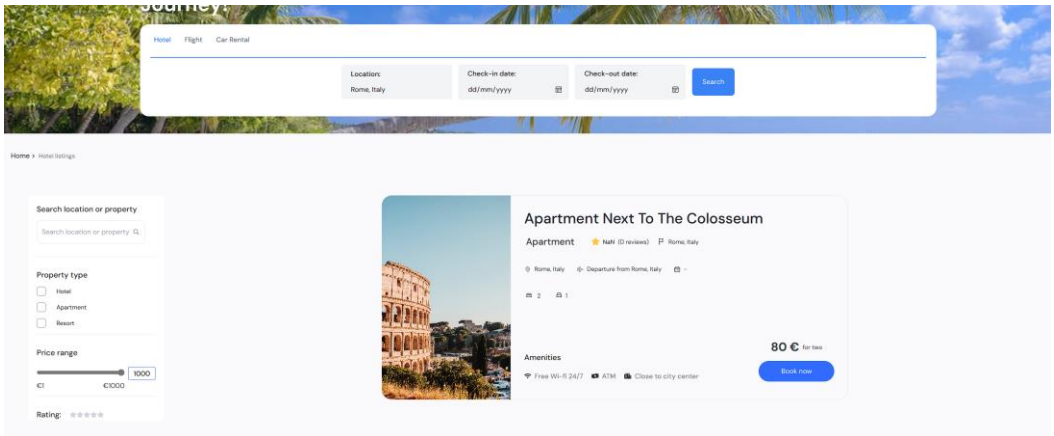
Kodni isječak 4.17: *API ruta za izlistaj smještaja po lokaciji*

4.4.2 Kartične komponente

Kartične komponente, vizualizirane na slici 4.15, imaju sličnu funkciju pretraživačkoj komponenti, omogućujući korisniku da klikom na neku od kartica odabere svoje odredište, te će mu sustav izlistati sve smještajne jedinice koje sadržavaju unesenu lokaciju (slika 4.16). Kartice vode na istu stranicu kao i pretraživačka komponenta te je proces dohvaćanja smještajnih jedinica identičan.



Slika 4.14: *Kartične komponente*



Slika 4.15: *Rezultat pretraživanje putem kartične komponente*

Kodni isječak 4.18 demonstrira kako se klikom na jednu od kartica korisnika šalje na stranicu /hotel-list/odabrana_lokacija, gdje mu se prikazuje sav smještaj na toj lokaciji.

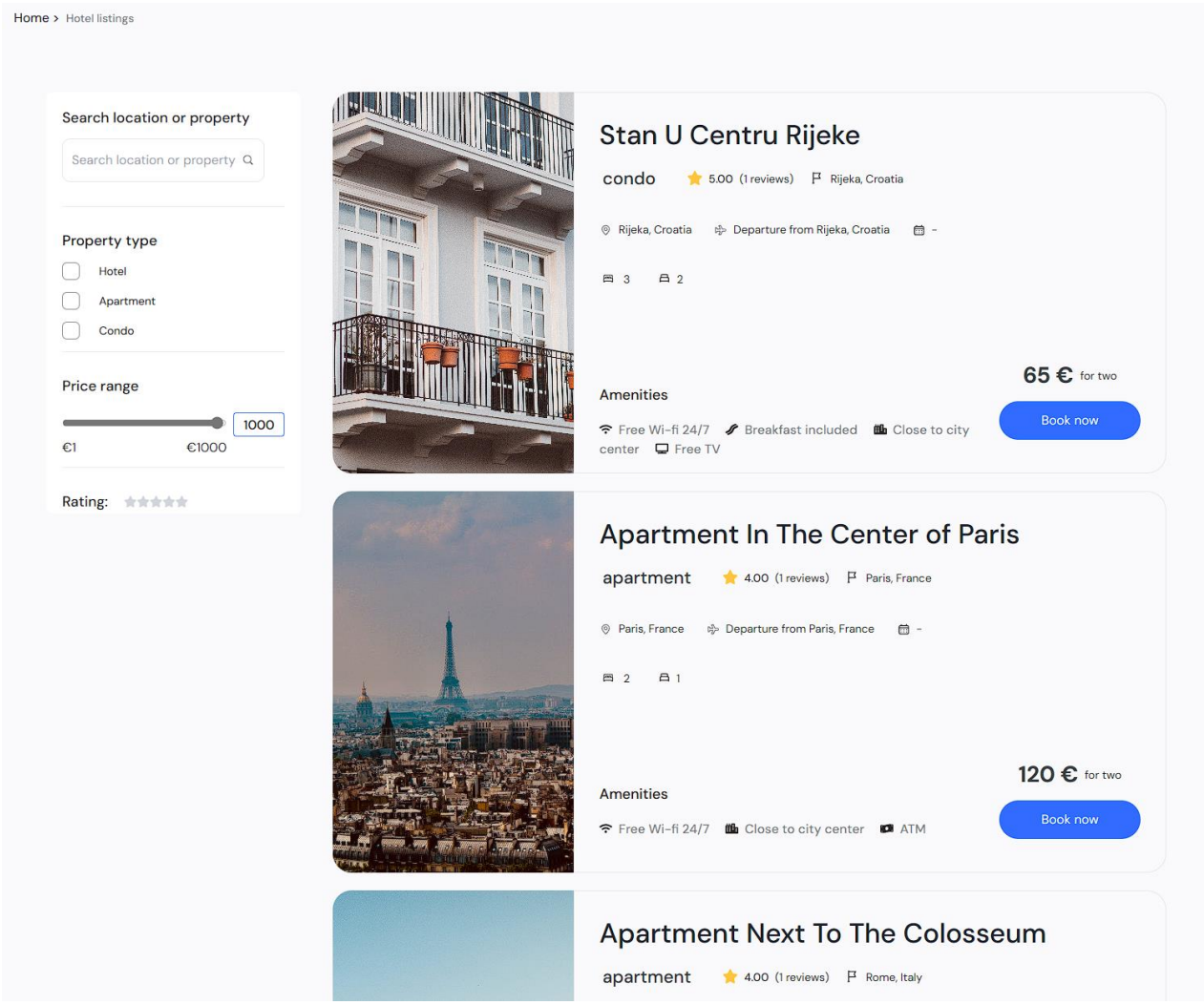
```
{cardList.map((card) => (
  //Klik na karticu šalje korisnika na stranicu /hotel-list/odabrana _lokacija
    <Link href={`/hotel-list/${card.title}`}>
      <div className="hover:shadow-opacity-50 hover:shadow-offset-2 hover:shadow-
offset-c00 over:opacity-90 cursor-pointer rounded-3xl shadow-xl transition-all
duration-200 ease-in-out hover:scale-105 hover:shadow-2xl hover:shadow-c3">
        <img className="rounded-t-3xl" src={card.img} alt={card.title} />
        <div className="p-5">
          <p className="mb-3 font-bold text-c3">{card.title}</p>
          <p className="text-lg font-normal text-c4">{card.text}</p>
        </div>
      </div>
   </Link>
))}
```

Kodni isječak 4.18: *Dio koda kartične komponente Cards.jsx*

4.4.3 Višefilterska pretraga smještajnih jedinica

Korisnik, dolaskom na bilo koju od /hotel-list, stranica ima mogućnost dodatno filtrirati izlistani sadržaj koristeći Filter.jsx komponentu (prikazano u gornjem lijevom kutu slike 4.17). Ova komponenta korisniku pruža mogućnost filtriranja smještajnih jedinica prema sljedećim parametrima:

- Vrsti smještaja
- Rasponu cijene
- Ocjeni smještajne jedinice



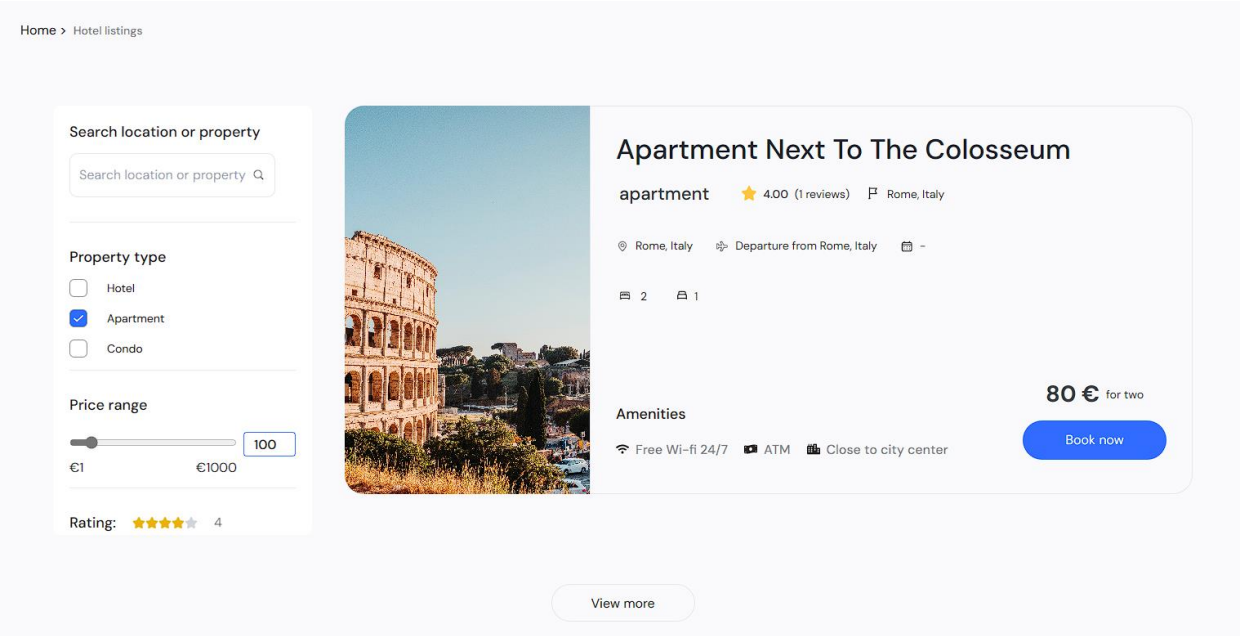
Slika 4.16: *Prikaz /hotel-list stranice*

Prilikom dolaska na neku od /hotel-list stranica, vrijednosti filtera su postavljene na maksimalni raspon cijene (0 do 1000 eura) i ocjenu smještaja 0, čime se i dalje prikazuju sve smještajne jedinice trenutne /hotel-list stranice. To se može vidjeti u URL-u stranice, čiji je sadržaj u tom slučaju sljedeći:

http://localhost:3000/hotel-list?price=1000&rating=0

Odabirom nekih od parametara filtriranja, slika 4.18, prikazat će se samo one smještajne jedinice koje zadovoljavaju te parametre, tj. ažurirat će se ReactQuery, čija se promjena opet vidi u URL-u stranice:

http://localhost:3000/hotel-list?category=apartment&price=100&rating=4



Slika 4.17: *Prikaz /hotel-list stranice nakon korištenja Filter.jsx komponente*

Taj se proces odvija tako što se promjenom u *Filter.jsx* komponenti ažurira ReactQuery, kodni isječak 4.19, te se on šalje u *API* rutu za izlistaj smještaja po parametrima filtriranja putem *fetchPosts()* funkcije (kodni isječak 4.20). *fetchPosts()* funkcija i *API* ruta prikazani su u kodnim isječcima 4.21 i 4.22.

```
1  //Kreira se novi query s ažuriranim parametrima pretrage
2  const updatedQuery = {
3    ...qs.parse(params.toString()),
4    category: selectedCategories,
5    rating,
6    price: value,
7  };
8
9  //Ažuriranje URL query-a
10 const url = qs.stringifyUrl(
11   {
12     url: "/hotel-list",
13     query: updatedQuery,
14   },
15   { skipNull: true }
16 );
17
18 router.push(url);
```

Kodni isječak 4.19: *Dio koda Filter.jsx komponente*

```
useEffect(() => {
  //Poziv funkcije za spajanje na server
  fetchPosts(query.toString())
    .then((data) => {
      setAccommodations(data);
    })
    .catch((error) => {
      console.error("Error fetching accommodations:", error);
    });
}, [query]);
```

Kodni isječak 4.20: *Dio koda page.jsx datoteke /hotel-list stranice*

```
export const fetchPosts = async (query) => {
  try {
    const response = await fetch("/API/posts/all?" + query);
    if (!response.ok) {
      throw new Error("Network response was not ok");
    }
    const data = await response.json();
    return data;
  } catch (error) {
    throw new Error(`Error fetching posts: ${error.message}`);
  }
};
```

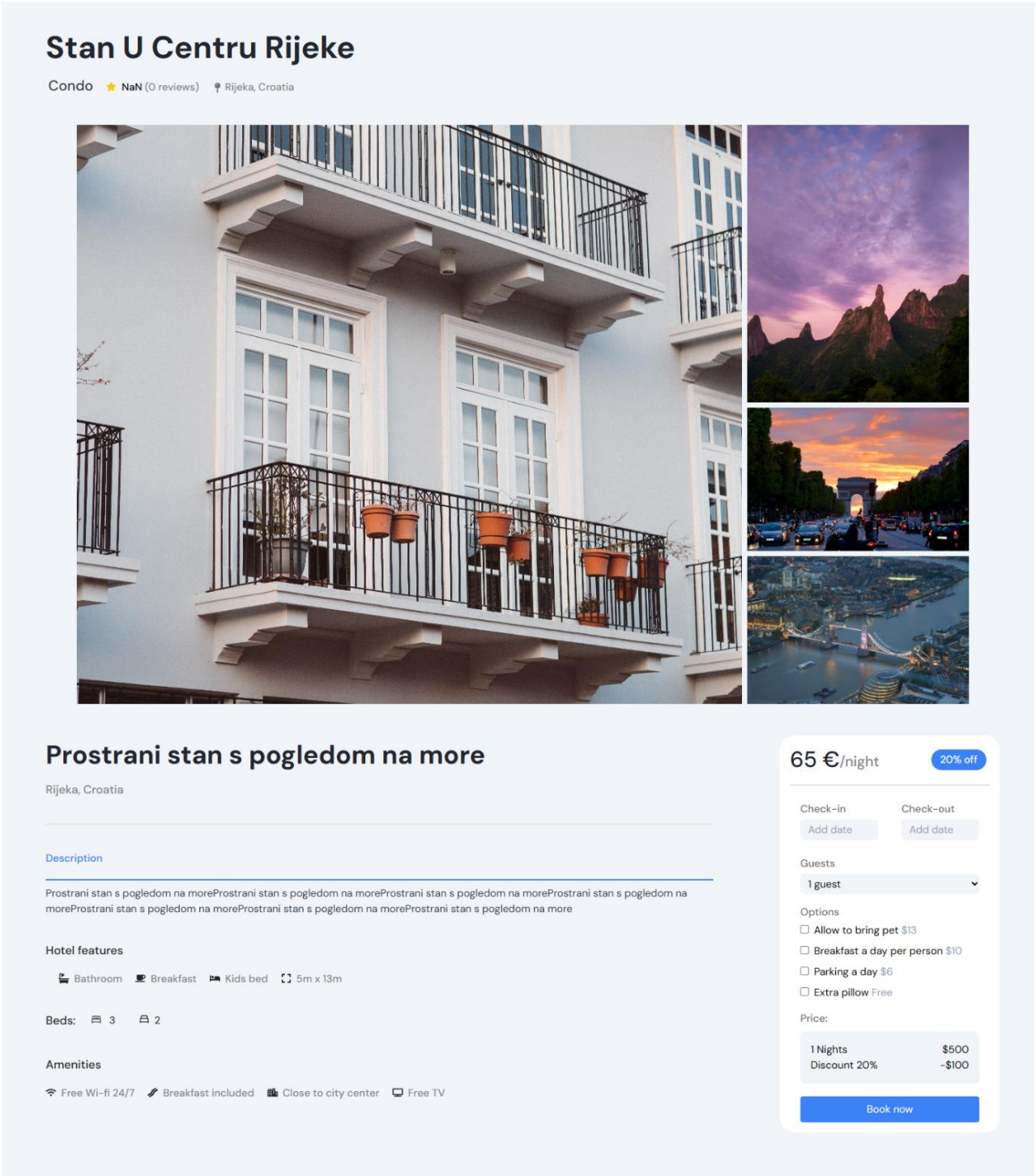
Kodni isječak 4.21: *Funkcija za dohvaćanje API rute za izlistaj smještaja po parametrima filtriranja*

```
import { db } from "@/lib/database";
import { NextResponse } from "next/server";

export const GET = async (request) => {
  try {
  //Učitavanje parametara iz query-a
    const categories = request.nextUrl.searchParams.getAll("category");
    let price = request.nextUrl.searchParams.get("price");
    if (price) price = parseInt(price);
    let rating = request.nextUrl.searchParams.get("rating");
    if (rating) rating = parseInt(rating);
    if (rating === 0) rating = null;

    console.log({
      ...(categories?.length ? { type: { $in: categories ?? [] } } : {}),
      ...(price ? { price: { $lte: price } } : {}),
    });

    let ids = [];

    if (rating) {
      const ratings = await db.Review.find({
        rating: rating
      });

      ids = ratings.map((rating) => rating.accomodation);
    }

    // Dohvaćanje svih smještajnih jedinica koje zadovoljavaju parametre
    const accommodations = await db.Post.find({
      ...(ids?.length ? { _id: { $in: ids ?? [] } } : {}),
      ...(categories?.length ? { type: { $in: categories ?? [] } } : {}),
      ...(price ? { price: { $lte: price } } : {}),

    }).populate("owner");

    return NextResponse.json(accommodations, { status: 200 });
  } catch (err) {
    return new NextResponse(err.message, {
      status: 500,
    });
  }
};
```

Kodni isječak 4.22: *API ruta za izlistaj smještaja po parametrima filtriranja*

4.4.4 Detaljan pregled smještajnih jedinica

Klikom na neki od pretraženih smještaja, korisnika se šalje na novu stranicu /hotel-details/ id_smještaja, slika 4.19. Ovdje korisnik dobiva mogućnost bolje sagledati ponudu odabrane smještajne jedinice, kao i opciju rezervacije smještaja, te sagledavanje i ostavljanje recenzija.



Slika 4.18: *Stranica za detaljan pregled smještajne jedinice*

Kodni isječak 4.23 prikazuje poziv *API* rute za dohvaćanje podataka o smještaju te izračun srednje ocjene smještaja s brojem recenzija. *API* ruta prikazana je u kodnom isječku 4.24.

```
useEffect(() => {
  if (id) {
//Dohvaćanje podataka o smještajnoj jedinici
    fetch(`/API/posts/${id}`)
      .then((response) => response.json())
      .then((data) => {
        console.log("API Response:", data);
        setPost(data);
      })
      .catch((error) => console.error("Error fetching post:", error));
  }

//Dohvaćanje recenzija smještajne jedinice
  fetch(`/API/reviews/${id}`)
    .then((response) => response.json())
    .then((data) => {
      setReviews(data);

      //Izračun srednje ocjene i broja recenzija
      let totalRating = 0;
      for (const review of data) {
        totalRating += parseInt(review.rating);
      }
      const averageRating = totalRating / data.length;

      setRatingg(averageRating.toFixed(2));
      setNum(data.length);
    })
    .catch((error) => {
      console.error("Error fetching reviews:", error);
    });
}, [id]);
```

Kodni isječak 4.23: *Dio koda HotelDetail.jsx komponente (poglavlje 4.4.4)*

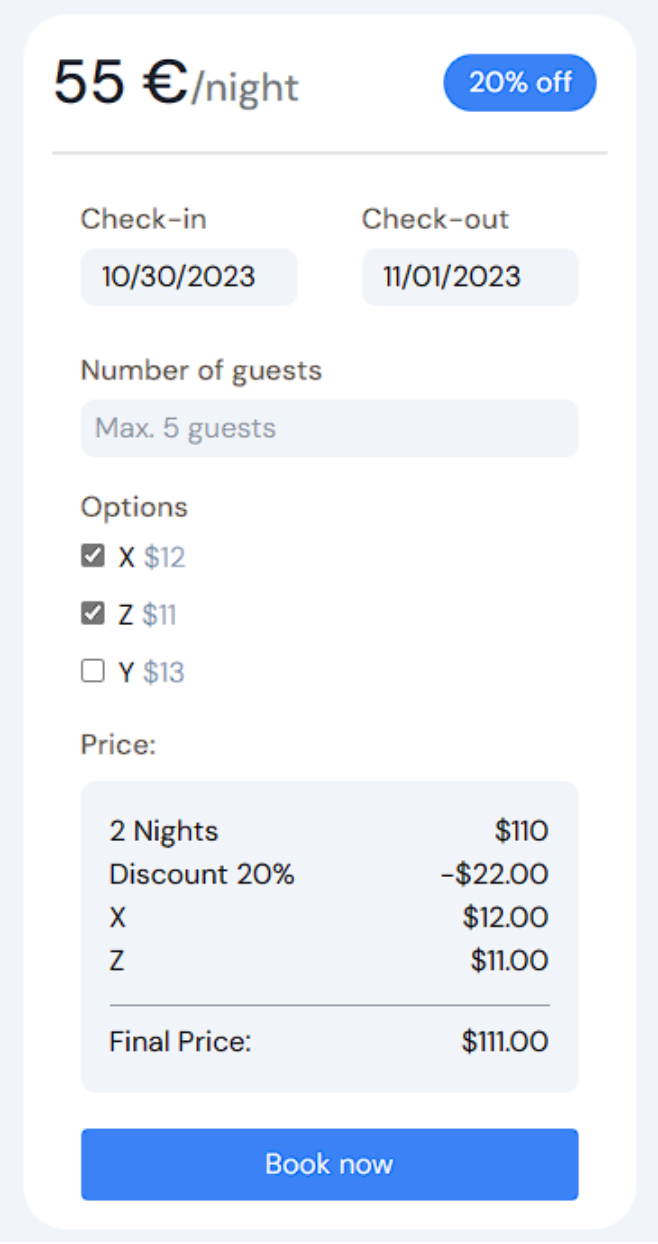
```
import { db } from "@/lib/database";
import { NextResponse } from "next/server";

export const GET = async (request, { params }) => {
  const { id } = params;

  try {
    // Dohvati smještajne jedinice prema id
    const accommodation = await db.Post.findById(id).populate("owner");

    if (!accommodation) {
      return new NextResponse("Accommodation not found", {
        status: 404,
      });
    }

    return NextResponse.json(accommodation, { status: 200 });
  } catch (err) {
    return new NextResponse(err.message, {
      status: 500,
    });
  }
};
```

Kodni isječak 4.24: *API ruta za dohvaćanje smještajne jedinice*

4.5 Rezervacija smještajnih jedinica

Kada se korisnik nalazi na stranici za detaljan pregled smještajne jedinice, ima mogućnost rezervirati taj smještaj. To čini pomoću Book.jsx komponente, slika 4.20, koja sadrži obrazac za rezervaciju smještaja. Obrazac prikuplja sljedeće informacije:

- Datum dolaska u smještaj
- Datum odlaska iz smještaja
- Broj gostiju
- Dodatne opcije



Slika 4.19: *Book.jsx komponenta za rezervaciju*

Rezervacija se kreira tako što se klikom na „Book now“ poziva *API* ruta za kreaciju rezervacije, kodni isječak 4.25. Ta *API* ruta, kodni isječak 4.26, prima podatke iz obrasca i kreira rezervaciju prema modelu rezervacije definirane u *database.js* datoteci, kodni isječak 4.27.

```
//Funkcija koja se pokreće klikom na „Book now“ gumb u obrazcu
const handleSubmit = async (e) => {
  e.preventDefault();

  try {
//Pozivanje „POST“ metode putem API rute za kreaciju rezervacije
    const response = await fetch("http://localhost:3000/API/booking", {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify({
        checkIn,
        checkOut,
        guests,
        options,
        a_id: id,
      }),
    });

    const data = await response.json();
    console.log(data);

    if (data.success) {
      const userName = data.name;
    } else {
      console.log("Review creation failed:", data.message);
    }
  } catch (error) {
    console.error(error);
  }
};
```

Kodni isječak 4.25: *Dio koda Book.jsx komponente*

```
import { db } from "@/lib/database";
import { authOptions } from "@/lib/session";
import { getServerSession } from "next-auth";
import { NextResponse } from "next/server";

export const POST = async (request) => {
  //Dohvaćanje podataka o prijavljenom korisniku
  const session = await getServerSession(authOptions);

  if (!session)
    return NextResponse.json({ message: "Not signed in" }, { status: 401 });

  const { _id } = session.user;

  const { checkIn, checkOut, guests, options, a_id } = await request.json();

  const newBook = new db.Book({
    checkIn,
    checkOut,
    guests,
    options,
    owner: _id,
    accomodation: a_id,
  });
  try {
    await newBook.save();
    return NextResponse.json(
      { message: "Book has been created!" },
      {
        status: 201,
      }
    );
  } catch (err) {
    return NextResponse.json(
      { message: err },
      {
        status: 500,
      }
    );
  }
};
```

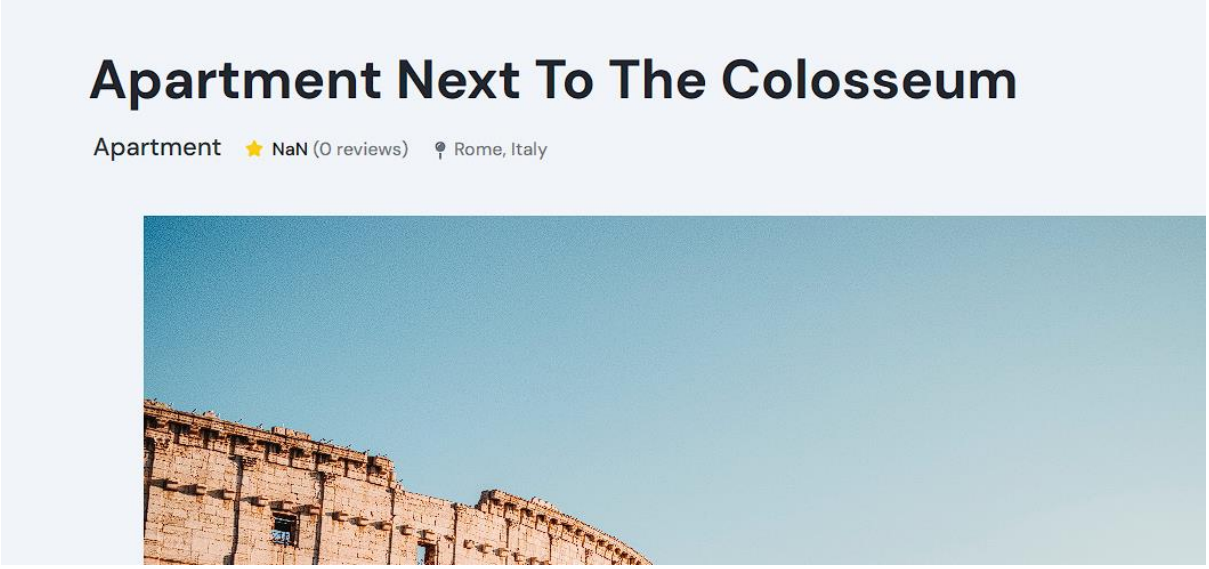
Kodni isječak 4.26: *API ruta za kreaciju rezervacije*

```
function bookModel() {
  const schema = new Schema(
    {
      checkIn: { type: String, required: true },
      checkOut: { type: String, required: true },
      guests: { type: String, required: true },
      options: { type: [String], required: true },
      owner: { type: Schema.Types.ObjectId, ref: "User", required: true },
      accomodation: {
        type: Schema.Types.ObjectId,
        ref: "Post",
        required: true,
      },
    },
    {
      timestamps: true,
    }
  );
  return mongoose.models.Book || mongoose.model("Book", schema);
}
```

Kodni isječak 4.27: *Dio koda database.js datoteke, model rezervacije*

4.6 Ocjenjivanje i komentiranje smještajnih jedinica

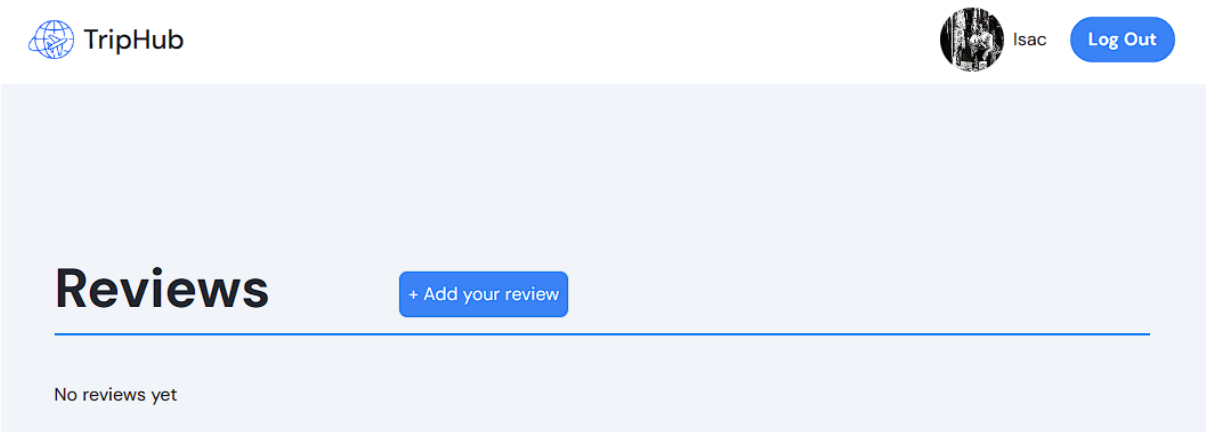
Kada se korisnik nalazi na stranici za detaljan pregled smještajne jedinice, ima mogućnost pregledati recenzije za taj smještaj. To čini klikom na sekciju recenzija ispod naslova, slika 4.21, koja šalje korisnika na stranicu /hotel-details/id_smještaja/reveiws, slika 4.22. Kodni isječak 4.6.2 demonstrira kako su te dvije stavke povezane.



Slika 4.20: *Stranica za detaljni pregled smještajne jedinice*

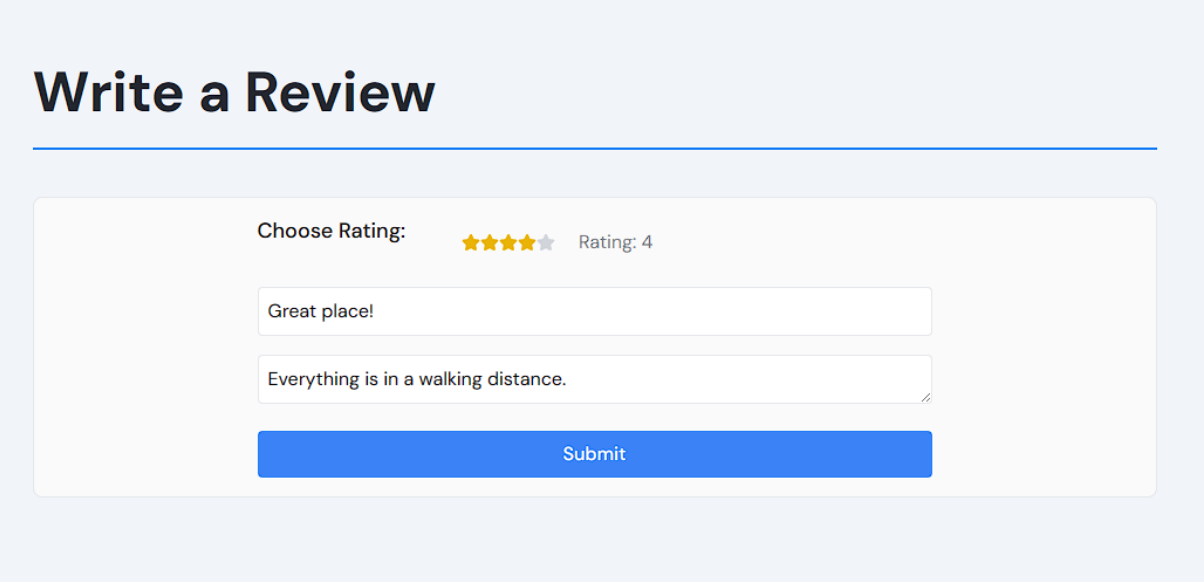
```
//Klikom se korisnika upućuje na stranicu /hotel-details/id_smještaja/reviews
<Link href={`/hotel-details/${id}/reviews`}>
              <FaStar className="mb-1 mr-1 inline text-yellow-400" /> {ratingg}{"
"}
              <span className="text-c4">({num} reviews)</span>
</Link>
```

Kodni isječak 4.28: *Dio koda HotelDetail.jsx komponente (poglavlje 4.6)*



Slika 4.21: *Stranica za pregled recenzija smještaja (Prije recenzije)*

Svaki korisnik ima pravo ostaviti samo jednu recenziju za tu smještajnu jedinicu, a to čini klikom na gumb „+ Add your review“, slika 4.22. Klikom na taj gumb korisnika se šalje na stranicu /hotel-details/id_smještaja/write-a-review, gdje se nalazi obrazac za slanje recenzije, slika 4.23.



Slika 4.22: *Obrazac za slanje recenzije*

Kreacija recenzije odvija se tako što se klikom na „Submit“ poziva *API* ruta za kreaciju recenzije, kodni isječak 4.29. Ta *API* ruta, kodni isječak 4.30, kreira recenziju prema modelu recenzije definirane u *database.js* datoteci, kodni isječak 4.31.

```
1  //Funkcija koja se pokreće klikom na „Submit“ gumb u obrazcu
2  const handleSubmit = async (e) => {
3    e.preventDefault();
4
5    try {
6  //Pozivanje „POST“ metode putem API rute za kreaciju rezervacije
7      const response = await fetch("http://localhost:3000/API/reviews", {
8        method: "POST",
9        headers: {
10         "Content-Type": "application/json",
11       },
12       body: JSON.stringify({
13         rating,
14         short,
15         description,
16         a_id: id,
17       }),
18     });
19
20     const data = await response.json();
21     console.log(data);
```

```
    if (data.success) {
      const userName = data.name;
      console.log("Review created by:", userName);
    } else {
      console.log("Review creation failed:", data.message);
    }
  } catch (error) {
    console.error(error);
  }
};
```

Kodni isječak 4.29: *Dio koda ReviewWindow.jsx komponente*

```
import { db } from "@/lib/database";
import { authOptions } from "@/lib/session";
import { getServerSession } from "next-auth";
import { NextResponse } from "next/server";

export const POST = async (request) => {
//Dohvaćanje podataka o prijavljenom korisniku
  const session = await getServerSession(authOptions);

  if (!session)
    return NextResponse.json({ message: "Not signed in" }, { status: 401 });

  const { _id } = session.user;
  const { rating, short, description, a_id } = await request.json();

//Poziva na model za kreaciju recenzije u database.js
  const newReview = new db.Review({
    rating,
    short,
    description,
    owner: _id,
    accomodation: a_id,
  });

  try {
    await newReview.save();
    return NextResponse.json(
      { message: "Review has been created!" },
      {
        status: 201,
      }
    );
  } catch (err) {
    return NextResponse.json(
      { message: err },
      {
        status: 500,
      }
```

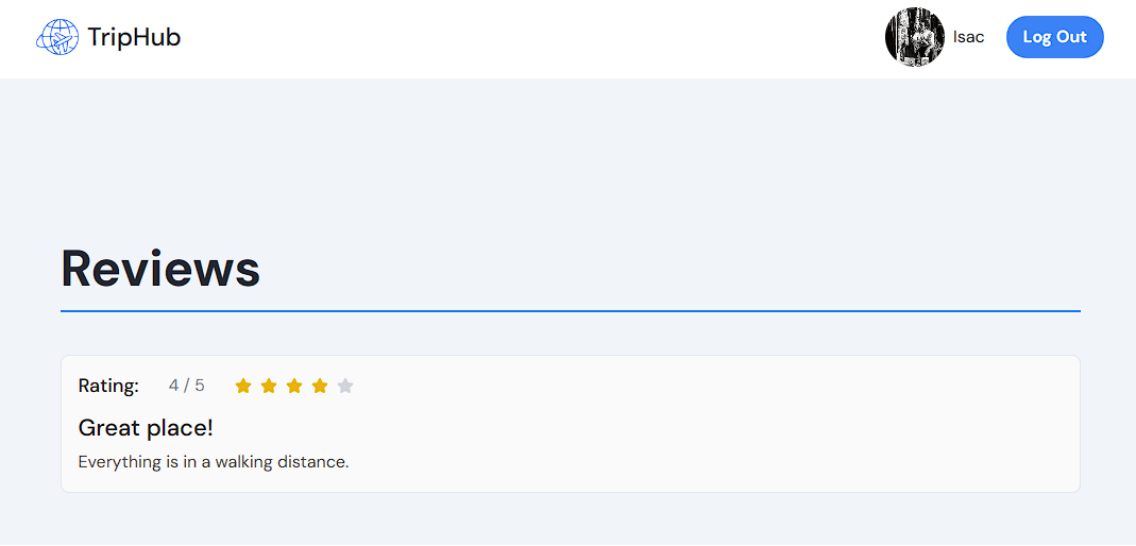
```
    );
  }
};
```

Kodni isječak 4.30: *API ruta za kreaciju recenzije*

```
function reviewModel() {
  const schema = new Schema(
    {
      rating: { type: String, required: true },
      short: { type: String, required: true },
      description: { type: String, required: true },
      owner: { type: Schema.Types.ObjectId, ref: "User", required: true },
      accomodation: {
        type: Schema.Types.ObjectId,
        ref: "Post",
        required: true,
      },
    },
    {
      timestamps: true,
    }
  );

  return mongoose.models.Review || mongoose.model("Review", schema);
}
```

Kodni isječak 4.31: *Dio koda database.js datoteke, model recenzije*

Povratkom na stranicu /hotel-details/id_smještaja/reveiws, vidljiva je korisnikova recenzija, te mu je uklonjena mogućnost dodavanja nove recenzije, slika 4.24. Lista recenzija za odabrani smještaj se dobiva pozivanjem *API* rute za dohvaćanje recenzija, kodni isječak 4.32. *API* ruta za dohvaćanje recenzija prikazana je u kodnom isječku 4.33. U kodnom isječku 4.34 se vidi kako sustav (pri učitavanju stranice /hotel-details/id_smještaja/reveiws) provjerava da li je prijavljeni korisnik ostavio recenziju za taj smještaj, te ako je, gumb „+ Add your review“ se ne iscrtava na stranici, slika 4.24.



Slika 4.23: *Stranica za pregled recenzija smještaja (Poslije recenzije)*

```
useEffect(() => {
  //Dohvaćanje recenzija smještajne jedinice
    fetch(`/API/reviews/${id}`)
      .then((response) => response.json())
      .then((data) => {
        setReviews(data);
      })
      .catch((error) => {
        console.error("Error fetching reviews:", error);
      });

    if (session?.user) {
      const { _id } = session.user;

      setUid(_id);
    }
  }, [id, session]);
```

Kodni isječak 4.32: *Dio koda Reveiw.jsx komponente (1)*

```
import { db } from "@/lib/database";
import { NextResponse } from "next/server";

export const GET = async (request, { params }) => {
  const { id } = params;
  const reviews = await db.Review.find({
    accomodation: id,
  }).populate("owner");

  try {
    return NextResponse.json(reviews, { status: 200 });
  } catch (err) {
    return new NextResponse(err.message, {
      status: 500,
    });
  }
};
```

Kodni isječak 4.33: *API ruta za dohvaćanje recenzija smještajne jedinice*

```
// Provjera je li korisnik već ostavio recenziju na smještaj
const userReviewed = reviews.some((review) => review.owner._id === uid);

// . . .
{session && !userReviewed ? (
// Prikaži "Add your review" gumb samo ako je korisnik prijavljen I nije ostavio
recenziju
          <Link href={`/hotel-details/${id}/write-a-review`}>
            <button className="ml-28 inline rounded-lg border bg-blue-500 p-2
align-super text-white">
              + Add your review
            </button>
          </Link>
        ) : null}
      </div>
      <div className="mb-10 mt-1 border-b-2 border-blue-500"></div>
      {reviews.map((review) => (
        <Review key={review._id} review={review} />
      ))}
// . . .
```

Kodni isječak 4.34: *Dio koda Reveiw.jsx komponente (2)*

5 ZAKLJUČAK

Odabranim tehnološkim stogom izgrađen je web sustav za kontrolu ponude i potražnje turističkog smještaja te demonstrirana suvremena web tehnologija korištena u razvoju ovog sustava. Razvoj je podrazumijevao izradu arhitekture baze podataka, dinamičkih elementa na strani klijenta te odgovarajućih mogućnosti na strani poslužitelja.

U radu je pokazano koliko je jednostavno koristiti Next.js - učinkovit React programski okvir fleksibilnog JavaScript-a; uparen s Tailwind CSS-om, te MongoDB - prilagodljivu NoSQL baza podataka. Spajanjem ovih tehnologija izvedena je skalabilna aplikacija koja osigurava veliku brzinu i integritet podataka.

Ovaj rad također pokazuje kako se industrija *online* rezervacija može razvijati i prilagođavati različitim potrebama klijenata i pružiti vrhunska iskustva. Izgradnjom web aplikacije koja koristi Next, Tailwind CSS i MongoDB za regulaciju ponude i potražnje turističkog smještaja, može se stjeći dojam o naprednim načinima kojima se mogu razvijati *online* rezervacijski sustavi kako tehnologija napreduje.

LITERATURA

[1] „Booking.com“. (2023.), adresa: *https://www.booking.com/* (pogledano 21.8.2023.)

[2] „Airbnb“. (2023.), adresa: *https://www.airbnb.com/* (pogledano 21.8.2023.)

[3] „Expedia“. (2023.), adresa: *https://www.expedia.com/* (pogledano 21.8.2023.)

[4] „React.js“. (2023.), adresa: *https://legacy.reactjs.org/* (pogledano 21.8.2023.)

[5] „Next.js“. (2023.), adresa: *https://nextjs.org/* (pogledano 21.8.2023.)

[6] „Tailwind CSS“. (2023.), adresa: *https://tailwindcss.com/* (pogledano 21.8.2023.)

[7] „Mongo DB“. (2023.), adresa: *https://www.mongodb.com/* (pogledano 21.8.2023.)

[8] „Mongo DB Atlas“. (2023.), adresa: *https://www.mongodb.com/atlas/database* (pogledano 21.8.2023.)

SAŽETAK

U ovom radu razvijen je i predtavljen web sustav koji omogućava oglašavanje, pretraživanje i rezervaciju turističkih smještajnih jedinica. Po uzoru na postojeće slične sustave, definirani su i implementirani odgovarajući slučajevi korištenja za cijeli kontekst. Sustav podržava više-parametarsko pretraživanje smještajnih jedinica te mogućnost komentiranja i ocjenjivanja. Posebna pažnja usmjerena je na oblikovanje korisničkog sučelja, s ciljem implementacije rješenja koje je responzivno, intuitivno i jednostavno za koristiti. U razvoju je korišten moderni tehnološki stog koji obuhvaća React.js, Next.js, Tailwind CSS i MongoDB.

Ključne riječi **– web aplikacija za rezervacije, React,js, Next.js, Tailwind CSS, MongoDB**

ABSTRACT

In this thesis, a web system was developed and presented that enables advertising, searching and booking of tourist accommodation. Based on existing similar systems, appropriate use cases for the target context were defined and implemented. The system supports a multi-parameter search accommodations and the possibility of commenting and rating. Special attention was paid to the design of the user interface, with the goal of implementing a solution that is responsive, intuitive, and easy to use. The development used a modern technology stack, which includes React.js, Next.js,Tailwind CSS and MongoDB.

Keywords **– booking web application, React,js, Next.js, Tailwind CSS, MongoDB**